Linköping studies in science and technology. Thesis.
No. 1652
Licentiate's Thesis

Sequential Monte Carlo for inference in nonlinear state space models

Johan Dahlin

REGLERTEKNIK
AUTOMATIC CONTROL
LINKÖPING

Division of Automatic Control
Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden
`http://www.control.isy.liu.se`
`johan.dahlin@isy.liu.se`

Linköping 2014

This is a Swedish Licentiate's Thesis.

Swedish postgraduate education leads to a Doctor's degree and/or a Licentiate's degree.
A Doctor's degree comprises 240 ECTS credits (4 years of full-time studies).
A Licentiate's degree comprises 120 ECTS credits,
of which at least 60 ECTS credits constitute the Licentiate's thesis.

Linköping studies in science and technology. Thesis.
No. 1652

Sequential Monte Carlo for inference in nonlinear state space models

Johan Dahlin

`johan.dahlin@isy.liu.se`
`www.control.isy.liu.se`
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping
Sweden

ISBN 978-91-7519-369-4 ISSN 0280-7971 LIU-TEK-LIC-2014:85

Printed by LiU-Tryck, Linköping, Sweden 2014

Denna avhandling tillägnas min familj!

Abstract

Nonlinear *state space models* (SSMs) are a useful class of models to describe many different kinds of systems. Some examples of its applications are to model; the volatility in financial markets, the number of infected persons during an influenza epidemic and the annual number of major earthquakes around the world. In this thesis, we are concerned with state inference, parameter inference and input design for nonlinear SSMs based on *sequential Monte Carlo* (SMC) methods.

The *state inference problem* consists of estimating some *latent variable* that is not directly observable in the output from the system. The *parameter inference problem* is concerned with fitting a pre-specified model structure to the observed output from the system. In *input design*, we are interested in constructing an input to the system, which maximises the information that is available about the parameters in the system output. All of these problems are analytically intractable for nonlinear SSMs. Instead, we make use of SMC to approximate the solution to the state inference problem and to solve the input design problem. Furthermore, we make use of *Markov chain Monte Carlo* (MCMC) and *Bayesian optimisation* (BO) to solve the parameter inference problem.

In this thesis, we propose new methods for parameter inference in SSMs using both Bayesian and maximum likelihood inference. More specifically, we propose a new proposal for the particle Metropolis-Hastings algorithm, which includes gradient and Hessian information about the target distribution. We demonstrate that the use of this proposal can reduce the length of the burn-in phase and improve the mixing of the Markov chain.

Furthermore, we develop a novel parameter inference method based on the combination of BO and SMC. We demonstrate that this method requires a relatively small amount of samples from the analytically intractable likelihood, which are computationally costly to obtain. Therefore, it could be a good alternative to other optimisation based parameter inference methods. The proposed BO and SMC combination is also extended for parameter inference in nonlinear SSMs with intractable likelihoods using approximate Bayesian computations. This method is used for parameter inference in a stochastic volatility model with α-stable returns using real-world financial data.

Finally, we develop a novel method for input design in nonlinear SSMs which makes use of SMC methods to estimate the expected information matrix. This information is used in combination with graph theory and convex optimisation to estimate optimal inputs with amplitude constraints. We also consider parameter estimation in ARX models with Student-t innovations and unknown model orders. Two different algorithms are used for this inference: reversible Jump Markov chain Monte Carlo and Gibbs sampling with sparseness priors. These methods are used to model real-world EEG data with promising results.

Populärvetenskaplig sammanfattning

Den värld som vi lever i är fylld av olika typer av system som kan beskrivas med matematiska modeller. Med hjälp av dessa modeller kan vi skapa oss en bättre förståelse av hur dessa system påverkas av sin omgivning, samt förutsäga hur de kommer att utvecklas över tid. Exempelvis kan man konstruera modeller av vädret baserad på kunskap om fysik samt tidigare års väder. Dessa modeller kan sedan användas för att exempelvis förutsäga om det kommer att regna imorgon. En annan typ av modeller kan användas för att prissätta olika typer av finansiella kontrakt, baserat på tidigare utfall och ekonomisk teori. Ett tredje exempel är modeller för att förutsäga antalet framtida jordbävningar i världen, givet historisk data och några modellantaganden.

Det som är gemensamt för dessa exempel är att de alla beskriver icke-linjära dynamiska system, alltså system som utvecklas över tid. I denna avhandling är vi intresserade av att bygga *icke-linjära tillståndsmodeller* av dynamiska system med hjälp av *datadrivna statistiska inferensmetoder*. Med hjälp av dessa modeller och metoder är det möjligt att kombinera teoretiska kunskaper som beskrivs av en modellstruktur med observationer från systemet. Den senare informationen kan användas för att bestämma värdet på några okända parametrar i modellen, detta kallas även för *parameterskattning*. Ett annat vanligt förekommande problem är *tillståndsskattning*, där vi vill bestämma värdet på någon dynamisk storhet i systemet som inte kan observeras direkt. Det huvudsakliga problemet med detta angreppssätt är att inget av dessa skattningsproblem kan lösas exakt med hjälp av analytiska metoder.

Istället nyttjar vi approximativa metoder som baseras på *statistiska simuleringer* för att lösa problemen. Så kallade *sekventiella Monte Carlo*-metoder används för att approximera lösningen till tillståndsskattningsproblemet. Detta görs med hjälp av en dator som simulerar en stor mängd hypoteser (även kallade partiklar) om hur systemet fungerar. De hypoteser som stämmer väl överens med det verkliga observerade beteendet sparas och förfinas i nästkommande steg. De övriga hypoteserna tas bort från simuleringen för att fokusera beräkningskraften på de hypoteser som har relevans enligt den observerade informationen. Parameterskattningsproblemet kan lösas approximativt med liknande metoder som även de bygger på simulering.

I denna avhandling arbetar vi främst med att försöka förbättra parameterskattningsmetoder som bygger på partikel Markovkedje-Monte Carlo (MCMC) och Bayesiansk optimering. De förbättringar som vi föreslår leder till att skattningarna kan beräknas snabbare än tidigare genom att bättre ta tillvara på den information som finns i det observerade datamaterialet. Dessa metoder används för att exempelvis prissätta finansiella optioner. Vi föreslår även en ny algoritm för att skapa insignaler till system så att observationerna som erhålls från systemet, innehåller så mycket information som möjligt om de okända parametrarna. Slutligen demonstrerar vi hur man kan använda MCMC-metoder för parameterskattning i modeller som kan användas för att beskriva EEG-signaler.

Acknowledgments

I could never have written my thesis without the help, love and support from all the people around me, now and in the past. I will now spend a few lines to express my gratitude to some of the special people that have helped me along the way.

First and foremost, I would like to acknowledge the help and support from my supervisors. My main supervisor Prof. Thomas Schön has always provided me with encouragement, ideas, new people to meet, new opportunities to grab and challenges to help me grow. Also, my co-supervisor Dr. Fredrik Lindsten has always been there for me with answers to my questions, explanations to my problems and ideas/suggestions for my work. I am most impressed by your enthusiasm and dedication in the roles as supervisors. I think that you both have surpassed what can be expected from a supervisor and for this I am extremely grateful. I could never have done this without you! Thank you for your support and all our time running together in Stanley park, Maastricht, Warwick and Söderköping!

Furthermore, to be able to write a good thesis you require a good working environment. Prof. Svante Gunnarsson and Ninna Stensgård are two very important persons in this effort. Thank you for all your kindness, support and helpfulness in all matters; small and large. I would also like to acknowledge Dr. Henrik Tidefelt and Dr. Gustaf Hendeby for constructing and maintaining the LaTeX-template in which this thesis is written. My brother Fredrik Dahlin, Lic. Joel Kronander, Jonas Linder, Dr. Fredrik Lindsten, Prof. Thomas Schön, Andreas Svensson, Patricio Valenzuela and Johan Wågberg have all helped with proof-reading and with suggestions to improve the thesis. All remaining errors are entirely my own.

I am most grateful for the financial support from the project *Probabilistic modelling of dynamical systems* (Contract number: 621-2013-5524) funded by the Swedish Research Council.

Another aspect of the atmosphere at work is all my wonderful colleagues. Especially, my room mate Lic. Michael Roth, who helps with his experience and advice about balancing life as a PhD student. Thank you for sharing the room with me and watering our plants while I am away! I would also like to thank Jonas Linder for all our adventures and our nice friendship together both at and outside of work. Manon Kok and I started in the group at the same time and have helped eachother over the years. Thank you for your positive attitude and for always being open for discussions. Also, I would like to acknowledge the wonderful BBQs and other funny things that Lic. Sina Khoshfetrat Pakazad arranges and lets me participate in!

Furthermore, I would like to thank my remaining friends and colleagues at the group. Especially, (without any specific ordering) Dr. Christian Lyzell, Lic. Ylva Jung, Isak Nielsen, Hanna Nyquist, Dr. Daniel Petersson, Clas Veibäck and Dr. Emre Özkan for all the funny things that we have done together. This includes everything from beer tastings, wonderful food in France and fitting as many of RTs PhD students into a jacuzzi as possible to hitting some shallows on open sea with a canoe, cross country skiing in Chamonix and eating food sitting on the floor in a Japanese restaurant with screaming people everywhere. You have given me the most wonderful memories and times!

I have also had the benefit of working with a lot of talented and enthusiastic researchers during my time in academia. I would first like to thank all my fellow students at Teknisk Fysik, Umeå University for a wonderful time as an undergraduate student. Also, Prof. Anders Fällström, Dr. Konrad Abramowicz, Dr. Ulf Holmberg and Dr. Sang Hoon Lee have inspired, supported and encouraged me to pursue a PhD degree, something I have not (a.s.) regretted!

Also, my wonderful (former) colleagues at the Swedish Defence Research Agency (FOI) have supported and encouraged me to continue climbing the educational ladder. Thank you, Dr. Pontus Svensson, Dr. Fredrik Johansson, Dr. Tove Gustavi and Christian Mårtensson. Finally, I would like to thank all my co-authors during my time at Linköping University for some wonderful and fruitful collaborations: Daniel Hultqvist, Lic. Daniel Jönsson, Lic. Joel Kronander, Dr. Fredrik Lindsten, Cristian Rojas, Dr. Jakob Roll, Prof. Thomas Schön, Fredrik Svensson, Dr. Jonas Unger, Patricio Valenzuela, Prof. Mattias Villani and Dr. Adrian Wills.

Furthermore, Lic. Joel Kronander and Dr. Jonas Unger have helped me with the images from the computer graphics application in the introduction. Prof. Mattias Villani together with Stefan Laséen and Vesna Crobo at Riksbanken helped with the economics application and made the forecasts from RAMSES II.

Finally, I am most grateful to my loving family and close relatives for their support all the way from childhood until now and beyond. I love you all every much! Also, my friends are always a great source for support, inspiration and encouragement both at work and in life! My life would be empty and meaningless without you all! I hope that we all can spend some more time now when my thesis is done and when new challenges awaits! Because, what would life be without challenges, meeting new people and spending time with your loved ones. Empty.

Linköping, May 2014
Johan Dahlin

Contents

II Publications

Notation

Probability

Notation	Meaning
$\xrightarrow{a.s.}$	Almost sure convergence.
$\xrightarrow{d}$	Convergence in distribution.
$\xrightarrow{p}$	Convergence in probability.
$\delta_z(\mathrm{d}x)$	Dirac point mass located at $x = z$.
$\mathbb{P}$,$\mathbb{E}$,$\mathbb{V}$	Probability, expectation and covariance operators.
$\sim$	Sampled from or distributed according to.

Statistical distributions

Notation	Meaning
$\mathcal{A}(\alpha, \beta, \gamma, \eta)$	α-stable distribution with stability α, skewness β, scale γ and location η.
$\mathcal{B}(p)$	Bernoulli distribution with success probability p.
$\mathcal{N}(\mu, \sigma^2)$	Gaussian (normal) dist. with mean μ and variance σ^2
$\mathcal{G}(\alpha, \beta)$	Gamma distribution with rate α and shape β.
$\mathcal{IG}(\alpha, \beta)$	Inverse Gamma distribution with rate α and shape β.
$\mathcal{P}(\lambda)$	Poisson distribution with mean λ.
$\mathcal{U}(a, b)$	Uniform distribution on the interval $[a, b]$.

Operators and other symbols

Notation	Meaning
I_d	$d \times d$ identity matrix.
$\triangleq$	Definition.
$\mathsf{diag}(v)$	Diagonal matrix with the vector v on the diagonal.
$\nabla f(x)$	Gradient of $f(x)$.
$\nabla^2 f(x)$	Hessian of $f(x)$.
$\mathbb{I}$	Indicator function.
$\mathsf{det}(A)$, $\lvert A \rvert$	Matrix determinant of A.
A^{-1}	Matrix inverse of A.
$\mathsf{tr}(A)$	Matrix trace of A.
$A^\top$	Matrix transpose of A.
$v^2 = vv^\top$	Outer product of the vector v.
$a_{n:m}$	Sequence $\{a_n, a_{n+1}, \ldots, a_{m-1}, a_m\}$, for $m > n$.
$\mathsf{sign}(x)$	Sign of x.
$\mathsf{supp}(f)$	Support of the function f, $\{x : f(x) > 0\}$.

Statistical quantities

Notation	Meaning
$\mathcal{I}(\theta)$	Expected information matrix evaluated at θ.
$\mathcal{L}(\theta)$	Likelihood function evaluated at θ.
$\ell(\theta)$	Log-likelihood function evaluated at θ.
$\widehat{\theta}_{\mathrm{ML}}$	Maximum likelihood parameter estimate.
$\mathcal{J}(\theta)$	Observed information matrix evaluated at θ.
$\widehat{\theta}$	Parameter estimate.
$p(\theta \mid y_{1:T})$	Parameter posterior distribution.
$p(\theta)$	Parameter prior distribution.
θ	Parameter vector, $\theta \in \Theta \subseteq \mathbb{R}^d$.
$\mathcal{S}(\theta)$	Score function evaluated at θ.

Algorithmic quantities

Notation	Meaning
$a_t^{(i)}$	Ancestor of particle i at time t.
Z	Normalisation constant.
$x_t^{(i)}$	Particle i at time t.
$R_\theta(x_t \mid x_{0:t-1}, y_t)$	Particle proposal kernel.
$W_\theta(x_t, x_{t-1})$	Particle weighting function.
$q(\theta)$	Proposal distribution.
$\pi(\theta)$	Target distribution.
$\gamma(\theta)$	Unnormalised target distribution.
$w_t^{(i)}, \widetilde{w}_t^{(i)}$	Un- and normalised weight of particle i at time t.

Abbreviations

Abbreviation	Meaning
a.s.	Almost surely (with probability 1).
ABC	Approximate Bayesian computations.
ACF	Autocorrelation function.
AIS	Adaptive importance sampling.
APF	Auxiliary particle filter.
AR(p)	Autoregressive process of order p.
ARD	Automatic relevance determination.
ARCH(p)	AR conditional heteroskedasticity process of order p.
ARX(p)	Autoregressive exogenous process of order p.
BIS	Bidirectional importance sampling.
BO	Bayesian optimisation.
bPF	Bootstrap particle filter.
BRDF	Bidirectional reflectance distribution function.
CDF	Cumulative distribution function.
CLT	Central limit theorem.
CPI	Consumer price index.
DSGE	Dynamic stochastic general equilibrium.
EB	Empirical Bayes.
EEG	Electroencephalography.
EI	Expected improvement.
EM	Environment map.
ESS	Effective sample size.
faPF	Fully-adapted particle filter.
FFBSm	Forward-filtering backward-smoothing.
FFBSi	Forward-filtering backward-simulation.
FL	Fixed-lag (particle smoother).
GARCH(p,q)	Generalised ARCH process of order (p, q).
GPO	Gaussian process optimisation.
GPU	Graphical processing unit.
HMM	Hidden Markov model.
IACT	Integrated autocorrelation time.
IBL	Image-based lightning.
IID	Independent and identically distributed.
IS	Importance sampling.
KDE	Kernel density estimate/estimator.
LGSS	Linear Gaussian state space.
LTE	Light transport equation.
MCMC	Markov chain Monte Carlo.
MH	Metropolis-Hastings.
MIS	Multiple importance sampling.
ML	Maximum likelihood.
MLE	Maximum likelihood estimator.
MLT	Metropolis light transport.
MSE	Mean square error.

Abbreviations (cont.)

Abbreviation	Meaning
PD	Positive definite.
PDF	Probability density function.
PMF	Probability mass function.
PF	Particle filter.
PG	Particle Gibbs.
PI	Probability of improvement.
PMCMC	Particle Markov chain Monte Carlo.
PMH	Particle Metropolis-Hastings.
PMH0	Marginal particle Metropolis-Hastings.
PMH1	PMH using first order information
PMH2	PMH using first and second order information
PS	Particle smoother.
RJ-MCMC	Reversible jump Markov chain Monte Carlo.
RTS	Rauch-Tung-Stribel.
RW	Random walk.
SIS	Sequential importance sampling.
SIR	Sequential importance sampling and resampling.
SLLN	Strong law of large numbers.
SMC	Sequential Monte Carlo.
SPSA	Simultaneous perturbation stochastic approximation.
SSM	State space model.
UCB	Upper confidence bound.

Part I

Background

1 Introduction

Science is the art of collecting and organising knowledge about the universe by tested explanations and validated predictions. Therefore, modelling the world using *observations* and *statistical inference* is an integral part of the scientific method. The resulting statistical models can be used to *describe* certain observed phenomena or to *predict* new phenomena and future behaviours. An example of the former is to discover new physical models by generalising from observed data using *induction*. Examples of prediction applications are to validate scientific theories or to forecast the future GDP of Sweden, the probability of rainfall tomorrow and the number of earthquakes during the coming year. We discuss some of the details of these problems in the following chapters.

This thesis is concerned with *building dynamical models from recorded observations*, i.e. models of systems that evolves over time. The observations are combined with past experiences and established scientific theory to build models using statistical tools. Here, we limit ourselves to discussing *nonlinear state space models* (SSMs), where most of the structure is known beforehand except a few parameters. A fairly general class of SSMs can be expressed as

$$\begin{aligned} x_{t+1} | x_t &\sim f_\theta(x_{t+1} | x_t), \\ y_t | x_t &\sim g_\theta(y_t | x_t), \end{aligned}$$

where x_t and y_t denotes an unobserved (latent) state and an observation from the system at time t. Here, $f_\theta(x_{t+1}|x_t)$ and $g_\theta(y_t|x_t)$ denotes two Markov kernels parametrised by an unknown static real-valued parameter vector θ. Applications of this class of SSMs can be found in almost all of the natural sciences and most of the social sciences. Some specific examples are biology (Wilkinson, 2011), control (Ljung, 1999), epidemiology (Keeling and Rohani, 2008) and finance (Tsay, 2005; Hull, 2009).

The procedure to determine the parameter vector θ from the observations $y_{1:T}$ is referred to as *parameter inference* and this problem is analytically intractable for nonlinear SSMs. Another related problem is the *state inference* problem, where we would like to determine the value of x_t given the information in the observations $y_{1:T}$ or $y_{1:t}$. This problem is also analytically intractable for most SSMs.

Instead, we make use of *statistical simulation* methods to estimate the parameters and states. As the name suggests, these methods are based on simulating many (often thousands or millions) of hypotheses referred to as *particles*. The particles that match the recorded observations are retained and the others are discarded. This procedure can be repeated in a sequential manner, where the solution of the problem is obtained as the solution to many subproblems.

This idea is the basis for the *sequential Monte Carlo* (SMC) methods that are an integral part of the methods that we consider for state inference in SSMs. For example, the marginal filtering distribution $p_\theta(x_t|y_{1:t})$ can be approximated by an *empirical distribution*,

$$\widehat{p}_\theta(\mathrm{d}x_t|y_{1:t}) = \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta_{x_t^{(i)}}(\mathrm{d}x_t),$$

where $x_t^{(i)}$ and $\widetilde{w}_t^{(i)}$ denotes the particle i and its corresponding (normalised) weight obtained from the SMC algorithm. Here, $\delta_z(\mathrm{d}x)$ denotes a Dirac point mass located at $x = z$. The empirical distribution summarises all the information that is contained within the data about the value of the latent state at some time t. This information can then be used by other methods for solving the parameter inference problem in SSMs.

The number of particles N in the SMC method controls both the accuracy of the empirical distributions and the computational cost. That is, a high accuracy requires many particles which incurs a high computational cost and this results in a trade-off between accuracy and speed. Also, we make use of the SMC algorithm within some iterative parameter inference methods to solve the state inference problem at each iteration. Therefore, we would like to limit the number of iterations required by the parameter inference methods to obtain an accurate parameter estimate with a reasonable computational cost.

These ideas are the two main themes of this thesis. The first theme is *to propose some developments to improve the efficiency of existing parameter inference methods based on SMC algorithms.* The second theme is to *extend some of the current methods for linear SSMs to nonlinear SSMs by making use of SMC algorithms.* We return to the contributions of this thesis in Section 1.2.

1.1 Examples of applications

In this section, we give two examples of problems in which computational statistical methods based on the SMC algorithm are useful. In the first example, we use a model to forecast the future development of the Swedish economy given past experience and economical theory. In the second example, we use a model constructed from the physics of light transport to render photorealistic images.

1.1.1 Predicting GDP growth

The economy of a country is a complex system with an emergent behaviour depending on the actions of many interacting heterogeneous agents. In an economy, these agents correspond to consumers, companies, banks, politicians, governmental agencies, other countries, etc. As such, these agents may or may not act rationally to their situation and could therefore be difficult to model on an individual level.

As a result, economical models mostly deal with the aggregated behaviour of many homogeneous agents, i.e. rational utility maximising agents with a common valuation of goods and services. These models can be used to produce forecasts, to gain understanding about the current situation in the economy and simulate the result of different policy decisions. An example of this could be to study the impact of changing the repo rate on the unemployment level and the GDP growth of the economy.

For this purpose, many central banks are today using *dynamic stochastic general equilibrium* (DSGE) models (An and Schorfheide, 2007; Del Negro and Schorfheide, 2004) for modelling the economy of a country. The outputs from these models are various macroeconomic quantities, such as GDP growth, unemployment rate, inflation, etc. The general structure is given by economic theory, but there are some unknown parameters that needs to be inferred from data.

Riksbanken (the Swedish central bank) has developed a DSGE model called *the Riksbank Aggregate Macromodel for Studies of the Economy of Sweden II* (RAMSES II) (Adolfson et al., 2013, 2007a) to model the Swedish economy. Essentially, RAMSES II is a nonlinear SSM with 12 outputs, about 40 latent states and about 65 unknown parameters.

For computational convenience, only the log-linearised version of the full model is considered in most of the analysis. Consequently, Kalman filtering methods can be used to solve the state inference problem. The parameter inference problem is solved using a *Metropolis-Hastings* (MH) algorithm, where the proposal is a multivariate Gaussian distribution with the covariance matrix given by the inverse of the observed information matrix at the posterior mode. The information matrix is estimated using Quasi-Newton optimisation algorithms such as the BFGS algorithm (Nocedal and Wright, 2006). For more details, see Adolfson et al. (2007b).

In Chapters 3 and 4, we discuss alternative methods that could solve the state and parameter inference problem in the original nonlinear version of RAMSES II. For related treatments using SMC and MCMC in combination with DSGE models, see Flury and Shephard (2011), Fernández-Villaverde and Rubio-Ramírez (2007) and Amisano and Tristani (2010).

In Figure 1.1, we give an example of how the RAMSES II model can be used for forecasting the changes in GDP, the consumer price index with fixed interest rates

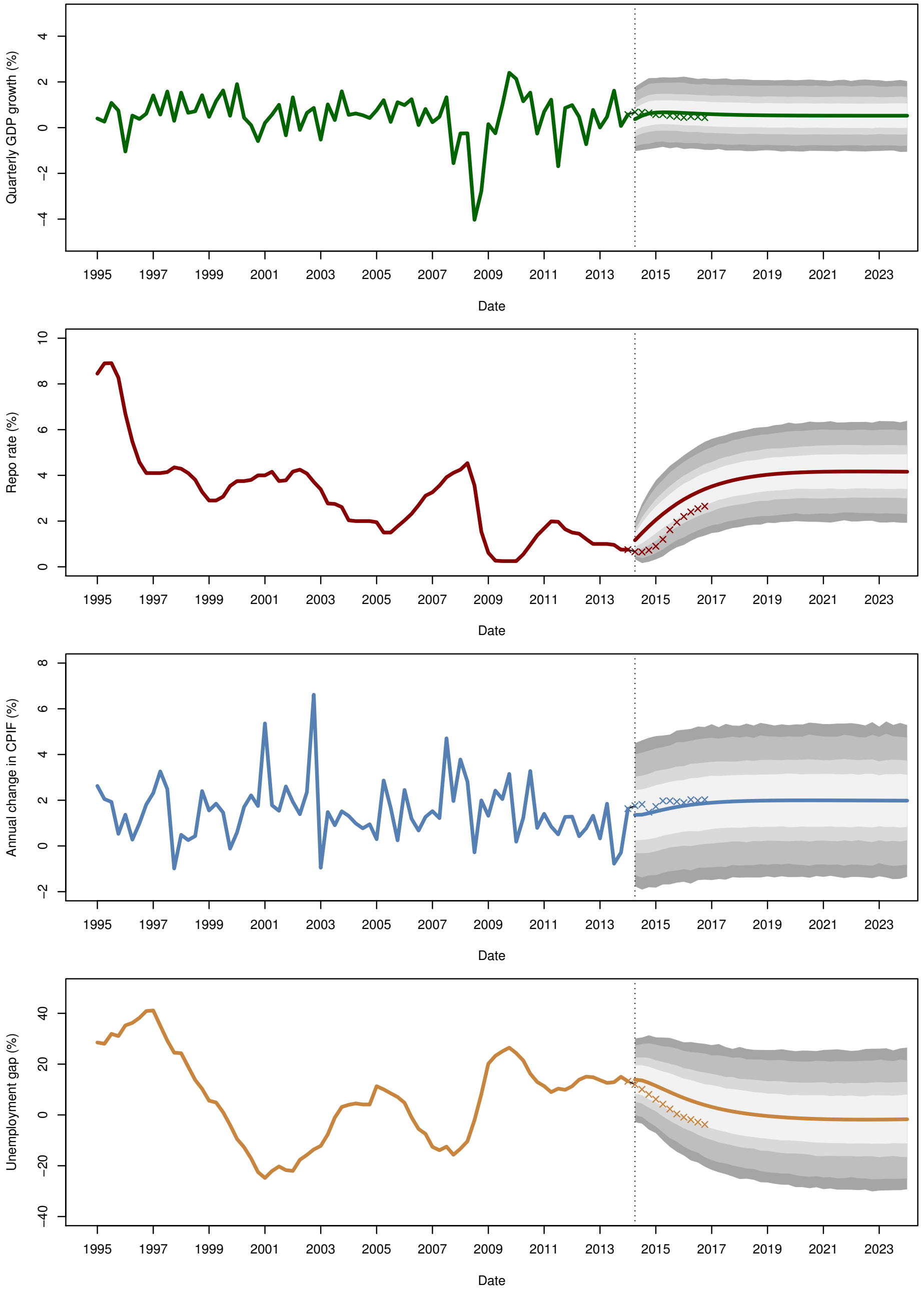

Figure 1.1: *The quarterly GDP growth (green), the repo rate (red), the annual change in the CPIF (blue) and unemployment gap (orange). The historical data is presented for each variable up until the dotted vertical lines. The predicted means are presented with the 50%, 70%, 90% and 95% credibility intervals (from lighter to darker gray). The published forecasts from Riksbanken are presented as crosses. The data and predictions are obtained by the courtesy of Riksbanken.*

(CPIF) inflation, the repo rate and the unemployment gap[1] in Sweden. We first make use of some historical data (up until the dotted vertical line) to estimate the parameters and the latent states. The resulting model is then used to forecast the predictive posterior mean (solid lines) with some credibility intervals (gray areas). These predictive means are used by Riksbanken together with experts and other models to construct forecasts of the economy. This forecast is presented by crosses and differs from the output of the RAMSES II model.

1.1.2 Rendering photorealistic images

To simulate light transport, we make use of a geometrical optics model, developed during centuries of research in the field of physics (Hecht, 2013). By the use of this models, we can simulate how light behaves in different environments and make use of this to *render* images. A popular related application is to add objects into an image or video sequence that were not present when the scene was captured. In this section, we shortly discuss how to do this using the so-called *image-based lighting* (IBL) method (Debevec, 1998; Pharr and Humphreys, 2010).

To make use of the IBL method, we require an panoramic image of the real scene captured using a *high dynamic range* (HDR) camera. This type of camera can record much larger variations in brightness than a standard camera, which are needed to capture all the different light sources within the scene. The resulting image is referred to as an *environment map* (EM). In IBL, this panoramic image serves as the source of illumination when rendering images, allowing the real objects to cast shadows and interact with the virtual objects. Secondly, we need geometrical models of the objects that we would like to add into the scene. Finally, we require a mathematical description of the optical properties of the materials in these objects to be able to simulate how the light scatters over their surfaces.

The IBL method combines all of this information using the *light transport equation* (LTE), which is a physical model of how light rays propagates through space and reflects off surfaces. The LTE model cannot be solved analytically, but it can be approximated using methods related to SMC algorithms. To see how this can be done, consider a cartoon of the setup presented in Figure 1.2. In the first step, a set of light rays originating from a pixel in the image plane is generated. We then track how these rays bounces around in the scene until they finally hit the EM. The colours and brightnesses of the EM in these locations are recorded and used to compute the resulting colour and brightness of the pixel in the image plane. This approach is repeated for all the pixels in the image plane.

However in the real world, there are infinitely many rays that bounces around in the scene before they hit the pixels in the image plan. As a result, it is computationally infeasible to simulate all the light rays and all the bounces in the scene. Instead, there are methods to select only the light rays which contributes the most to

[1] The unemployment gap is the amount (in percent) that the GDP must increase to be able to achieve full employment of the work force. The decrease of the unemployment gap is an important aim of financial policy making and can be achieved (in Keynesian economic theory) by increasing public spending or lowering taxes.

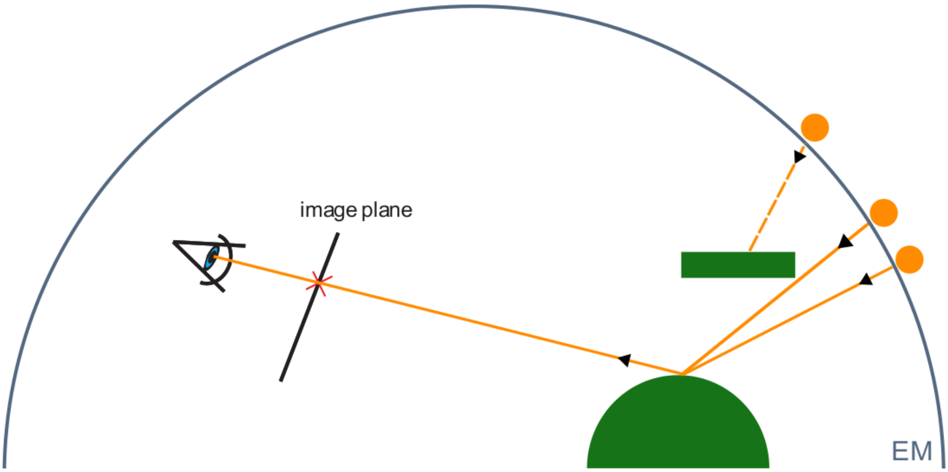

Figure 1.2: *The basic setup of ray tracing underlying photorealistic image synthesis. The colour and brightness of a pixel in the image plane is determined by the EM, the geometry of the scene and the optical properties of the objects.*

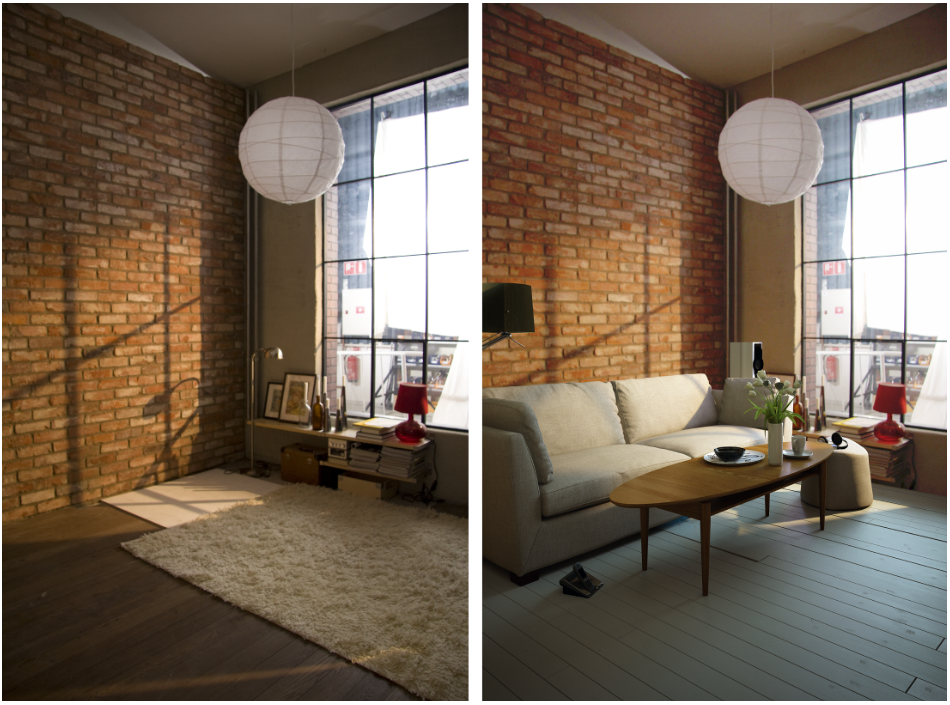

Figure 1.3: *The scene before (left) and after (right) the rendering using a version of the IBL method. The image is taken from Unger et al. (2013) and is used with courtesy of the authors.*

the brightness and colour each pixel in the image plane. This can be done in a similar manner to the methods discussed later in Section 4.2 resulting in the *Metropolis light transport* (MLT) algorithm (Veach and Guibas, 1997). The basis for these methods is to solve the LTE problem by simulating different hypotheses and improving them in analogue with SMC methods. That is, light rays that hit bright areas of the EM are kept and modified, whereas rays that does hit the EM in dim regions or bounces around too long are discarded.

Note that, it can take several days to render a single image using the IBL algorithm, even when only allowing for a few bounces and light rays per pixel in the image plane. This problem grows even further when we would like to render a sequence of images. A possible solution could be to start from the solution from the previous frame and adapt it to the new frame. If the EMs are similar, this could lead to a decrease in the total computational cost. We return to this idea in Section 3.5.

In Figure 1.3, we present an example from Unger et al. (2013) of a scene before (left) and after (right) it is rendered in a computer by the use of the IBL method and the methods discussed in Section 4.2. Note that, in the final result we have added several photorealistic objects into the scene such as the sofa, the table and have also changed the floor in the room. These methods are used in many entertainment applications to create special effects and to modify scenes in post production. Furthermore, they are useful in rendering images of scenes that are difficult or costly to build in the real-world. Some well-known companies (such as IKEA and Volvo) make use of these methods for digital design and advertisements as a cost effective alternative to traditional photography.

1.2 Thesis outline and contributions

This thesis is divided into two parts. In Part I, we give some examples of models and applications together with an introduction to the different computational inference methods that are used. In Part II, we present edited versions of some published peer-reviewed papers and unpublished technical reports.

Part I - Background

In this part, we begin by introducing the SSM and provide some additional examples of its real-world applications in Chapter 2. Furthermore, we introduce two different statistical paradigms for parameter inference problems in SSMs: the maximum likelihood (ML) based approach and the Bayesian approach. Finally, we discuss why computational methods are required for estimating the solution to these problems.

In Chapter 3, we review the state inference problem in SSMs and discuss the use of SMC methods for approximating the solution to these problems. We also discuss the use of SMC algorithms for other classes of models, which includes the computer graphics example discussed in Section 1.1.2.

Chapter 4 is devoted to discussing the parameter inference problem for nonlinear

SSMs. We begin by giving an overview of different parameter inference methods and then discuss Markov chain Monte Carlo (MCMC) and Bayesian optimisation (BO) in more detail. The former can be used for Bayesian parameter inference and the latter can be used for ML or maximum a posteriori (MAP) based parameter inference.

We conclude Part I by Chapter 5 which contains a summary of the contributions of the thesis together with some general conclusions and possible avenues for future work.

Part II - Publications

The main part of this thesis is the compilation of five papers published in peer-reviewed conference proceedings or as technical reports. These papers contain the main contributions of this thesis:

- In Paper A, we develop a novel particle MCMC algorithm that combines the Particle Metropolis-Hastings (PMH) with the Langevin dynamics. The resulting algorithm explores the posterior distribution more efficient then the marginal PMH algorithm, is invariant to affine transformations of the parameter vector and reduces the length of the the burn-in. As a consequence, the proposed algorithm requires less iterations, which makes it more computationally efficient than the marginal PMH algorithm.

- In Paper B, we develop a novel algorithm for ML parameter inference by combining ideas from BO with SMC for log-likelihood estimation. The resulting algorithm is computationally efficient as it requires less samples from the log-likelihood compared with other popular methods.

- In Paper C, we extend the combination of BO and SMC to parameter inference in nonlinear SSMs with intractable likelihoods. Computationally costly approximate Bayesian computations (ABC) are used to approximate the likelihood. We illustrate the proposed algorithm for parameter inference in stochastic volatility model with α-stable returns using real-world data.

- In Paper D, we develop a novel algorithm for input design in nonlinear SSMs, which can handle amplitude constraints on the input. The proposed method makes use of SMC for estimating the expected information matrix. The algorithm performs well compared with some other methods in the literature and decreases the variance of the parameter estimates with almost an order of magnitude.

- In Paper E, we propose two algorithms for parameter inference in ARX models with Student-t innovations which includes automatic model order selection. These methods makes use of *reversible jump MCMC* (RJMCMC) and the Gibbs sampler together with sparseness priors to estimate the model order and the parameter vector. We illustrate the use of the proposed algorithm to model real-world EEG data with promising results.

Here, we present an abstract of each paper together with an account of the contribution of the author of this thesis.

Paper A

Paper A of this thesis is an edited version of,

> J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Pre-print*, 2014b. arXiv:1311.0686v2.

which is a combination and development of the two earlier publications

> J. Dahlin, F. Lindsten, and T. B. Schön. Second-order particle MCMC for Bayesian parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014a. (accepted for publication).

> J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis Hastings using Langevin dynamics. In *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013a.

Abstract: PMH allows for Bayesian parameter inference in nonlinear state space models by combining MCMC and particle filtering. The latter is used to estimate the intractable likelihood. In its original formulation, PMH makes use of a marginal MCMC proposal for the parameters, typically a Gaussian random walk. However, this can lead to a poor exploration of the parameter space and an inefficient use of the generated particles.

We propose two alternative versions of PMH that incorporate gradient and Hessian information about the posterior into the proposal. This information is more or less obtained as a byproduct of the likelihood estimation. Indeed, we show how to estimate the required information using a fixed-lag particle smoother, with a computational cost growing linearly in the number of particles. We conclude that the proposed methods can: (i) decrease the length of the burn-in phase, (ii) increase the mixing of the Markov chain at the stationary phase, and (iii) make the proposal distribution scale invariant which simplifies tuning.

Contributions and background: The author of this thesis contributed with the majority of the work including the design, the implementation, the numerical illustrations and the written presentation.

Paper B

Paper B of this thesis is an edited version of,

> J. Dahlin and F. Lindsten. Particle filter-based Gaussian process optimisation for parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication).

Abstract: We propose a novel method for maximum-likelihood-based parameter inference in nonlinear and/or non-Gaussian state space models. The method is an iterative procedure with three steps. At each iteration a particle filter is used to estimate the value of the log-likelihood function at the current parameter iterate. Using these log-likelihood estimates, a surrogate objective function is created by utilizing a Gaussian process model. Finally, we use a heuristic procedure to obtain a revised parameter iterate, providing an automatic trade-off between exploration and exploitation of the surrogate model. The method is profiled on two state space models with good performance both considering accuracy and computational cost.

Contributions and background: The author of this thesis contributed with the majority of the work including the design, the implementation, the numerical illustrations and the written presentation.

Paper C

Paper C of this thesis is an edited version of,

> J. Dahlin, T. B. Schön, and M. Villani. Approximate inference in state space models with intractable likelihoods using Gaussian process optimisation. Technical Report LiTH-ISY-R-3075, Department of Electrical Engineering, Linköping University, Linköping, Sweden, April 2014c.

Abstract: We propose a novel method for MAP parameter inference in nonlinear state space models with intractable likelihoods. The method is based on a combination of BO, SMC and SBC. SMC and ABC are used to approximate the intractable likelihood by using the similarity between simulated realisations from the model and the data obtained from the system. The BO algorithm is used for the MAP parameter estimation given noisy estimates of the log-likelihood.

The proposed parameter inference method is evaluated in three problems using both synthetic and real-world data. The results are promising, indicating that the proposed algorithm converges fast and with reasonable accuracy compared with existing methods.

Contributions and background: The author of this thesis contributed with the majority of the work including the design, the implementation, the numerical illustrations and the written presentation. This contribution resulted from the participation in the course *Bayesian learning* given by Prof. Mattias Villani at Linköping University during the autumn of 2013.

Paper D

Paper D of this thesis is an edited version of,

> P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. A graph/particle-based method for experiment design in nonlinear systems. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication).

Abstract: We propose an extended method for experiment design in nonlinear state space models. The proposed input design technique optimizes a scalar cost function of the information matrix, by computing the optimal stationary probability mass function (PMF) from which an input sequence is sampled. The feasible set of the stationary PMF is a polytope, allowing it to be expressed as a convex combination of its extreme points. The extreme points in the feasible set of PMFs can be computed using graph theory.

Therefore, the final information matrix can be approximated as a convex combination of the information matrices associated with each extreme point. For nonlinear SSMs, the information matrices for each extreme point can be computed by using particle methods. Numerical examples show that the proposed technique can be successfully employed for experiment design in nonlinear SSMs.

Contributions and background: This is an extension of the work presented in Valenzuela et al. (2013) and a result of the cooperation with the Department of Automatic Control at the Royal Institute of Technology (KTH). The author of this thesis designed and implemented the algorithm for estimating the expected information matrix and the Monte Carlo method for estimating the optimal input. The corresponding sections in the paper were also written by the author of this thesis.

Paper E

Paper E of this thesis is an edited version of,

> J. Dahlin, F. Lindsten, T. B. Schön, and A. Wills. Hierarchical Bayesian ARX models for robust inference. In *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*, Brussels, Belgium, July 2012b.

Abstract: Gaussian innovations are the typical choice in most ARX models but using other distributions such as the Student-t could be useful. We demonstrate that this choice of distribution for the innovations provides an increased robustness to data anomalies, such as outliers and missing observations. We consider these models in a Bayesian setting and perform inference using numerical procedures based on MCMC methods. These models include automatic order determination by two alternative methods, based on a parametric model order and a sparseness prior, respectively. The methods and the advantage of our choice of innovations are illustrated in three numerical studies using both simulated data and real EEG data.

Contributions and background: The author of this thesis contributed to parts of the the implementation, generated most of the numerical illustrations and wrote the sections covering the numerical illustrations and the conclusions in the paper. The EEG data was kindly provided by Eline Borch Petersen and Thomas Lunner at Eriksholm Research Centre, Oticon A/S, Denmark.

1.3 Publications

Published work of relevance to this thesis are listed below in reverse chronological order. Items marked with ⋆ are included in Part II of this thesis.

⋆ J. Dahlin, T. B. Schön, and M. Villani. Approximate inference in state space models with intractable likelihoods using Gaussian process optimisation. Technical Report LiTH-ISY-R-3075, Department of Electrical Engineering, Linköping University, Linköping, Sweden, April 2014c.

⋆ J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Pre-print*, 2014b. arXiv:1311.0686v2.

⋆ J. Dahlin and F. Lindsten. Particle filter-based Gaussian process optimisation for parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication).

J. Dahlin, F. Lindsten, and T. B. Schön. Second-order particle MCMC for Bayesian parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014a. (accepted for publication).

⋆ P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. A graph/particle-based method for experiment design in nonlinear systems. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication)

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis Hastings using Langevin dynamics. In *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013a.

⋆ J. Dahlin, F. Lindsten, T. B. Schön, and A. Wills. Hierarchical Bayesian ARX models for robust inference. In *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*, Brussels, Belgium, July 2012b.

Other published works related to but not included in the thesis are:

J. Kronander, J. Dahlin, D. Jönsson, M. Kok, T. B. Schön, and J. Unger. Real-time Video Based Lighting Using GPU Raytracing. In *Proceedings of the 2014 European Signal Processing Conference (EUSIPCO)*, Lisbon, Portugal, September 2014a. (submitted, pending review).

J. Kronander, T. B. Schön, and J. Dahlin. Backward sequential Monte Carlo for marginal smoothing. In *Proceedings of the 2014 IEEE Statistical Signal Processing Workshop (SSP)*, Gold Coast, Australia, July 2014b. (accepted for publication).

D. Hultqvist, J. Roll, F. Svensson, J. Dahlin, and T. B. Schön. Detection and positioning of overtaking vehicles using 1D optical flow. In *Proceedings of the IEEE Intelligent Vehicles (IV) Symposium*, Dearborn, MI, USA, June 2014. (accepted for publication).

J. Dahlin and P. Svenson. Ensemble approaches for improving community detection methods. *Pre-print*, 2013. arXiv:1309.0242v1.

J. Dahlin, F. Lindsten, and T. B. Schön. Inference in Gaussian models with missing data using Equalisation Maximisation. *Pre-print*, 2013b. arXiv:1308.4601v1.

J. Dahlin, F. Johansson, L. Kaati, C. Mårtensson, and P. Svenson. A Method for Community Detection in Uncertain Networks. In *Proceedings of International Symposium on Foundation of Open Source Intelligence and Security Informatics 2012*, Istanbul, Turkey, August 2012a.

J. Dahlin and P. Svenson. A Method for Community Detection in Uncertain Networks. In *Proceedings of 2011 European Intelligence and Security Informatics Conference*, Athens, Greece, August 2011.

2

Nonlinear state space models and statistical inference

In this chapter, we introduce the SSM and give some motivating examples of different applications in which the model is used. We also review ML inference and Bayesian inference in connection with SSMs. Interested readers are referred to Douc et al. (2014), Cappé et al. (2005), Ljung (1999), Shumway and Stoffer (2010) and Brockwell and Davis (2002), for more detailed accounts of the topics covered here.

2.1 State space models and inference problems

An SSM or *hidden Markov model* (HMM) consists of a pair of discrete-time stochastic processes[1] $x_{0:T} \triangleq \{x_t\}_{t=0}^{T}$ and $y_{1:T} \triangleq \{y_t\}_{t=1}^{T}$. Here, $x_t \in \mathsf{X} \in \mathbb{R}^n$ denotes the *latent state* and $y_t \in \mathsf{Y} \in \mathbb{R}^m$ denotes the *observation* obtained from the system at time t.

The latent state is modelled as a Markov chain with initial state $x_0 \sim \mu(x_0)$ and the transition kernel $f_\theta(x_{t+1}|x_t, u_t)$. Furthermore, we assume that the observations are mutually independent given the latent states and have the conditional observation density $g_\theta(y_t|x_t, u_t)$. In both kernels, $\theta \in \Theta \subseteq \mathbb{R}^d$ denotes the static parameter vector of the Markov kernel and u_t denotes a known input to the system. With these definitions, we can write the SSM on the compact form

$$x_0 \sim \mu(x_0), \tag{2.1a}$$

$$x_{t+1}|x_t \sim f_\theta(x_{t+1}|x_t, u_t), \tag{2.1b}$$

$$y_t|x_t \sim g_\theta(y_t|x_t, u_t), \tag{2.1c}$$

[1] In this thesis, we do not make any difference in notation between a random variable and its realisation. This is done to ease the notation.

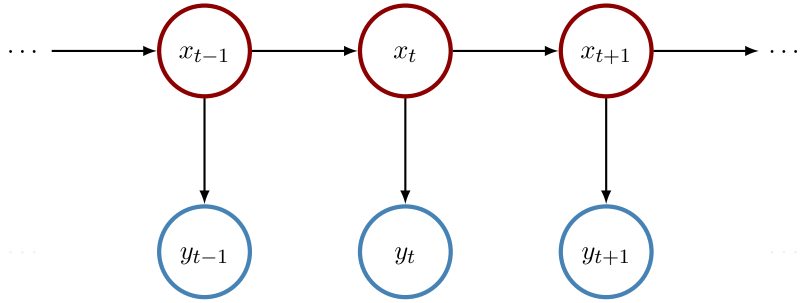

Figure 2.1: *Graphical model of an SSM with latent process (red) and observed process (blue).*

which we make use of in this thesis. This is a fairly general class of models and can be used to model both nonlinear and non-Gaussian systems.

Another popular description of stochastic models like the class of SSMs is *graphical model* (Murphy, 2012; Bishop, 2006). The corresponding graphical representation of an SSM is depicted in Figure 2.1, where the latent state process is presented in red and the observed process in blue. From this graphical model, we see that the state x_t is only dependent on the last state x_{t-1} due to the *Markov property* inherent in the model. That is, all the information about the past is summarised in the state at time $t-1$. Also, we see that the observations are mutually independent given the states, as there are no arrows directly between two observations.

In this thesis, we are interested in two different inference problems connected to SSMs: (i) the *state inference* problem and (ii) the *parameter inference* problem. The first problem is to infer the density of the latent state process given the observations and the model. If we are interested in the current state given all the observations up until now, we would like to estimate the *marginal filtering density* $p_\theta(x_t|y_{1:t})$. We return to this and other related problems in Chapter 3. There, we define the problem in mathematical terms and present numerical methods designed to approximate the filtering and smoothing densities.

The second problem is to infer the values of the parameter θ given the set of observations $y_{1:T}$ and the model structure encoded by the Markov transition kernels $f_\theta(x_{t+1}|x_t)$ and $g_\theta(y_t|x_t)$. It turns out that we have to solve the state inference problem as a part of the parameter inference problem. We later return to the mathematical formulation of this problem in the ML setting in Section 2.3 and in the Bayesian setting in Section 2.4. These problems are analytically intractable and cannot be computed in closed-form. Therefore, we present computational methods based on sampling methods for parameter inference in Chapter 4.

2.2 Some motivating examples

SSMs have been successfully applied in various areas for modelling dynamical systems. In this section, we give three examples from different research fields and connect them with the inference problems discussed in the previous section. The first model is taken from finance, where we would like to model the real-valued latent volatility given some stock or exchange rate data. In the second model, we would like to make predictions of the number of annual major earthquakes. The third model is taken from meteorology, where we would like predict the probability of rain fall during the coming days. However, we start the the well-known linear Gaussian state space (LGSS) model, which we make use of as a benchmark throughout the thesis.

2.2.1 Linear Gaussian model

Consider the scalar LGSS model[2],

$$x_{t+1}|x_t \sim \mathcal{N}\Big(x_{1+1}; \phi x_t + \gamma u_t, \sigma_v^2\Big), \tag{2.2a}$$

$$y_t|x_t \sim \mathcal{N}\Big(y_t; x_t, \sigma_e^2\Big), \tag{2.2b}$$

where the parameter vector is $\theta = \{\phi, \gamma, \sigma_v, \sigma_e\}$. Here, ϕ describes the persistence of the state and $\{\sigma_v, \sigma_e\}$ controls the noise levels. In this model, we have added an optional input $u_{1:T}$ to the system, which is scaled by the parameter γ. Here, we require that $\phi \in (-1, 1) \subset \mathbb{R}$ to obtain a stable system and that $\{\sigma_v, \sigma_e\} \in \mathbb{R}_+^2$ as they correspond to standard deviations.

The state inference problem can be solved exactly for this model using Kalman filters and smoothers (Kailath et al., 2000). This is a result of that the model is linear and only includes Gaussian kernels. Due to this property, we make use of the model as a benchmark problem for some of the algorithms reviewed in this thesis. Comparing the methods that we develop for the nonlinear SSMs with the LGSS model can reveal important properties of the algorithms and help with insights about how to calibrate them. The Kalman methods can also be used to estimate the log-likelihood, the score function and the information matrix, which are important quantities in the ML parameter inference problem discussed in Section 2.3.

2.2.2 Volatility models in econometrics and finance

Nonlinear SSMs are often encountered in econometric and financial problems, where we would like to e.g. model the variations in the log-returns of a stock or an index. The log-returns are calculated by $y_t = \log(s_t/s_{t-1})$, where s_t denotes the price of some financial asset at time t. The variations in the log-returns can be seen as the instantaneous standard deviation and is referred to as the *volatility*.

[2] This model is also known as ARX(1) in noise, where ARX(1) denotes an exogenous autoregressive process of order 1.

The volatility plays an important role in the famous Black-Scholes pricing model (Black and Scholes, 1973). In this model, the log-returns are assumed to follow a Brownian motion with independent and identically distributed IID increments distributed according to some Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, where σ denotes the volatility. Therefore, the volatility is an important component when calculating the price of options and other financial instrument based on the Black-Scholes model. For a discussion of how the volatility is used for pricing options, see Hull (2009), Björk (2004) and Glasserman (2004). For a more extensive treatment of financial time series and alternative inference methods for volatility models, see Tsay (2005).

In Figure 2.2, we present the closing prices and daily log-returns for the the NASDAQ OMX Stockholm 30 Index during a 14 year period. Note, the large drops in the closing prices (middle) in the period around the years 2001, 2008 and 2011 in connection with the most recent financial crises (shocks). At these drops, we see that the log-returns are quite volatility, as they vary much between consecutive days. During other periods, the volatility is quite low and the log-returns does not vary much between consecutive days. These variations in the volatility are referred to as *volatility clustering* in finance.

Also, from the QQ-plots we see that the log-returns are heavy-tailed and clearly non-Gaussian with large deviations from theoretical quantiles in the tail behaviour. All these features (and some other) are known as *stylized facts* (Cont, 2001). In this section, we present three different volatility models that tries to capture different aspects of the observed properties of financial data. A complication is that the resulting inference problems become more challenging as when try to capture more and more of the stylized facts. This results in a trade-off between accuracy of the model and the computational complexity of the resulting inference problems.

A common theme for all the models considered here, is that they are all based on a (slowly) varying random walk-type model of the volatility. This model can be motivated by the volatility clustering behaviour, i.e. the underlying volatility varies slowly and gives rise to volatility clustering. Furthermore, the log-returns are modelled as a non-stationary white noise process where the variance is determined by the latent volatility. This corresponds quite well with the log-returns presented in the upper part of Figure 2.2. For a more through discussion about different volatility models, see Mitra (2011) and Kim et al. (1998).

The first model is the generalised autoregressive conditional heteroskedasticity (GARCH) model (Bollerslev, 1986), which is a generalisation of the ARCH model (Engle, 1982). Here, we consider the GARCH(1,1) in noise model given by

$$h_{t+1}|x_t, h_t = \alpha + \beta x_t^2 + \gamma h_t, \tag{2.3a}$$

$$x_{t+1}|x_t, h_t = \mathcal{N}\Big(x_{t+1}; 0, h_{t+1}\Big), \tag{2.3b}$$

$$y_t|x_t = \mathcal{N}\Big(y_t; x_t, \tau^2\Big), \tag{2.3c}$$

where the parameter vector is $\theta = \{\alpha, \beta, \gamma, \tau\}$ with the constraints $\{\alpha, \beta, \gamma, \tau\} \in$

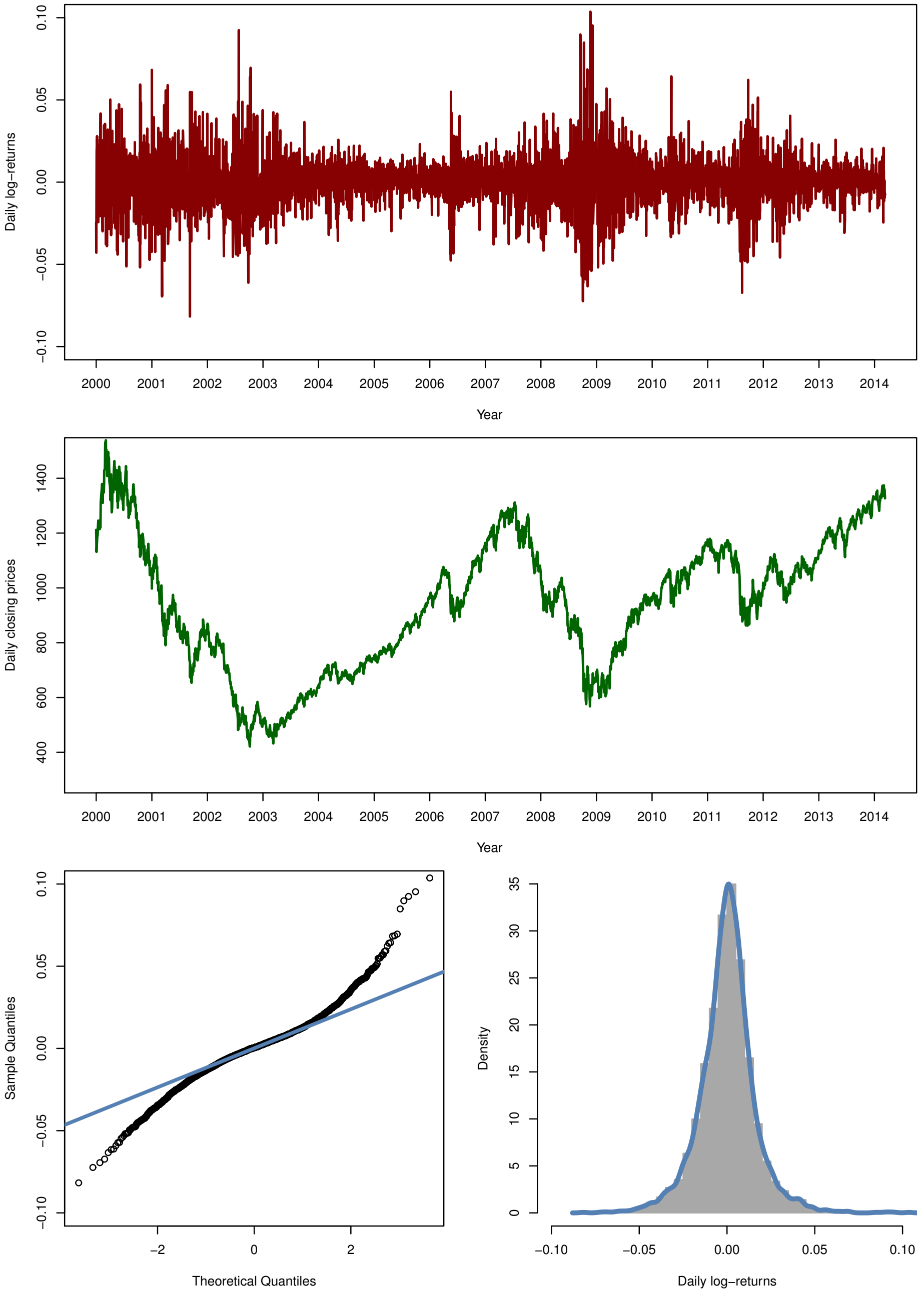

Figure 2.2: *Daily log-returns (upper) for the NASDAQ OMX Stockholm 30 Index from 2000-01-04 to 2014-03-14. The daily closing prices (middle), QQ-plot of the log-returns (lower left) and histogram of the log-returns and the kernel density estimate (KDE) (lower right) are also presented.*

$\mathbb{R}^4_+$ and $\beta + \gamma \in (0, 1) \subset \mathbb{R}$ for stability. In this model, the current log-return is taken into account when computing the volatility at the next time step. This construction tries to capture the volatility clustering.

The second model is the Hull-White stochastic volatility (HWSV) model[3] (Hull and White, 1987) given by

$$x_{t+1}|x_t \sim \mathcal{N}\Big(x_{t+1}; \mu + \phi(x_t - \mu), \sigma^2\Big), \tag{2.4a}$$

$$y_t|x_t \sim \mathcal{N}\Big(y_t; 0, \beta^2 \exp(x_t)\Big), \tag{2.4b}$$

where the parameter vector is $\theta = \{\mu, \phi, \sigma, \beta\}$ with the constraints $\{\mu, \sigma, \beta\} \in \mathbb{R}^3_+$ and $\phi \in (-1, 1) \subset \mathbb{R}$ for stability. There are many variants of the HWSV model that includes correlations between the noise sources (called leverage models) and outliers in the form of jump processes. See Chib et al. (2002) and Jacquier et al. (2004) for more information.

One problem with HWSV is that the Gaussian observation noise in some cases cannot fully capture the heavy-tail behaviour found in real-world data. Instead, e_t is often assumed to be simulated from a Student-t distribution, which has heavier tails than the Gaussian distribution. Another modification is to assume that e_t is generated from an α-stable distribution, which can model both large outliers in the data and non-symmetric log-returns. This results in the third model, the stochastic volatility model with symmetric α-stable returns (SVα) Casarin (2004) given by

$$x_{t+1}|x_t \sim \mathcal{N}\Big(x_{t+1}; \mu + \phi x_t, \sigma^2\Big), \tag{2.5a}$$

$$y_t|x_t \sim \mathcal{A}\Big(y_t; \alpha, 0, \exp(x_t/2), 0\Big), \tag{2.5b}$$

where the parameter vector is $\theta = \{\mu, \phi, \sigma, \alpha\}$ and the constraints $\{\mu, \sigma\} \in \mathbb{R}^2_+$, $\alpha \in (0, 2] \setminus \{1\} \subset \mathbb{R}$ and $\phi \in (-1, 1) \subset \mathbb{R}$ for stability. Here, $\mathcal{A}(\alpha, 0, 1, 0)$ denotes a symmetric α-stable distribution[4] with stability parameter α. For this distribution, we cannot evaluate $g_\theta(y_t|x_t)$ and this results in problems when inferring the states and parameter vector of the model. In Paper C, we apply ABCs for solving this problem.

As previously discussed, the inference problem in volatility models is mainly state inference, which requires a model and hence results in the need for parameter inference. For example, the state estimate can be used as the volatility estimate for option pricing. Also, the parameter vector of the model can be used to analyse if the log-return are symmetric and heavy-tailed or the persistence of the underlying volatility process.

[3] A similar model is used for modelling the glacial varve thickness (the thickness of the clay collected within the glacial) in Shumway and Stoffer (2010). This model is obtain by replacing the noise in (2.4b) with gamma distributed noise, i.e. $e_t \sim \mathcal{G}(\alpha^{-1}, \alpha)$ for some parameter $\alpha \in \mathbb{R}_+$.

[4] See Appendix A of Paper C for a brief summary about α-stable distributions and their properties. For a more detailed presentation, see Nolan (2003).

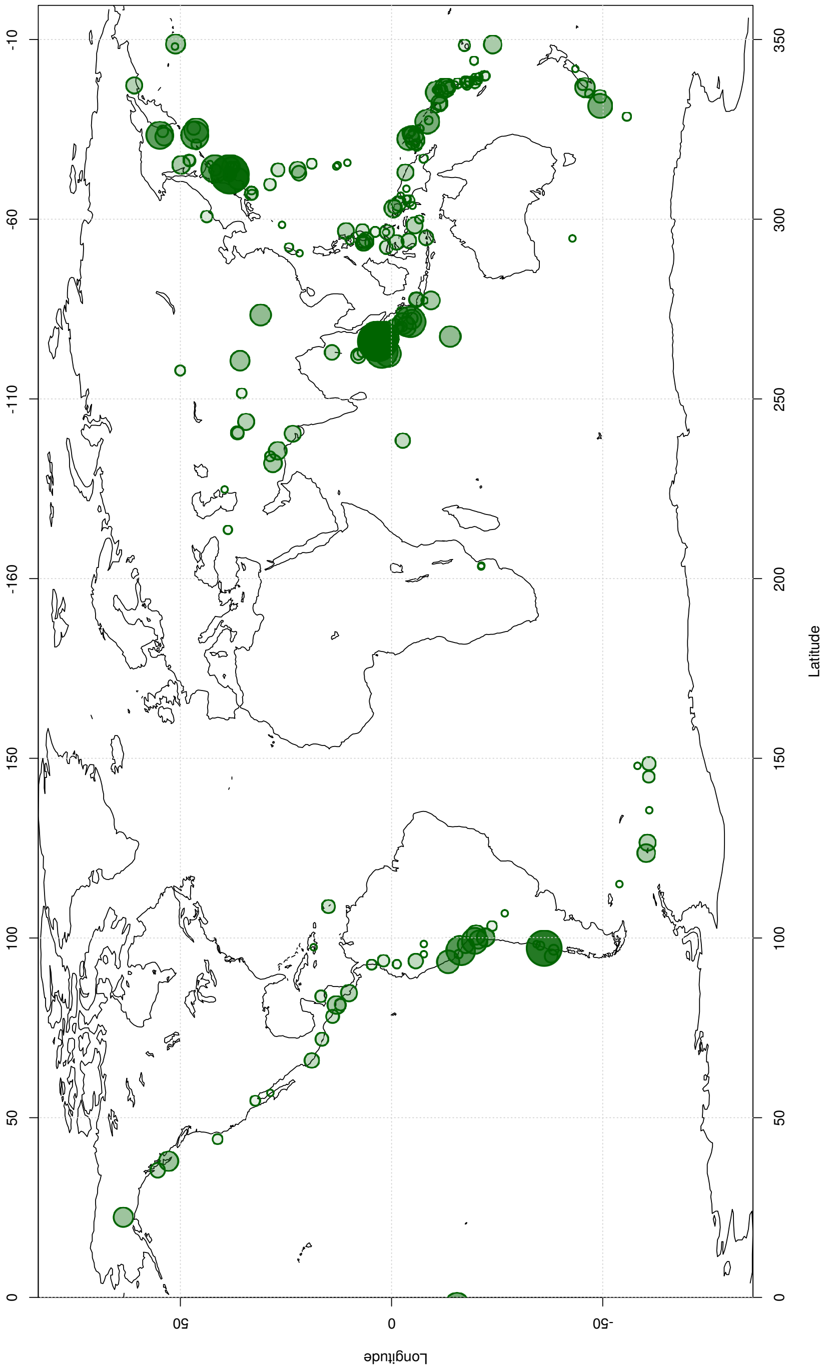

Figure 2.3: *The major earthquakes in the world between* 2000 *and* 2013*. The size of the circle is proportional to the relative magnitude of the earthquake.*

2.2.3 Earthquake count model in geology

Another application of SSMs is to model the number of major (with magnitude 7 or higher on the Richter scale) earthquakes each year. As a motivation for the model, we consider some real-world data from the *Earthquake Data Base System of the U.S. Geological Survey*[5]. The data describes the number of major earthquakes around the world between the years 1900 and 2013. In Figure 2.3, we present the locations of the major earthquakes during the period 2000 to 2013. In Figure 2.4, we present the annual number of major earthquakes with some exploratory plots. From the data, we see that the number of earthquakes is clearly correlated, similar to the clustering behaviour found in the volatility models from the previous application. That is, a year with many earthquakes is likely to be followed by another year with high earthquake intensity. The reason for this follows quite intuitive as pointed out in Langrock (2011) since earthquakes are due to stresses in the tectonic plates of the Earth. Therefore, the underlying process which model these stresses should be slowly varying over the time span of years.

Due to the autocorrelation in the number of earthquakes, it is reasonable to assume that the intensity is determined by some underlying latent variable. Therefore, we can model the number of major earthquakes as an SSM. To this end, we use the model[6] from Zeger (1988) and Chan and Ledolter (1995), which assumes that the number of earthquakes y_t is a Poisson distributed variable (corresponding a positive integer or count data). Also, we assume that the mean of the Poisson process λ_t follows an AR(1) process,

$$\log(\lambda_t) - \mu = \phi\Big(\log\big(\lambda_{t-1}\big) - \mu \Big) + \sigma_v v_t,$$

where v_t denotes a standard Gaussian random variable. By introducing $x_t = \log(\lambda_t) - \mu$ and $\beta = \exp(\mu)$, we obtain the SSM

$$x_{t+1} | x_t \sim \mathcal{N}\Big(x_{t+1}; \phi x_t, \sigma_v^2 \Big), \tag{2.6a}$$

$$y_t | x_t \sim \mathcal{P}\Big(y_t; \beta \exp(x_t) \Big), \tag{2.6b}$$

where the parameter vector is $\theta = \{\phi, \sigma_v, \beta\}$ with the constraints $\phi \in (-1, 1) \subset \mathbb{R}$ and $\{\sigma_v, \beta\} \in \mathbb{R}_+^2$. Here, $\mathcal{P}(\lambda)$ denotes a Poisson distributed variable with mean λ. That is, the probability of $k \in \mathbb{N}$ earthquakes during year t is given by the probability mass function (PMF),

$$\mathbb{P}[N_t = k] = \frac{\exp(-\lambda)\lambda^k}{k!}.$$

The inference problem in this type of model could be to determine the underlying intensity of the process (the state). This information could be useful in making predictions of the future number of major earthquakes.

[5] This data can be accessed from `http://earthquake.usgs.gov/earthquakes/eqarchives/`.

[6] This type of Poisson count model can also be used to model the number of yearly polio infections (Zeger, 1988) and the number of transactions per minute of a stock on the market (Fokianos et al., 2009).

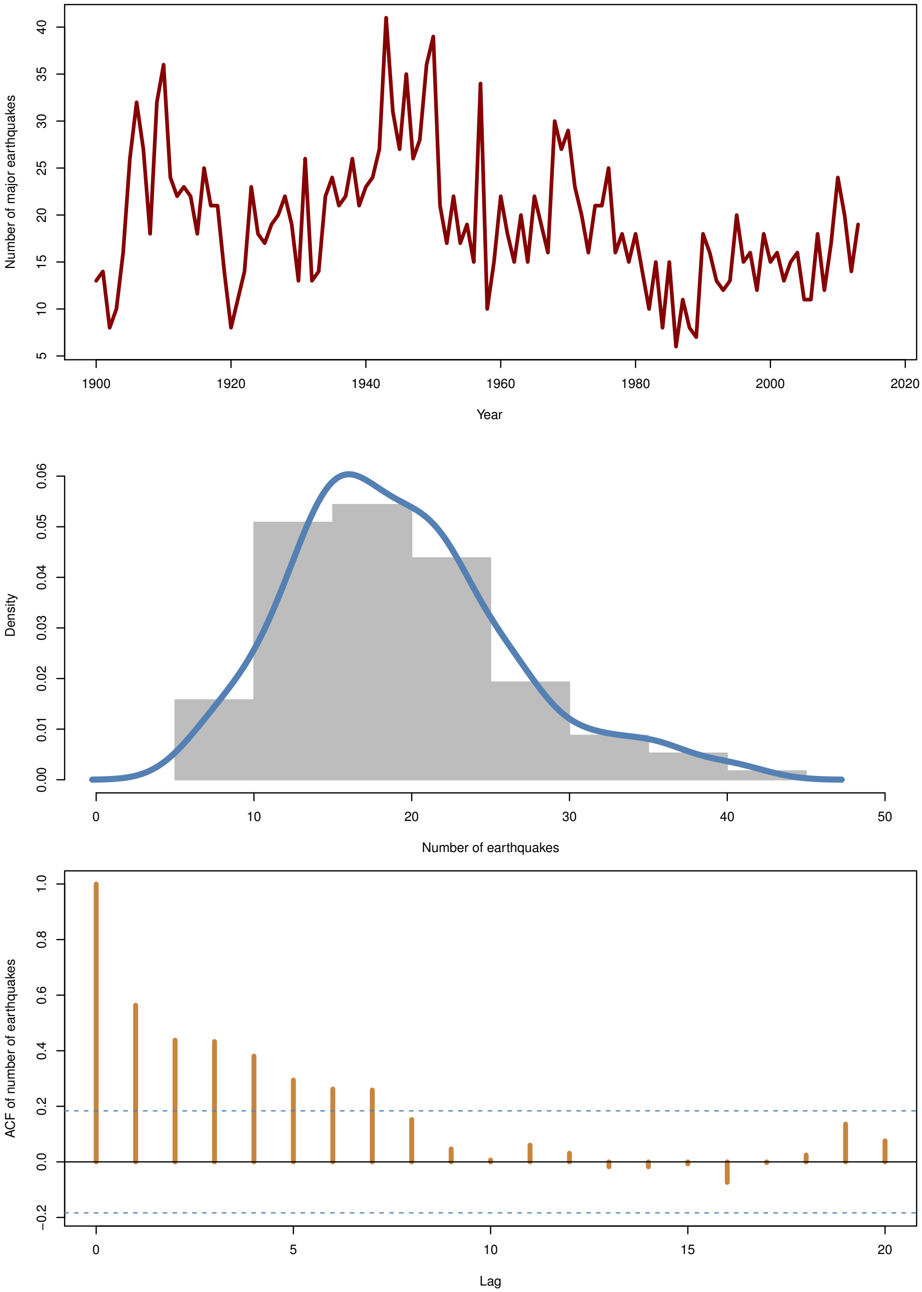

Figure 2.4: *Upper: number of annual major earthquakes (with magnitude 7 or higher on the Richter scale) during the period between the years 1900 and 2013. Middle: the corresponding histogram with the KDE (blue). Lower: the estimated autocorrelation function (ACF).*

2.2.4 Daily rainfall models in meteorology

A common problem in meteorology and weather forecasting is to estimate the probability and amount of rainfall. In practice, this problem is often split into two subproblems, each with its own model. The first model determines the probability of rainfall and the second model determines the amount of rainfall. For more information, see Srikanthan and McMahon (1999) and Woolhiser (1992). Here, we consider the problem to construct a model to determine the probability of rainfall given historic data of rainfall in the region of interest. It is also possible to use user data from the Internet to predict the probability of rainfall in some region. An example of this is to make use of some collected data from Twitter, see Naesseth (2012) for more information about this approach.

We first consider some real-world data from the Swedish weather service (SMHI) collected daily at Malmslätt near Linköping during the period between the years 1952 and 2002. The data is presented in Figure 2.5 as the daily probability of rainfall (upper) and the average daily amount of rainfall (middle) calculated per week. We also present the ACF of rainfall, which indicates that there is a correlation between rainy and non-rainy days. Furthermore, the probability of rainfall seems to follow a cyclic behaviour with a high probability during the latter part of the year. The amount of rainfall also varies with a peak around week 30.

To construct a model of the probability of rainfall, we follow the insights from the previous analysis and review the model proposed by Langrock and Zucchini (2011). To account for the autocorrelation in the rainfall, we make use of a latent process to describe the persistence of the weather. This construction can be used to create a rough model to account for the structure of low pressure weather systems which passes over a period of days. Hence, this can be seen as a short term model of the probability of rainfall. From practical knowledge, we also know that the probability of rainfall is connection with the season of the year. This cyclic behaviour was also seen in the weather data from Malmslätt. Therefore, we assume that there also exists a cyclical part of the latent process and that this together with the short term persistence determines the probability of rainfall.

Furthermore, we assume that the output from the model is a binary variable $y_t \sim \mathcal{B}(p_t)$ from a Bernoulli distribution with success probability p_t. That is, the variable assumes the value 1 with probability p_t (rain falls during day t) and the value 0 with probability $1 - p_t$ (no rain falls during day t). The final model on SSM form is given by

$$h_t = \sum_{k=1}^{2} \alpha_k \cos\left(\frac{2k\pi t}{365}\right) + \sum_{k=1}^{2} \beta_k \sin\left(\frac{2k\pi t}{365}\right), \tag{2.7a}$$

$$x_{t+1}|x_t \sim \mathcal{N}\Big(x_{t+1}; \phi x_t, \sigma_v^2\Big), \tag{2.7b}$$

$$y_t \sim \mathcal{B}\bigg(\frac{\exp(\mu + x_t + h_t)}{1 + \exp(\mu + x_t + h_t)}\bigg), \tag{2.7c}$$

where the parameter vector is $\theta = \{\phi, \sigma_v, \mu, \alpha_1, \alpha_2, \beta_1, \beta_2\}$ with the constraints

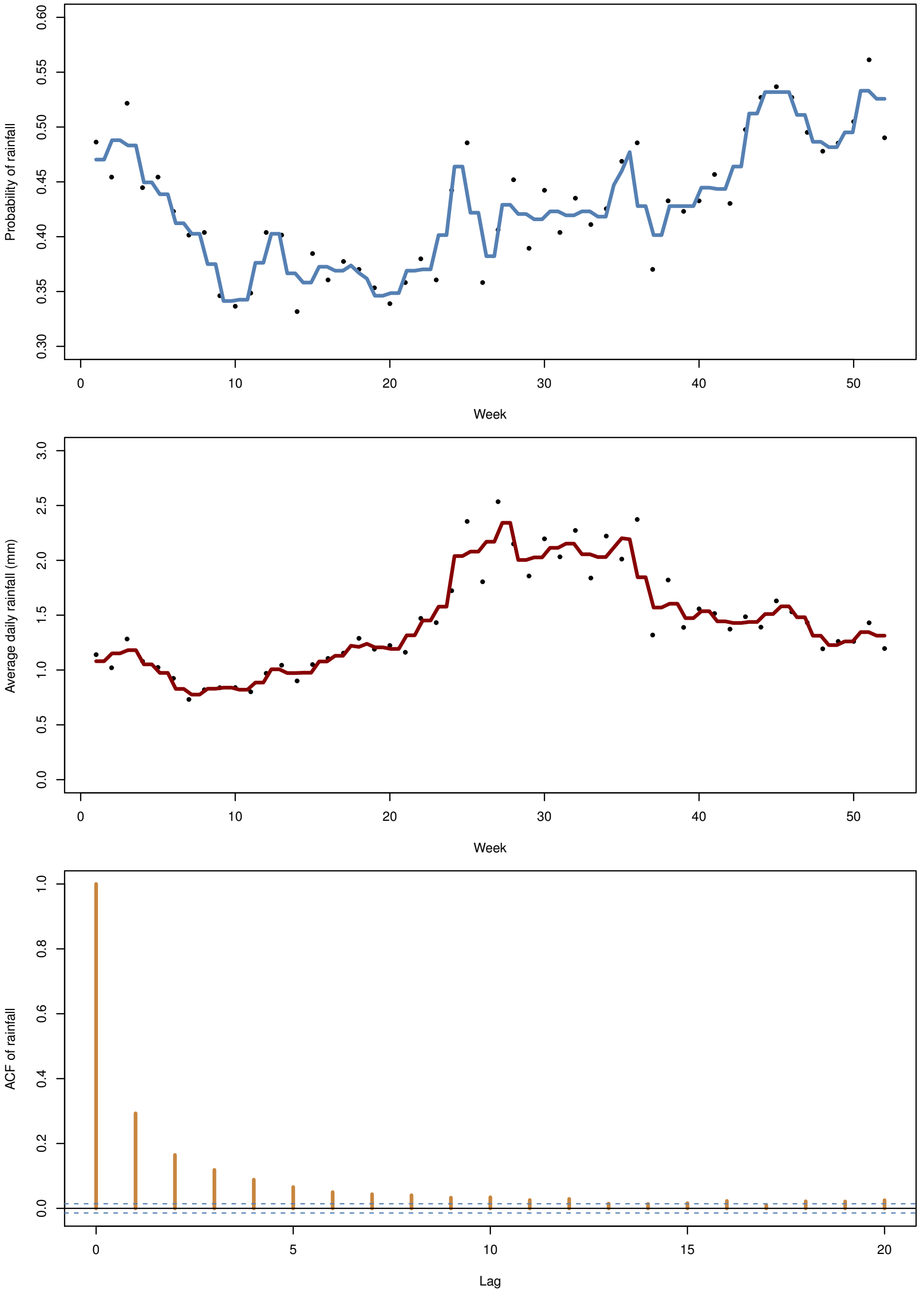

Figure 2.5: *The daily probability of rainfall (upper), the daily amount of rainfall (middle) and the ACF of rainfall (lower). The values are calculated as weekly averages of daily data from Malmslätt during the period between the years 1952 and 2002. The data is provided by the Swedish weather service (SMHI) and is used under the creative commons license.*

$\phi \in (-1, 1) \subset \mathbb{R}$ and $\sigma_v \in \mathbb{R}_+$. The inference problem in this model could be to determine the probability of rainfall (estimate p_t) given the data. Also, it could be interesting to determining the strength of the persistence in the system determined by ϕ or the strength of the seasonal components determined by $\{\alpha_1, \alpha_2, \beta_1, \beta_2\}$.

2.3 Maximum likelihood parameter inference

In this section, we present the fundamentals of ML based parameter inference in SSMs. This presentation is mainly included to set the notation and to highlight some features of the method that are needed in the sequel. An accessible general introduction to ML inference is given by Casella and Berger (2001). For more extensive treatments, see Rao (1965) and Lehmann and Casella (1998).

Inference in the ML paradigm is focused on optimising the likelihood function $\mathcal{L}(\theta) = p_\theta(y_{1:T})$, which also appears in Bayesian inference. The likelihood encodes the information contained with the observations $y_{1:T}$ into a quantity that can be used for inference.

2.1 Definition (Likelihood function for an SSM). The *likelihood (function)* of an SSM can be expressed as the decomposition

$$\mathcal{L}(\theta) = p_\theta(y_{1:T}) = p_\theta(y_1) \prod_{t=2}^{T} p_\theta(y_t | y_{1:t-1}), \tag{2.8}$$

where $p_\theta(y_t | y_{1:t-1})$ denotes the one-step-ahead predictor.

It is common to replace the likelihood function with the *log-likelihood (function)* in many inference problems. This is done to simplify analytical calculations and improve the numerical stability of many algorithms. The log-likelihood is given by

$$\ell(\theta) = \log p_\theta(y_{1:T}) = \log p_\theta(y_1) + \sum_{t=2}^{T} \log p_\theta(y_t | y_{1:t-1}). \tag{2.9}$$

In general, there are two different interpretations of the likelihood in statistics. The first is that the likelihood is a *function of the data* for a fixed parameter θ. A common name for this distribution is the *sampling distribution* and it plays an important role when calculating the distribution of some sampled data. The second interpretation (which we adopt in this thesis) is that the likelihood is a function of the parameter. That is the data is fixed and that the likelihood therefore summaries the information in the data.

The ML parameter inference problem is formulated as a maximisation problem of the likelihood or equivalently the log-likelihood. This follows from that the logarithm is a monotone function and hence any maximiser of the likelihood is also a maximiser of the log-likelihood.

2.2 Definition (Maximum likelihood parameter inference problem). The parameter inference problem in the ML setting is given by

$$\widehat{\theta}_{\mathrm{ML}} = \operatorname*{argmax}_{\theta \in \Theta} \mathcal{L}(\theta) = \operatorname*{argmax}_{\theta \in \Theta} \ell(\theta), \tag{2.10}$$

where $\widehat{\theta}_{\mathrm{ML}}$ denotes the ML parameter estimate.

The interpretation of (2.10) is that we should select the parameter that together with the model is the most likely to have been generated the observations. This definition makes good intuitive sense, but as we shall see in the following section it is not the only method for estimating the parameter given the data.

We continue by discussing some useful quantities connected with the log-likelihood and will then return to discuss the properties of the estimator in (2.10). The gradient of the log-likelihood is referred to as the *score function* and the negative Hessian of the log-likelihood is referred to as the *observed information matrix.* These quantities are useful in many optimisation algorithms as they are the first order and second order information about the objective function (the log-likelihood) in (2.10), respectively. Also, this information can be used to build efficient proposals in some sampling algorithms, see Paper A.

2.3 Definition (Score function). The *score function* is defined as the gradient of the log-likelihood,

$$\mathcal{S}(\theta') = \nabla \ell(\theta)\big|_{\theta=\theta'}, \tag{2.11}$$

where the gradient is taken with respect to the parameter vector.

The score function has a natural interpretation as the slope of the log-likelihood. Hence, the score function is zero when evaluated at the true parameter vector, $\mathcal{S}(\theta^\star) = 0$. However, note that this is not necessarily true when we work with finite data samples as is discussed in Example 2.6.

2.4 Definition (Observed information matrix). The *observed information matrix* is defined as the negative Hessian of the log-likelihood,

$$\mathcal{J}(\theta') = -\nabla^2 \ell(\theta)\big|_{\theta=\theta'} \tag{2.12}$$

where the Hessian is taken with respect to the parameter vector.

The statistical interpretation of the observed information matrix is as a measure of the amount of information in the data regarding the parameter θ. That is, if the data is informative the resulting information matrix is large (according to some measure). Also, the information matrix can geometrically be seen as the negative curvature of the log-likelihood. As such, we expect it to be positive definite (PD) at the ML parameter estimate (c.f. the second-derivative test in basic calculus). Finally, we note that there exists a limiting behaviour for the observed information matrix, which approaches the so called expected information matrix as the number of data points tends to infinity.

2.5 Definition (Expected information matrix). The *expected information matrix* (or the Fisher information matrix) is defined by the expected value of the observed information matrix (2.12),

$$\mathcal{I}(\theta') = -\mathbb{E}_{y_{1:T}}\Big[\nabla^2 \ell(\theta)\big|_{\theta=\theta'}\Big] = \mathbb{E}_{y_{1:T}}\bigg[\Big(\nabla \ell(\theta)\big|_{\theta=\theta'}\Big)^2\bigg], \qquad (2.13)$$

which is evaluated with respect to the data record.

Note, that the expected information matrix is independent of the data realisation, whereas the observed information is dependent on the realisation. The expected information matrix is PD for all values of θ as it can be seen as the variance of the score function. We make use of this property in Section 4.2.2 and in Paper A to construct a random walk on a Riemann manifold using the information matrices. We conclude the discussion on the score function and the information matrix by Example 2.6, where we investigate the defined quantities for an LGSS model as a function of the parameter ϕ.

2.6 Example: Score and information matrix in the LGSS model

Consider the LGSS model (2.2) with the parameter vector $\theta^\star = \{0.5, 0, 1, 0.1\}$ from which we generate a realisation of length $T = 250$ using the initial value $x_0 = 0$. We fix $\{\sigma_v, \sigma_e\}$ at their *true* values and create a grid over ϕ. For each grid point, we calculate the log-likelihood, the score function and the expected information matrix. The results are presented in Figure 2.6.

We note that the log-likelihood has a distinct maximum near the *true* parameter (presented as dotted vertical lines) and that the score function is zero close to this point. The small difference in the score function is due to that we use a finite amount of data. Finally, the zero of the score function results in a maximum of the log-likelihood function as the expected information (negative Hessian) is positive.

With the definition of the expected information matrix in place, we now return to discussing the properties of the ML parameter estimate. The ML estimator obtained from (2.10) has a number of strong asymptotic properties, i.e. when the number of observations tends to infinity. It is possible to show that this estimator is *consistent*, *asymptotically normal* and *efficient* under some regularity conditions. These conditions include that the parameter space is compact and that the likelihood, score function and information matrix exist and are well-behaved.

The (ML) estimator is said to be *consistent* as it fulfils that

$$\widehat{\theta}_{\text{ML}} \xrightarrow{a.s.} \theta^\star, \qquad T \to \infty.$$

That is, the estimate almost surely converges to the true value of the parameter in the limit of infinite data. Furthermore, as the estimator is asymptotically normal, we have that the error in the estimate satisfies a CLT given by

$$\sqrt{T}\left(\widehat{\theta}_{\text{ML}} - \theta^\star\right) \xrightarrow{d} \mathcal{N}\Big(0, \mathcal{I}^{-1}(\theta^\star)\Big),$$

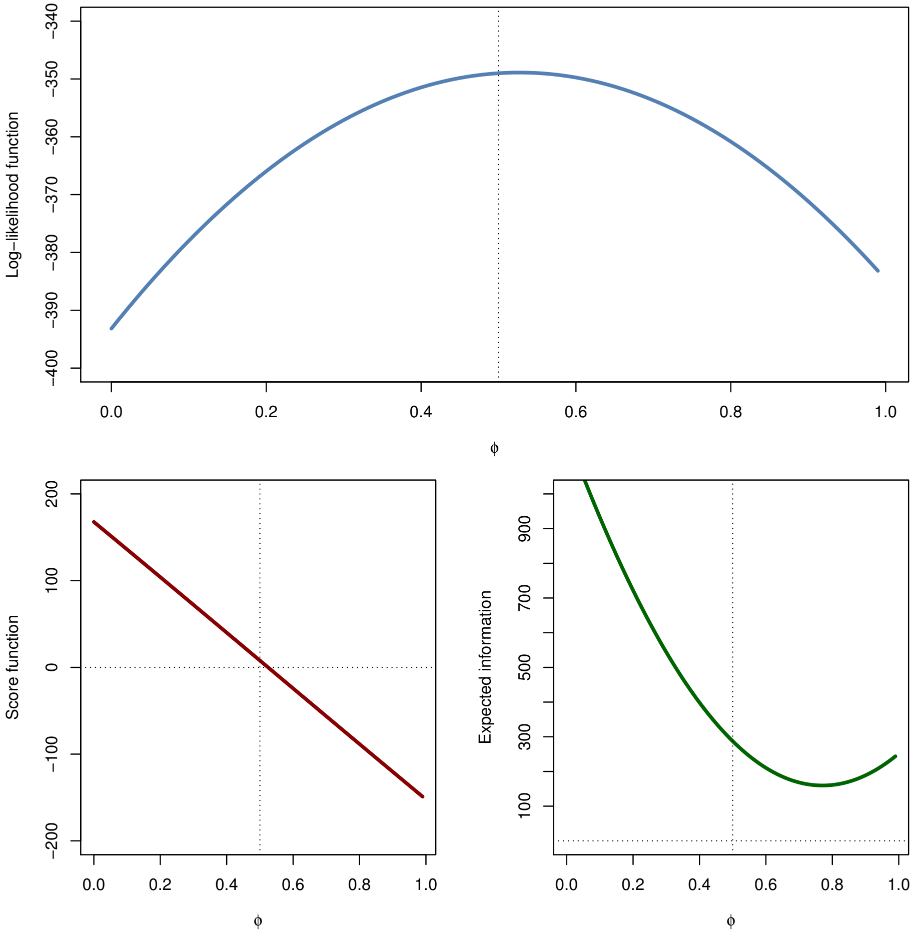

Figure 2.6: *The estimates of the log-likelihood function (upper) the score function (lower left) and the expected information matrix (lower right) of the LGSS model in Example 2.6. The dotted lines indicate the true parameter and the zero-level.*

which follows from a Taylor expansion around the point $\theta = \theta^{\star}$ of the log-likelihood. Note that, the expected information matrix enters into the expression and limits the accuracy of the estimate.

Therefore, a natural question is if we can somehow change the size of this matrix to decrease the lower-bound on the variance of the estimate. This is a key problem in the field of input design, where an input is added to the SSM to maximise some scalar function of the expected information matrix. This problem is discussed for nonlinear SSMs in Example 4.10 and Paper D.

Lastly, we say that an estimator is *efficient* if it attains the *Cramér-Rao lower bound*, which means that no other consistent estimator has a lower mean square error (MSE). That is, the ML estimator is the best unbiased estimator in the MSE sense and there are no better unbiased estimators. This last property is appealing and one might be tempted to say that the this estimator is the *best* choice for parameter inference. However, this result is only valid when we have an infinite number of samples. Therefore, other estimators (e.g. Bayes estimators discussed in the next section) could have better properties in the finite sample regime. Also, there can exist biased estimators with lower MSE than the ML estimator.

We have now introduced the ML parameter inference problem and some of the properties of the resulting estimate. In practice, we usually encounter a number of problems when we try to solve the optimisation problem in (2.10). These includes that for nonlinear SSMs: (i) the log-likelihood, the score and the information matrix are all intractable, (ii) the optimisation problem cannot be solved in closed-form and (iii) there could exist many local maxima of the log-likelihood making numerical optimisation problematic.

To solve these problems, a number of different numerical approaches have been developed. We return to the use of these approaches for estimating the log-likelihood, the score and the information matrix in Section 3. We also discuss other numerical methods for solving the ML parameter inference problem in Chapter 4.

2.4 Bayesian parameter inference

Previously in the ML paradigm, we implicitly assumed that the true parameter is a specific value. In this section, we instead assume that the true parameter is distributed according to some probability distribution. From this assumption, we can construct a different statistical paradigm called Bayesian inference, which can be to use for state and parameter inference in nonlinear SSMs.

As in the previous section, we only briefly discuss Bayesian inference for SSMs to set the notation and highlight some important features that we make use of in this thesis. For general treatments of Bayesian analysis, see Robert (2007) and Berger (1985). Furthermore, two accessible introductions are Gelman et al. (2013) and Casella and Berger (2001).

In the Bayesian parameter inference problem, we combine the information in the data described by the likelihood $p_\theta(y_{1:T})$ with prior information encoded as a probability distribution denoted $p(\theta)$. This combination of the prior information and the information in the data is made using Bayes' theorem,. This procedure is referred to as the *prior-posterior update* in Bayesian inference. The result is an updated probability distribution called the *posterior distribution*, which we denote $p(\theta|y_{1:T})$ in the parameter inference problem. Similar distributions can be defined for the state inference problem and we return to this is Section 3.1.

2.7 Definition (Bayesian parameter inference problem). Given the parameter prior $p(\theta)$ and the likelihood $p_\theta(y_{1:T})$, we obtain the *parameter posterior* by *Bayes' theorem* as

$$p(\theta|y_{1:T}) = \frac{p_\theta(y_{1:T})p(\theta)}{p(y_{1:T})} \propto p_\theta(y_{1:T})p(\theta), \tag{2.14}$$

where $p(y_{1:T})$ denotes the *marginal likelihood* (or the *evidence*). The name is due to that it can be computed by marginalisation

$$p(y_{1:T}) = \int p_\theta(y_{1:T})p(\theta)\,\mathrm{d}\theta. \tag{2.15}$$

One of the main questions in Bayesian inference is the choice of the prior and how we may encode our prior beliefs about the data into it. A common view is that this choice is *subjective* as it is often done using intuition and previous knowledge, which depends on subjective experiences. In this thesis, we make use of simple priors to encode stability properties of the nonlinear system or to keep the standard deviation positive at all times. For this, we make use of uniform priors over different subsets of the parameter space. An example of this is that we assume a uniform prior over $\phi \in (-1, 1) \subset \mathbb{R}$ for the LGSS model (2.2), which can be expressed as $p(\phi) = \mathcal{U}(\phi; -1, 1)$. Note, that the use of improper priors can be seen as a link between ML and Bayesian inference, but we shall not discuss this further. Instead, interested readers are referred to Robert (2007) for more information.

Another important class of priors that we make use of in Paper E is *conjugate priors*, which enables us to compute closed-form expressions for the posterior. A conjugate prior has the same functional form as the posterior and depends on the form of the likelihood. For example, the conjugate prior for the mean (given that the variance is known) of the Gaussian distribution is again a Gaussian distribution. This results from the fact that the product of two Gaussian distributions (the likelihood and the prior) is again a Gaussian distribution and hence it is a conjugate prior. We give another example of a conjugate prior in Example 2.8. Conjugate priors exist mainly for some combinations of priors and likelihoods in the exponential family of distributions, see Robert (2007) for a discussion.

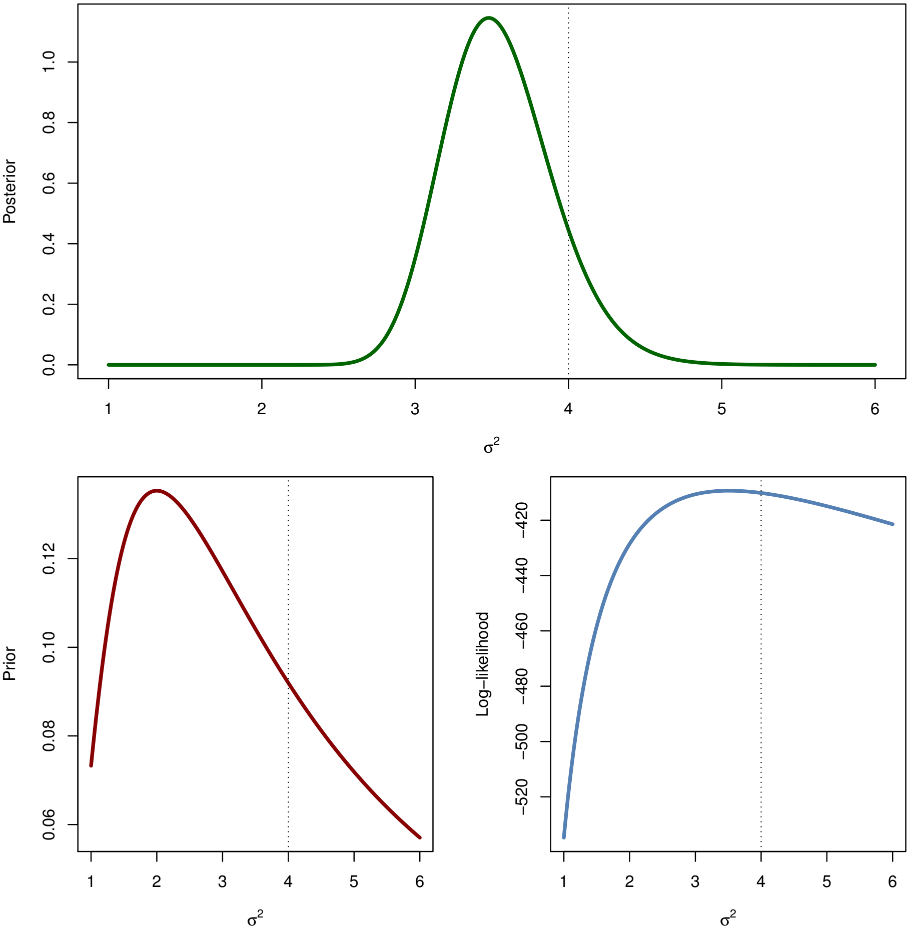

Figure 2.7: *The posterior (upper) of the parameter σ^2 in the Gaussian distribution resulting from the combination of an inverse-Gamma prior (lower left) and the data log-likelihood (lower right).*

2.8 Example: Prior-posterior update in a conjugate model

Consider the problem of inferring the value of the variance σ^2 in a Gaussian distribution given that we know the value of the mean μ. Also, we assume that we have obtained a set of IID data $y_{1:T}$ generated by the model. The conjugate prior to the variance in a Gaussian distribution is the inverse-Gamma distribution with rate α_0 and shape β_0 denoted $\mathcal{IG}(\alpha_0, \beta_0)$. By direct calculation, we obtain that the posterior has the parameters $\alpha_T = \alpha_0 + T/2$ and $\beta_T = \beta_0 + \frac{1}{2}\sum_{t=1}^{T}(y_i - \mu)^2$.

Here, we simulate the data realisation using the parameter $\{\mu, \sigma^2\} = \{1.5, 4\}$ of length $T = 200$ and use the prior coefficients $\{\alpha_0, \beta_0\} = \{1, 4\}$. The corresponding posterior, prior and log-likelihood are presented in Figure 2.7. We note that the prior assigns a small probability of the variance equal to 4. However, as the data record is relatively large, the resulting posterior is centred close to the true value of the parameter.

The Bayesian parameter inference problem is completely described by (2.14) and everything that we need is know is encoded in the posterior. This follows from the *likelihood principle*, which states that all information obtained from the data is contained within the likelihood function, see Robert (2007). However, we are sometimes interested in computing *point estimates* of the parameter vector. To compute point estimates, we can make use of *statistical decision theory* to make a decision about what information from the posterior to use. Consider a *loss function* $L : \Theta \times \Theta \to \mathbb{R}_+$, which takes the parameter and its estimate as input and returns a real-valued positive loss. The *expected posterior loss* (or posterior risk) $\rho\big(p(\theta), \delta \big| y_{1:T}\big)$ is given by

$$\rho\big(p(\theta), \delta \big| y_{1:T}\big) = \int L\big(\theta, \delta(y_{1:T})\big) p(\theta | y_{1:T}) \,\mathrm{d}\theta, \tag{2.16}$$

where $\delta(y_{1:T})$ denotes the decision of the parameter estimate given the data. The *Bayes estimator* is defined as the minimising argument of the expected posterior loss,

$$\delta^\star(y_{1:T}) = \operatorname*{argmin}_{\delta(y_{1:T}) \in \Theta} \rho\big(p(\theta), \delta \big| y_{1:T}\big).$$

Here, we restrict ourselves to discussing the resulting Bayes estimators for three different loss functions in Table 2.1, but there are many other possibly suitable choices for our application. From this, we see that the estimate that minimises the quadratic loss function is the expected value of the posterior,

$$\mathbb{E}[\theta] = \int \theta \, p(\theta | y_{1:T}) \,\mathrm{d}\theta. \tag{2.17}$$

That is, the Bayes estimator is given by the posterior mean for this choice of loss function. This is an example of a common expectation operation in Bayesian inference, as the expected value of the parameter is computed with respect to the parameter posterior distribution. Similar expressions can be written for many other Bayes estimators.

	Loss function	Bayes estimator
Linear	$L(\theta, \delta) = \lvert\theta - \delta\rvert$	Posterior median
Quadratic	$L(\theta, \delta) = (\theta - \delta)^2$	Posterior mean
0-1	$L(\theta, \delta) = \mathbb{I}(\theta = \delta)$	Posterior mode

Table 2.1: *Different loss functions and the resulting Bayes estimator.*

The statistical properties of the Bayes estimator depends on the choice of loss function when we work with finite data. However, it follows (under some conditions) from the *Bernstein–von Mises theorem* that the influence of the prior diminishes when the amount of data grows. Consequently, the MAP estimator (the posterior mode) converges to the ML estimator in some cases when the amount of data increases. Therefore it is possible to see ML parameter inference as a special case of Bayesian parameter inference in some sense. Finally, we note that Bayes estimators asymptotically have the same properties as the ML estimator, i.e. they are *consistent*, *asymptotically normal* and *efficient*. Also, as the number of data tends to infinity, the posterior tends to a Gaussian distribution which is known as the *Bayesian CLT*. For more information about the statistical properties of the Bayes estimators, see Robert (2007), Lehmann and Casella (1998) and Berger (1985).

We have now introduced the Bayesian paradigm for parameter and state inference in nonlinear SSMs. We have seen that the key quantity in Bayesian inference is the posterior distribution, which depends on the likelihood. Therefore, we in practice encounter the same problems with analytically intractable likelihoods as for the ML paradigm. Another complication is that the posterior might not be described by any known distribution. As a consequence, we cannot compute many of the integrals depending on the posterior distribution encountered in Bayesian analysis in closed-form. Examples of such integrals are the normalisation in (2.15), the marginalisation in (2.16) and the expectation in (2.17).

To counter these problem, we can make use of the same numerical methods from Chapter 3 as for the ML paradigm to estimate the likelihood function. To solve the Bayesian parameter inference problem, we can make use of sampling methods to obtain approximations of the posterior distributions of interest. These methods use computer simulations and have therefore only been available for practical use during the last decades. However, during this period these methods have been well-studied and are today established tools for Bayesian inference. Another approach is to make analytical approximations of the posterior distribution using known distributions or iterative procedures. We discuss some of these methods in Chapter 4

3

State inference using particle methods

In this chapter, we discuss the state inference problem in detail and presents some algorithms for approximate state inference. The foundation of state inference in SSMs is given by the *Bayesian filtering and smoothing recursions* that are discussed in Section 3.1. Unfortunately, these recursions are analytically intractable for most SSMs. Therefore, the remaining part of this chapter is devoted to discussing numerical methods based on SMC for approximate state inference. We also discuss how to make use of SMC methods to estimate the likelihood, the score function and the information matrix in SSMs. This chapter is concluded by discussing the direct illumination problem in computer graphics and how it can be solved using SMC methods.

3.1 Filtering and smoothing recursions

There are mainly two types of state inference problems in an SSM: *filtering* and *smoothing*. Both the filtering and smoothing problem can be *marginal* (inference on a single state), *k-interval* (inference of k states) or *joint* (inference on all states). In marginal filtering, only observations $y_{1:t}$ collected until the current time step t are used to infer the current value of the state x_t. In marginal smoothing, all the collected observations including (possibly) future observations $y_{1:T}$ are used to infer the value of the current state x_t with $t \leq T$. In Table 3.1, we summarise some common filtering and smoothing problems in SSMs.

The solutions to the filtering and smoothing problems are given by the Bayesian filtering and smoothing recursions (Jazwinski, 1970; Anderson and Moore, 2005). These (as the names suggest) are based on Bayesian inference similar to the parameter inference problems discussed in Section 2.4. These recursions can be used

Name	Density
(Marginal) filtering	$p_\theta(x_t \vert y_{1:t})$
(Marginal) smoothing ($t \leq T$)	$p_\theta(x_t \vert y_{1:T})$
Joint smoothing	$p_\theta(x_{0:T} \vert y_{1:T})$
Fixed-interval smoothing ($s < t \leq T$)	$p_\theta(x_{s:t} \vert y_{1:T})$
Fixed-lag smoothing (for lag Δ)	$p_\theta(x_{t-\Delta-1:t} \vert y_{1:t})$

Table 3.1: *Common filtering and smoothing densities in SSMs.*

to iteratively solve the filtering problem for each time t using two steps given by

$$p_\theta(x_t|y_{1:t}) = \frac{p_\theta(x_t|y_{1:t-1}) g_\theta(y_t|x_t)}{p_\theta(y_t|y_{1:t-1})}, \tag{3.1a}$$

$$p_\theta(x_t|y_{1:t-1}) = \int f_\theta(x_t|x_{t-1}) p_\theta(x_{t-1}|y_{1:t-1}) \, \mathrm{d}x_{t-1}. \tag{3.1b}$$

In the first step, the state estimate at time t is updated with the new observation y_t (also known as the *measurement update*) using Bayes' theorem. In the second step, a marginalisation is used to predict the next state x_t using the information in the current state and the collected observations (also known as the *time update*). In a similar manner, the smoothing problem can be solved by iterating the marginalisation

$$p(x_t|y_{1:T}) = p(x_t|y_{1:t}) \int \frac{f(x_{t+1}|x_t) p(x_{t+1}|y_{1:T})}{p(x_{t+1}|y_{1:t})} \, \mathrm{d}x_{t+1}, \tag{3.2a}$$

$$p(x_{t+1}|y_{1:t}) = \int f(x_{t+1}|x_t) p(x_t|y_{1:t}) \, \mathrm{d}x_t, \tag{3.2b}$$

backwards in time. Here the smoothing recursion, makes use of the filtering distribution computed in a forward pass. The resulting smoother from this procedure is therefore known as the *forward filtering backward smoothing* (FFBSm) algorithm. We return to a numerical method based on the FFBSm algorithm for the use in SSMs later in this chapter.

The filtering (3.1) and smoothing (3.2) recursions can only be solved analytically for two different classes of SSMs: linear Gaussian SSMs and SSMs with finite state processes. For the former, the recursions can be solved by the *Kalman filter* (KF) and e.g. the *Rauch–Tung–Striebel* (RTS) smoother (Rauch et al., 1965), respectively. Using the Kalman methods, we can exactly compute the likelihood and the states in an LGSS model. We do not discuss the details of the Kalman methods here and refer readers to Anderson and Moore (2005) and Kailath et al. (2000) for extensive treatments of Kalman filtering and different Kalman smoothers.

The general intractability of the Bayesian filtering and smoothing recursions results from that there are no closed-form expressions for the densities in the recursions. This is similar to the problems that we encountered in the Bayesian parameter inference problem. Instead, we consider numerical approximations that relies on

Monte Carlo (MC) methods to estimate the filtering and smoothing distributions. This results in SMC and related methods, which we return to in Section 3.3.

3.2 Monte Carlo and importance sampling

MC methods are a collection of statistical simulation methods based on sampling and the *strong law of large numbers* (SLLN). For a general introduction to MC methods, see Robert and Casella (2004) for an extensive treatment or Ross (2012) for an accessible introduction. In this thesis, we mainly use MC methods for estimating the expected value (an integral) of an arbitrary well-behaved *test function* $\varphi(x)$,

$$\widehat{\varphi} = \mathbb{E}_\pi\big[\varphi(x)\big] = \int \varphi(x)\pi(x)\,\mathrm{d}x, \tag{3.3}$$

where $\pi(x)$ denotes a (normalised) *target distribution* from which we can simulate IID *particles* (or samples). As a consequence of the SSLN[1], we can estimate the expected value by the sample average

$$\widehat{\varphi}_{\mathrm{MC}} = \frac{1}{N}\sum_{i=1}^{N} \varphi\big(x^{(i)}\big), \qquad x^{(i)} \sim \pi(x),$$

taken over independent realisations $x^{(i)}$ simulated from $\pi(x)$. This estimator is *strongly consistent*, i.e.

$$\widehat{\varphi}_{\mathrm{MC}} \xrightarrow{a.s.} \widehat{\varphi}, \qquad N \to \infty.$$

Also, we can construct a CLT for the error of the MC estimator, given by

$$\sqrt{N}\Big(\widehat{\varphi} - \widehat{\varphi}_{\mathrm{MC}}\Big) \xrightarrow{d} \mathcal{N}\Big(0, \sigma^2_{\mathrm{MC}}\Big), \qquad \sigma^2_{\mathrm{MC}} = \mathbb{V}[\varphi(x)] < \infty,$$

where we assume that the function $\varphi(x)$ has a finite second moment. Here, we see that the MC estimator is asymptotically unbiased with Gaussian errors. Also, the variance of the error decreases as $1/N$ independent of the dimension of the problem, which is one of the main advantages of the MC methods.

Consider, the problem of estimating (3.3) when the normalised target $\pi(x) = \gamma(x)Z^{-1}$ is unknown together with the normalisation factor Z. Instead, we can only evaluate the unnormalised target $\gamma(x)$ point-wise. In this case, *importance sampling* (IS) (Marshall, 1956) can be used to sample from another distribution called the *proposal distribution* (or importance distribution) $q(x)$ and adapt the particles using a weighting scheme. For this, it is required that $q(x) \neq 0$ for almost all $x \in \mathsf{supp}(\pi)$ and that the support of $\varphi(x)q(x)$ contains the support of $\pi(x)$, i.e. $\mathsf{supp}(\pi) \subset \mathsf{supp}(\varphi q)$. The IS algorithm follows by rewriting the expected value in

[1] The SSLN states that the sample average $\bar{x}$ computed using N IID samples from $\pi(x)$, converges almost surely to $\mu = \mathbb{E}_\pi[x]$ when N tends to ∞, i.e. $\bar{x} \xrightarrow{a.s.} \mu$.

Algorithm 1 Importance sampling (IS)

INPUTS: $\gamma(x)$ (unnormalised target), $q(x)$ (proposal) and $N > 0$ (no. particles).
OUTPUT: $\widehat{p}(\mathrm{d}x)$ (empirical distribution).

```
for $i = 1$ to $N$ do
    Sample a particle by $x^{(i)} \sim q(x)$.
    Compute the weight by $w^{(i)} = \gamma(x^{(i)})/q(x^{(i})$.
end for
Normalise the particle weights by (3.5).
Estimate $\widehat{p}(\mathrm{d}x)$ using (3.7).
```

(3.3) to obtain

$$\mathbb{E}_\pi\big[\varphi(x)\big] = \int \varphi(x)\pi(x)\,\mathrm{d}x = \int \varphi(x)\,\underbrace{\frac{\gamma(x)}{q(x)}}_{\triangleq w(x)}\,q(x)\,\mathrm{d}x = \mathbb{E}_q\big[w(x)\varphi(x)\big],$$

where $w(x)$ denotes the (unnormalised) *importance weights.* The IS estimator follows in analogue with the vanilla MC estimator,

$$\widehat{\varphi}_{\mathrm{IS}} = \sum_{i=1}^{N} \widetilde{w}^{(i)}\varphi\Big(x^{(i)}\Big), \qquad x^{(i)} \sim q(x), \tag{3.4}$$

where the *normalised weights* are given by

$$\widetilde{w}\big(x^{(i)}\big) \triangleq \widetilde{w}^{(i)} = \frac{w^{(i)}}{\sum_{k=1}^{N} w^{(k)}}, \qquad i = 1, \ldots, N. \tag{3.5}$$

We can also construct an unbiased estimator for the normalisation constant by

$$\widehat{Z}_{\mathrm{IS}} = \frac{1}{N}\sum_{i=1}^{N} w^{(i)}, \tag{3.6}$$

which we later shall make use of to estimate the (marginal) likelihood in SSMs. The IS algorithm can be interpreted as generating an *empirical distribution*, which can be seen by rewriting the estimator in (3.4) as a Dirac mixture given by

$$\widehat{p}(\mathrm{d}x) = \sum_{i=1}^{N} \widetilde{w}^{(i)}\delta_{x^{(i)}}(\mathrm{d}x), \tag{3.7}$$

which we can insert into (3.3) to recover (3.4) as presented in Algorithm 1. Here, $\delta_z(\mathrm{d}x)$ denotes a Dirac point mass located at $x = z$. In the following, we make use of this property to compute expectations in state inference problems using sequential variants of the IS algorithm.

We conclude the discussion of the IS algorithm by Example 3.1 in which we estimate the parameters of the HWSV model using some real-world data. Note that, the IS algorithm can be developed further to include multiple and adaptive proposals. In *multiple importance sampling* (MIS), the algorithm can make use of several proposal algorithms that are then combined to form the estimate. In *adaptive importance sampling* (AIS), we update a mixture of proposals or the proposal after each iteration to fit the target better. These methods could be interesting in developing new approximate state inference algorithms similar to the particle methods discussed in the next section. For more information, see Kronander and Schön (2014), Cornuet et al. (2011) and Veach and Guibas (1995).

3.1 Example: IS for Bayesian parameter inference in the HWSV model

Consider the Bayesian parameter inference problem in the Hull-White SV model (2.4) and the data presented in Section 2.2.2. In this model, we would like to estimate the parameters given the data by the use of the IS algorithm. The unnormalised target distribution is given by

$$\gamma(\theta) = p_\theta(y_{1:T})p(\theta),$$

where we assume that we can evaluate the likelihood point-wise. Note that, the solution to this problem is analytically intractable but can approximated using methods discussed later in Section 3.3.4. Furthermore, we assume the uniform priors over $\phi \in (-1,1) \subset \mathbb{R}$ and $\{\sigma_v, \beta\} \in \mathbb{R}^2_+$. The parameter proposals are given by

$$\begin{aligned} q(\phi,) &= \mathcal{U}(\phi; -1, 1), \\ q(\sigma_v) &= \mathcal{G}(\sigma_v; 2, 0.1), \\ q(\beta) &= \mathcal{G}(\beta; 7, 0.1). \end{aligned}$$

These proposals are based on prior information regarding the typical values for parameters in the model. Hence, they can be seen as a kind of prior distributions in the Bayesian inference. For the IS algorithm, we use $N = 200$ samples and the procedure outlined in Algorithm 1.

In Figure 3.1, we present the resulting posterior distributions with the proposal distributions. The parameter estimates obtain by the posterior mode are $\widehat{\theta}_{\text{IS}} = \{0.996, 0.129, 0.837\}$, which indicates a slowly varying latent volatility process. In Chapter 4, we discuss other methods for parameter inference that are more efficient when the number of parameters increases. However, for small problems the IS algorithm could be a useful alternative in cases when the prior information is good. Otherwise, the number samples required from the parameter posterior increases exponentially with the number of elements in the parameter vector.

3.3 Particle filtering

The IS algorithm discussed in the previous section can be combined with the filtering recursions in Section 3.1 to sequentially estimate the joint smoothing distribution $\pi(x_{0:t}) = p_\theta(x_{0:t}|y_{1:t})$. To this end, we can apply Algorithm 1 to sequentially approximate the target using an empirical distribution. The resulting algorithm is known as the *sequential importance sampling* (SIS) algorithm. The

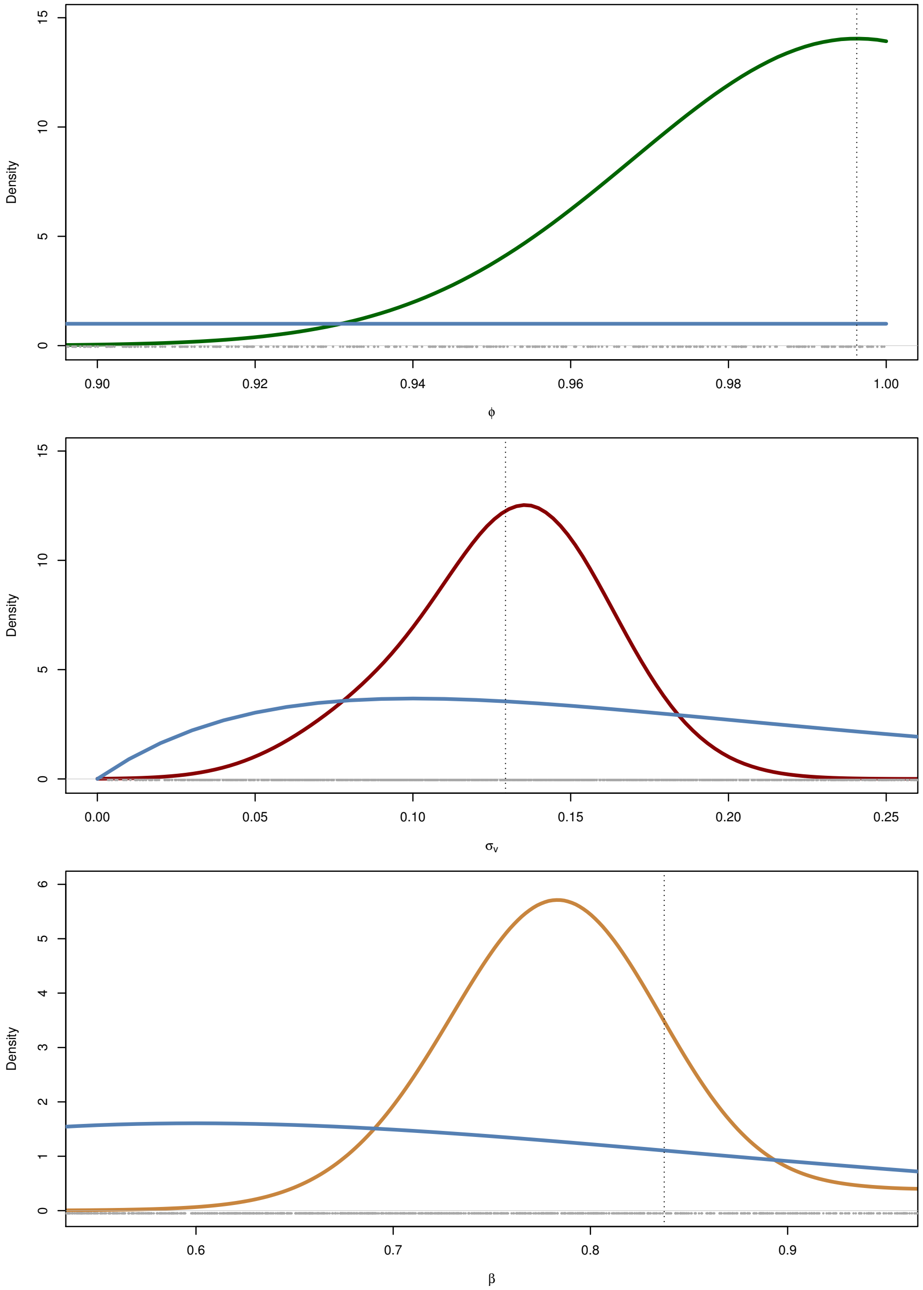

Figure 3.1: *The estimates posterior distributions for ϕ (green), σ_v (red) and β (orange) compute as KDEs with the proposal distributions presented in blue. The grey dots indicates the samples obtained from the proposal and the dotted lines indicate the posterior means.*

main problem with this algorithm is that the variance of the estimate increases rapidly as t increases. For a long time, this was a major obstacle in approximate state inference. This results from that the particle weights deteriorate over time and in the limit only a single particle has a non-zero weight after a few iterations. Hence, the effective number of particles is one, which is the reason for the high variance in the estimates.

However, by including a resampling step into the SIS algorithm, we can focus the attention of the algorithm on the areas of interest. The resampling step essentially duplicates particles with large weights and discard particles with small weights, while keeping the total number of particles fixed. Hence, we do not end up with only one effective particle in the estimator. This development of the algorithm leads to the SIS with resampling (SIR) algorithm. When this method is applied on SSMs, we obtain the basic particle filtering algorithm introduced by Gordon et al. (1993). In this thesis, we refer to this algorithm as the *bootstrap particle filter* (bPF).

In subsequent developments, the particle filter is generalised to include more advanced proposals and resampling schemes. In this section, we present a refined version of the bPF called the *auxiliary particle filter* (APF) (Pitt and Shephard, 1999), which can use more general particle proposals and weighting functions than in the original formulation. This can result in a large decrease in the variance of the estimate and also a decrease in the number of particles required to achieve a certain accuracy. The APF also allows for the use of different resampling schemes than the multinomial resampling that is used in the original formulation of the bPF.

To keep the presentation brief, we do not derive the APF and refer interested readers to Doucet and Johansen (2011) for a derivation of the APF starting from the IS algorithm. Furthermore, we note that the APF is a member of the more general family of SMC algorithms, which are discussed in Cappé et al. (2007) and Del Moral et al. (2006).

3.3.1 The auxiliary particle filter

The APF algorithm operates by constructing a particle system $\{w_t^{(i)}, x_t^{(i)}\}_{i=1}^N$ sequentially over time. By the use of this particle system, we can construct an *empirical marginal filtering distribution* in analogue with (3.7) as

$$\widehat{p}_\theta(\mathrm{d}x_t | y_{1:t}) \triangleq \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta_{x_t^{(i)}}(\mathrm{d}x_t), \tag{3.8a}$$

$$\widetilde{w}_t^{(i)} \triangleq \frac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}}, \tag{3.8b}$$

which can be seen as a discrete approximation of the filtering distribution using a collection of weighted Dirac point masses. Here $x_t^{(i)}$ and $w_t^{(i)}$ denote particle i at time t and its corresponding (unnormalised) importance weight. Also, the *empirical joint smoothing distribution* follows similarly as

$$\widehat{p}_\theta(\mathrm{d}x_{0:t}|y_{1:t}) \triangleq \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta_{x_{0:t}^{(i)}}(\mathrm{d}x_{0:t}). \tag{3.9}$$

Three steps are carried out during each iteration of the APF to update the particle system from time $t-1$ to t:

(i) The particles are *resampled* according to their auxiliary weights. The means that particles with small weights are discarded and particles with large weights are multiplied. This mitigates the weight depletion problem experienced by the SIS algorithm. This is often the computational bottleneck of the APF algorithm as all the other steps can easily be implemented in parallel.

(ii) The particles are *propagated* from time $t-1$ to t. This can be seen as a simulation step, where new particles are generated by sampling from a Markov kernel.

(iii) The (unnormalised) particle weight is calculated for each particle. These weights compare the particle with the recorded output from the system. A high weight is given to a particle that is likely to generate y_t.

After these three steps, we can construct empirical distributions using (3.8) and (3.9). It is also possible to make use of the particle system for estimating other quantities. We return to this in the following. Note that, by comparison with the filtering recursions, Step (ii) corresponds to the time update in (3.1b) and Step (iii) corresponds to the measurement update in (3.1a). We now proceed to discuss each step of the APF in more detail.

Step (i) can be seen as a sampling of *ancestor indices* denoted $a_t^{(i)}$, i.e. the index of the particle at time $t-1$ from which the particle i at time t originates from. This can be expressed as simulating from a multinomial distribution with probabilities

$$\mathbb{P}(a_t^{(i)} = j) = \widetilde{\nu}_{t-1}^{(j)}, \qquad j = 1, \ldots, N, \quad i = 1, \ldots, N, \tag{3.10}$$

where $\widetilde{\nu}_t^{(j)}$ denotes a *normalised auxiliary weight* given by

$$\widetilde{\nu}_{t-1}^{(j)} = \nu_{t-1}^{(j)} \left[\sum_{k=1}^{N} \nu_{t-1}^{(k)} \right]^{-1},$$

for some auxiliary weight function $\nu_{t-1}^{(k)}$ determined by the user. After the resampling step, we obtain the unweighted particle system $\{\widetilde{x}_{t-1}^{(i)}, 1/N\}_{i=1}^{N}$.

In this thesis, we mainly consider two different types of APFs. The bPF is obtained by selecting the auxiliary weight as the particle weights, i.e. $\nu_t = w_t$. The *fully-adapted particle filter* (faPF) (Pitt and Shephard, 1999), which takes into account the (future) observation in the resampling step. This information is included by using the auxiliary weight given by

$$\nu_{t-1} = p_\theta(y_t|x_{t-1}) = \int g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}) \, \mathrm{d}x_t, \tag{3.11}$$

when it can be computed in closed-form. This is only possible for some systems, e.g. the LGSS model (2.2) and the GARCH(1,1) model (2.3).

In (3.10), we make use of multinomial resampling but there are other resampling algorithms that can be useful. *Systematic resampling* and *stratified resampling* are good alternatives to the multinomial resampling. These resampling methods often decrease the variance in the estimates of the filtering distribution. See Douc and Cappé (2005) and Hol et al. (2006) for comparisons of different resampling strategies.

In Step (ii), each particle is propagated using a propagation kernel as,

$$x_t^{(i)} \sim R_\theta\Big(x_t | x_{0:t-1}^{a_t^{(i)}}, y_t\Big), \qquad i = 1, \ldots, N. \tag{3.12}$$

The bPF is recovered by selecting the state dynamics as the propagation kernel,

$$R_\theta(x_t|x_{0:t-1}, y_t) = f_\theta(x_t|x_{t-1}).$$

The faPF makes use of the current observation when proposing new particles. This results in the propagation kernel

$$R_\theta(x_t|x_{0:t-1}, y_t) = p_\theta(x_t|y_t, x_{t-1}) = \frac{p_\theta(y_t|x_t) p(x_t|x_{t-1})}{p(y_t|x_{t-1})}, \tag{3.13}$$

when it can be computed in closed-form. Each new particle is appended to the particle trajectory by $x_{0:t}^{(i)} = \{x_{0:t-1}^{a_t^{(i)}}, x_t^{(i)}\}$, for $i = 1, \ldots, N$.

Finally, in Step (iii) each particle is assigned an importance weight by a weighting function $W_\theta(x_t, x_{t-1})$, similar to the IS algorithm in (3.4). The resulting weight is given by

$$w_t^{(i)} = W_\theta\Big(x_t^{(i)}, x_{0:t-1}^{a_t^{(i)}}\Big), \qquad i = 1, \ldots, N \tag{3.14a}$$

$$W_\theta\big(x_t, x_{0:t-1}\big) \triangleq \frac{w_{t-1}}{\nu_{t-1}} \frac{g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1})}{R_\theta(x_t|x_{0:1:t-1}, y_t)}. \tag{3.14b}$$

For the bPF, the choice of resampling step and propagation kernel leads to that the weights are given by

$$\nu_t = w_t = g_\theta(y_t|x_t),$$

i.e. the probability of observing y_t given x_t. For the faPF, we have that the particle weights are given by $w_t \equiv 1$.

The general form of the APF is presented in Algorithm 2 for approximate state inference in nonlinear SSMs. The complexity of the algorithm is linear in the number of time steps and particles, i.e. $\mathcal{O}(NT)$. The user choices are mainly the number of particles N, the auxiliary weights ν, the propagation kernel $R_\theta(x_t|x_{0:t-1}, y_t)$ and

Algorithm 2 Auxiliary particle filter (APF)

INPUTS: $y_{1:T}$ (observations), $R_\theta(x_t|x_{0:t-1}, y_t)$ (propagation kernel), ν_t (auxiliary weights) and $N > 0$ (no. particles).
OUTPUTS: $\widehat{p}_\theta(x_t|y_{1:t})$ for $t = 1$ to T (the empirical marginal filtering distributions) and $\widehat{p}_\theta(x_{0:t}|y_{1:t})$ for $t = 1$ to T (the empirical joint smoothing distributions).

```
Initialise each particle $x_0^{(i)}$.
for $t = 1$ to $T$ do
    Sample new the ancestor indices to obtain $\{a_t^{(i)}\}_{i=1}^N$ using (3.10).
    Propagate the particles to obtain $\{x_t^{(i)}\}_{i=1}^N$ using (3.12).
    Calculate new importance weights to obtain $\{w_t^{(i)}\}_{i=1}^N$ using (3.14).
    Estimate the marginal filtering distribution $\widehat{p}_\theta(x_t|y_{1:t})$ using (3.8).
    Estimate the joint smoothing distribution $\widehat{p}_\theta(x_{0:t}|y_{1:t})$ using (3.9).
end for
```

the resampling method. We now proceed with discussing the use of the APF for state inference and log-likelihood estimation.

3.3.2 State inference using the auxiliary particle filter

Algorithm 2 can be used to estimate the marginal filtering and joint smoothing distributions in an SSM given some observations. From these empirical distributions, we can compute an estimate of the filtered marginal state $x_{t|t}$,

$$\widehat{x}_{t|t} = \int x_t \, \widehat{p}_\theta(x_t|y_{1:t}) \, \mathrm{d}x_t = \sum_{i=1}^{N} \widetilde{w}_t^{(i)} x_t^{(i)}, \tag{3.15}$$

where we have inserted (3.8). Note that, the state trajectory $x_{0:t|t}$ can be estimated using an analogue expression given by

$$\widehat{x}_{0:t|t} = \int x_{0:t} \, \widehat{p}_\theta(x_{0:t}|y_{1:t}) \, \mathrm{d}x_{0:t} = \sum_{i=1}^{N} \widetilde{w}_t^{(i)} x_{0:t}^{(i)}, \tag{3.16}$$

by the use of the empirical smoothing distribution (3.9). In Example 3.2, we illustrate the use of the bPF for marginal state inference in the earthquake model (2.6) and data discussed in Section 2.2.3,

3.2 Example: State inference in the earthquake count model

Consider the earthquake count model (2.6) using the data presented in Figure 2.4. To infer the state in the model, we make use of the bPF with $N = 1\,000$ particles and the parameter vector $\widehat{\theta} = \{0.88, 0.15, 17.65\}$ estimated later in Example 4.9.

In the upper part of Figure 3.2, we present the filtered number of major earthquakes obtained from the bPF (red line) together with the data (black dots). We also present the predicted number of major earthquakes (dark green) for 2014 to 2022 together with a 95% CI for the predictions. The uncertainty grows fast for the predictions (gray area), which indicates that a more advanced model is probably required to make good predictions.

In the lower part of Figure 3.2, we present the estimated state obtained from the bPF, which indicates that the world currently is in a calmer period with a smaller earthquake intensity compared with e.g. the 1950s. This conclusion is also supported in the actual number of earthquakes recorded during these two periods. We also present the predicted latent state (dark green) for 2014 to 2022 together with a 95% CI for the predictions (gray area). Again, we see that there is a large uncertainty in the future latent state.

From Algorithm 2, we know that the APF algorithm can be used to approximate the joint smoothing distribution. In the following, we require estimates of this distribution for the use in some of the proposed methods for approximate parameter inference. However, the accuracy of these estimates are poor due to problems with *path degeneracy*. To explain the nature of this problem, consider the resampling step in the APF algorithm. During each resampling, we discard some particle with a non-zero probability due to that they have small auxiliary weights. This means that the number of unique particles is smaller than N with a non-zero probability after each resampling step.

As t increases, we repetitively resample the particle system and this leads to that the number of unique particles tends to one before some time $s < t$. That is, the particle trajectory collapses into a single trajectory for all the particles before time s. Consequently, the variance of the estimates of the joint smoothing distribution is large due to the same problem as in the SIS algorithm. To mitigate this effect, we could make use of the faPF or increase the number of particles. However, this could be problematic as the faPF is not available for many interesting models. Also the computational cost of the APF algorithm increases linearly with N. We illustrate this effect and compare these two alternatives in Example 3.3.

There exists additional alternatives to mitigate the path degeneracy to obtain accurate approximation of the joint smoothing distribution. A popular alternative in practice is to resample only when the variance in the particle weights is larger than some threshold (Doucet and Johansen, 2011). This leads to a decrease in the path degeneracy problem but this is often not enough to completely solve the problem. Instead, a particle smoother is often used for this problem, which gives better accuracy in the estimates but often increases the computational cost. In Section 3.4, we return to the use of particle smoothing for estimating the smoothing distributions. We now continue with discussing the statistical properties of the APF and how to use the APF to estimate the likelihood and log-likelihood for an SSM.

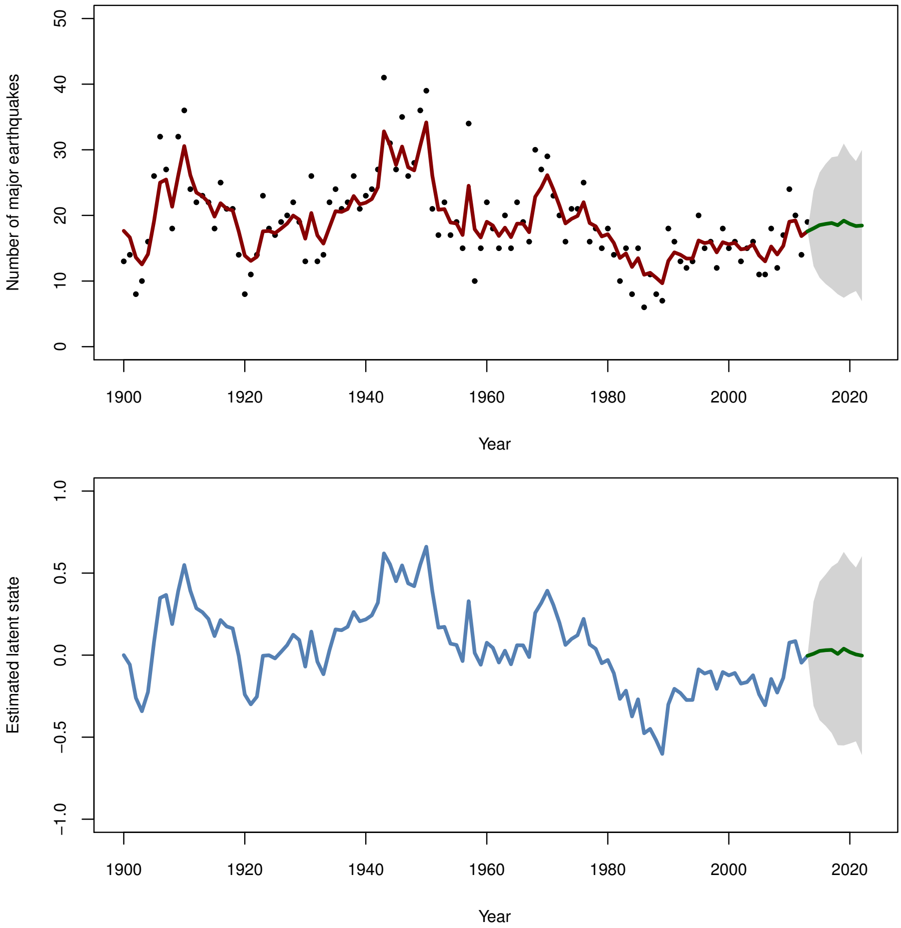

Figure 3.2: *Upper: the filtered number of major earthquakes (red line) and the data (black dots) obtained from the bPF using the data and the model from Example 3.2. The predicted number of earthquakes (green line) is also presented together with a 95% CI (gray area). Lower: the estimated latent earthquake intensity obtained from the bPF. The predicted latent state (green line) is also presented together with a 95% CI (gray area).*

3.3 Example: Path degeneracy in the GARCH(1,1) model

Consider the GARCH(1,1) model in (2.3) from which we generate $T = 20$ observations using the parameter vector $\theta^\star = \{0.10, 0.80, 0.05, 0.30\}$. Here, we make use of the bPF and faPF with systematic resampling at every iteration. The aim is to estimate the state trajectory $x_{0:t|t}$ using the APF and (3.16).

For this model, the weight function (3.11) and the proposal (3.13) required for the faPF can be computed in closed from by properties of the joint Gaussian distributions, see Bishop (2006). The required quantities are given by

$$\begin{aligned}
p(y_t|x_{t-1}, h_t) &= \int \mathcal{N}\left(y_t; x_t, \tau^2\right) \mathcal{N}\left(x_t; 0, h_t\right) \mathrm{d}x_t \\
&= \mathcal{N}\left(y_t; 0, \tau^2 + h_t\right), \\
p(x_t|y_t, x_{t-1}) &\propto p(y_t|x_t) p(x_t|x_{t-1}) \\
&= \mathcal{N}\big(y_t; x_t, \tau^2\big) \mathcal{N}\big(x_t; 0, h_t\big) \\
&= \mathcal{N}\Big(x_t; \left[\tau^{-2} + h_t^{-1}\right]^{-1} \tau^{-2} y_t, \tau^{-2} + h_t^{-1}\Big),
\end{aligned}$$

from the marginalisation property and the conditioning property.

In Figure 3.3, we present the state trajectories obtained by tracing the ancestor linage backwards from time T to 0. We see that for the bPF with $N = 10$ particles, the ancestral lines collapse to a single unique particle before time $t = 10$ due to the path degeneracy problem. Increasing the number of particles to $N = 20$ results in that the path degenerate before $t = 8$ instead. However, the faPF does not have the same problem with degeneracy as many ancestors survive the repeated resamplings and contributes with information for estimating the joint smoothing distribution.

3.3.3 Statistical properties of the auxiliary particle filter

In this section, we review some results regarding two aspects of the statistical properties of the APF. Note that, the analysis of the APF is rather complicated compared with other estimators that makes use of independent samples from the target distribution. Instead, the particles obtained from the APF are not independent due to the interaction during the resampling step. However, there are many strong results regarding the statistical properties of the bPF in the literature. Extensive technical accounts are found in Del Moral (2013) and Del Moral (2004). Some of the statistical properties are also discussed in a more application oriented setting in Crisan and Doucet (2002), Douc et al. (2014) and Doucet and Johansen (2011).

Assume that we would like to compute the expected value of some well-behaved test function $\varphi(x)$ using the particle system generated by the APF. This is in analogue with the IS estimator in (3.3). From the empirical filtering distribution

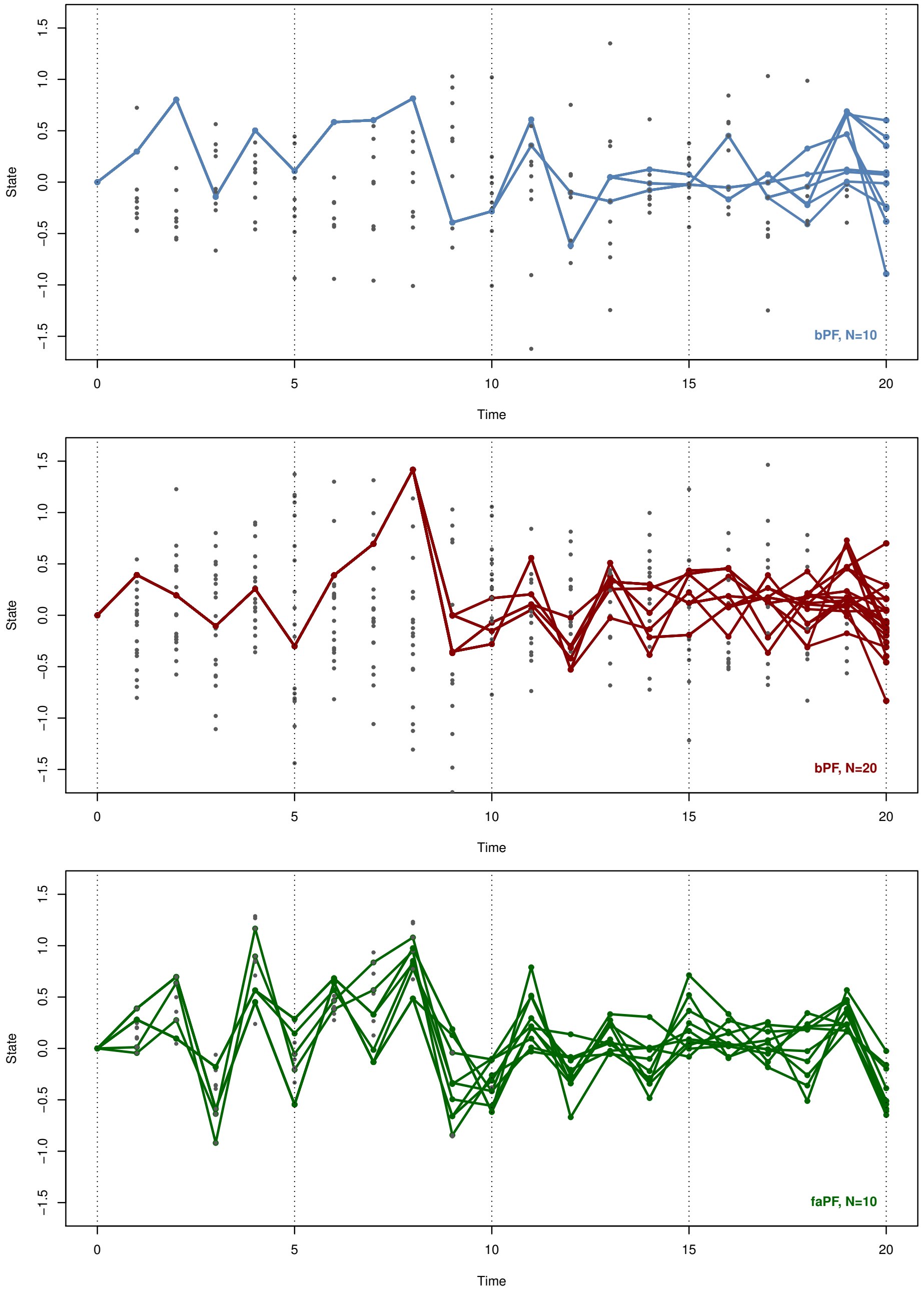

Figure 3.3: *The ancestral paths obtained from the bPF with 10 particles (upper), the bPF with 20 particles (middle) and the faPF with 10 particles (lower) with $T = 20$ in the GARCH(1,1) model in Example 3.3. The discarded particles are presented as gray dots.*

(3.8), we have that the expected value can be approximated by

$$\widehat{\varphi}_{\text{APF}} = \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \varphi\Big(x_t^{(i)}\Big). \tag{3.17}$$

Under some assumptions, it is possible to prove that this estimator is consistent and therefore converges in probability to the true expectation as the number of particles tend to infinity,

$$\widehat{\varphi}_{\text{APF}} \xrightarrow{p} \mathbb{E}\Big[\varphi(x)\Big], \qquad N \longrightarrow \infty.$$

Hence, this estimator is asymptotically unbiased and equivalent with some unknown *optimal filter*. However, for finite N the estimator is often biased and we return to this problem in Section 3.4.3.

We would also like to say something about the MSE of the estimator in (3.17). For this we assume that the function $\varphi(x)$ is bounded[2] for all x and some additional assumptions that are discussed by Crisan and Doucet (2002). It then follows that the MSE of the estimator obtained by the bPF can be upper bounded as

$$\mathbb{E}\Bigg[\Big(\widehat{\varphi}_{\text{APF}} - \mathbb{E}[\varphi(x)]\Big)^2\Bigg] \leq C_T \frac{\|\varphi\|^2}{N}, \tag{3.18}$$

where $\|\cdot\|$ denotes the supremum norm. Here, C_T denotes a function that possibly depends on T but is independent of N.

There exists numerous other results regarding the MSE of the estimator in (3.17). For example, it is possible to relax the assumption that $\varphi(x)$ should be bounded and that we only use the bPF with multinomial resampling. The resulting upper bounds have a similar structure to (3.18) but with different functions C. For more information, see Del Moral (2013) and Douc et al. (2014).

From this upper bound on the MSE, we would like to give some general recommendations regarding how to select N given T. However, this is difficult as the accuracy of the estimates is connected with the mixing property of the SSM (see Example 3.5). However, in practice it is recommended to use at least $N \propto T$ particles in the APF but sometimes even more particles are required to obtain reasonable estimates. Hence, we recommend that the user estimates the MSE for each model using e.g. a pilot run with some Monte Carlo simulations or by comparing with solution obtain from a particle smoother.

3.3.4 Estimation of the likelihood and log-likelihood

As previously discussed, the likelihood $\mathcal{L}(\theta)$ and log-likelihood $\ell(\theta)$ play important roles in both ML and Bayesian parameter inference. We also discussed that they are analytically intractable for nonlinear SSMs. However, we can obtain an unbiased estimate of the likelihood for any number of particles using the weights

[2] This is a rather restrictive assumption as it does not satisfied by the function $\varphi(x) = x$, which is used to compute the estimate of the filtered state $\widehat{x}_{t|t}$.

generated by the APF. To understand how this can be done, we consider the decomposition in Definition 2.1 and the fact that each predictive likelihood can be written as

$$\begin{aligned} p(y_t|y_{1:t-1}) &= \int p_\theta(y_t, x_t|y_{1:t-1})\,\mathrm{d}x_t \\ &= \int g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}) p_\theta(x_{t-1}|y_{1:t-1})\,\mathrm{d}x_{t-1:t}. \end{aligned}$$

If we consider the bPF algorithm, we can rewrite this as

$$\begin{aligned} p_{\mathrm{bPF}}(y_t|y_{1:t-1}) &= \int \frac{g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1})}{R_\theta(x_t|x_{t-1}, y_t)} R_\theta(x_t|x_{t-1}, y_t) p_\theta(x_{t-1}|y_{1:t-1})\,\mathrm{d}x_{t-1:t} \\ &= \int W_\theta(x_t, x_{t-1}) R_\theta(x_t|x_{t-1}, y_t) p_\theta(x_{t-1}|y_{1:t-1})\,\mathrm{d}x_{t-1:t} \end{aligned}$$

by the expression for the weight function (3.14) as $w_{t-1} = \nu_{t-1}$ for the bPF.

To approximate the predictive likelihood, we make use of that $\{\widetilde{x}_t^{(i)}, x_{t-1}^{(i)}\}_{i=1}^N$ is approximately distributed according to $R_\theta(x_t|x_{t-1}, y_t)p_\theta(x_{t-1}|y_{1:t-1})$. From this, it follows that

$$\begin{aligned} p_{\mathrm{bPF}}(y_t|y_{1:t-1}) &\approx \frac{1}{N}\sum_{i=1}^N \int W_\theta(x_t, x_{t-1}) \delta_{\widetilde{x}_t^{(i)}, x_{t-1}^{(i)}}(\mathrm{d}x_{t-1:t}) \\ &= \frac{1}{N}\sum_{i=1}^N W_\theta\big(\widetilde{x}_t^{(i)}, x_{t-1}^{(i)}\big) \\ &= \frac{1}{N}\sum_{i=1}^N w_t^{(i)}. \end{aligned}$$

It is also possible to show that the faPF leads to a similar estimator by replacing w_t with ν_t, see Pitt (2002) for the derivation. The resulting estimator for the likelihood using the APF (including both the bPF and faPF as special cases) is given by

$$\widehat{\mathcal{L}}(\theta) = \widehat{p}_\theta(y_{1:T}) = \frac{1}{N^{T+1}} \left\{ \sum_{i=1}^N w_T^{(i)} \right\} \left\{ \prod_{t=0}^{T-1} \sum_{i=1}^N \nu_t^{(i)} \right\}, \tag{3.19}$$

where the first summation is unity for the faPF and where $\nu_t = w_t$ for the bPF. To implement this estimator, we run Algorithm 2 and then calculate the likelihood estimate using (3.19) by inserting the particle weights generated by the APF.

The statistical properties of the likelihood estimator are studied by Del Moral (2004). It turns out that the estimator is consistent and unbiased for any $N \geq 1$. Furthermore, the error of the estimate satisfies a CLT,

$$\sqrt{N}\Big[\mathcal{L}(\theta) - \widehat{\mathcal{L}}(\theta)\Big] \xrightarrow{d} \mathcal{N}\Big(0, \psi^2(\theta)\Big), \tag{3.20}$$

for some asymptotic variance $\psi^2(\theta)$, see Proposition 9.4.1 in Del Moral (2004).

It is also straightforward to obtain an estimator for the log-likelihood from (3.19),

$$\widehat{\ell}(\theta) = \log \widehat{p_\theta(y_{1:T})} = \log \left\{ \sum_{i=1}^{N} w_T^{(i)} \right\} + \sum_{t=0}^{T-1} \log \left\{ \sum_{i=1}^{N} \nu_t^{(i)} \right\} - T \log(N+1). \quad (3.21)$$

However, this estimator is biased for a finite number of particles, but it is still consistent and asymptotically normal. This result follows from applying the *second-order delta method* (Casella and Berger, 2001) on (3.20). The resulting CLT for the log-likelihood estimator is given by

$$\sqrt{N}\left[\ell(\theta) - \widehat{\ell}(\theta) + \frac{\gamma^2(\theta)}{2N}\right] \xrightarrow{d} \mathcal{N}\Big(0, \gamma^2(\theta)\Big), \quad (3.22)$$

where we introduce $\gamma(\theta) = \psi(\theta)/\mathcal{L}(\theta)$. As a result, we have an expression for the bias of the estimator given by $-\gamma^2(\theta)/2N$ for a finite number of particles. Consequently, it is possible to compensate for the bias as the variance of the estimator $\gamma^2(\theta)$ can be estimated using Monte Carlo simulations by repeated application of the APF on the same data. This could be an interesting improvement for the proposed methods in Papers B and C, where we make use of the log-likelihood estimator.

3.4 Example: Bias and variance of the log-likelihood estimate

Consider the setup in Example 2.6 and the problem of estimating the log-likelihood at $\theta = \theta^\star$ using the faPF with $N = 10$ particles. We repeat the estimation over 1 000 Monte Carlo simulations using the same data. The error of the log-likelihood estimate is calculated using the true valued obtain from the Kalman filter.

In Figure 3.4, we present the histogram of the error together with a Gaussian approximation (upper), the box plot of the errors (lower left) and the QQ-plot of the errors (lower right). The Gaussian approximation fits the data quite well and the resulting average error is -0.03 with variance $\widehat{\gamma}^2(\theta^\star) = 0.05$. The predicted bias is calculated using (3.22) by $-\widehat{\gamma}^2(\theta^\star)/2\sqrt{N} = -0.01$. The QQ-plot validates the Gaussian assumption as we do not seen any deviating tail behaviour.

3.4 Particle smoothing

Particle smoothers approximate the solution to the smoothing problem in an SSM similar to how the APF approximates the corresponding filtering problem. However, there exists a number of different approaches to carry out the smoothing given the particle system from the APF. The simplest smoother is to make use of the APF to approximate the joint smoothing distribution as discussed in the previous. The main problem with this approach is that the path degeneracy problem limits the accuracy of the estimate. Another similar approach is to make use of the fixed-lag (FL) particle smoother (Kitagawa and Sato, 2001), which is based on using the APF to estimate the fixed-lag smoothing distribution (recall Table 3.1). In the following, we make use of this smoother as it has a low computational cost and a reasonable accuracy compared with other particle smoothers.

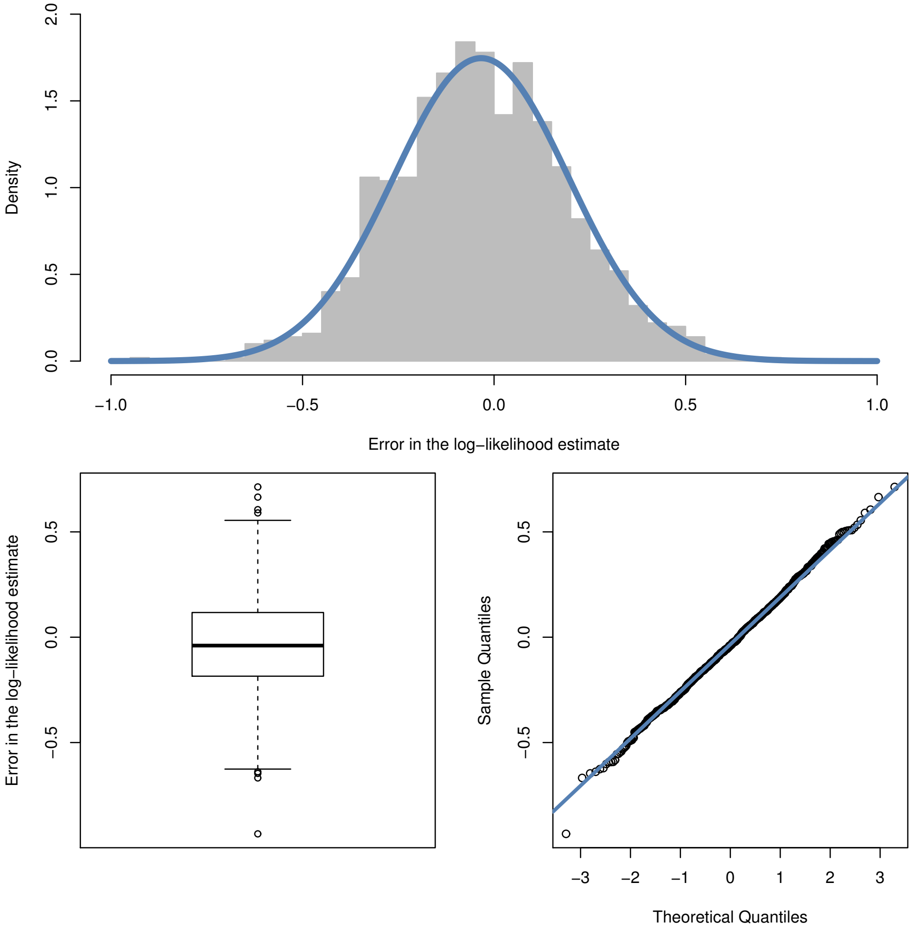

Figure 3.4: *The histogram with a Gaussian approximation (blue line) of the error in the log-likelihood estimates (upper) in the LGSS model using a faPF. The boxplot (lower left) and QQ-plot (lower right) supports the Gaussian assumption of the error.*

More advanced smoothers are often based on approximations of the Bayesian smoothing recursion (3.2). There are two main families of smoothers that results from this approach: the FFBSm and the forward filtering backward simulator (FFBSi). The original marginal FFBSm and FFBSi algorithms are discussed in Doucet et al. (2000) and Godsill et al. (2004), respectively. Another type of smoother makes use of two APFs (one running forward in time and the other backwards) and combines the output of these by using a *two-filter formula* (Briers et al., 2010; Fearnhead et al., 2010). Furthermore, other types of smoothers have been proposed in Bunch and Godsill (2013) and Dubarry and Douc (2011) using MCMC methods (discussed in the next chapter). The interesting improvement in these two smoothers are that they generate new particles in the backward sweep, which is not done in the FFBSm and FFBSi. See Lindsten and Schön (2013) for a recent survey of different particle smoothing methods.

In this section, we focus on the use of the FL smoother for estimating some quantities that are required for the proposed methods in Papers A and D. We introduce the underlying assumptions of the smoother and discuss how to implement it. We also show how it can be used to estimate the score function of an SSM and parts of the information matrix. We conclude by discussing the properties of the estimates obtained from the FL smoother.

3.4.1 State inference using the particle fixed-lag smoother

The FL smoother relies on the forgetting properties of an SSM, i.e. that the Markov chain quickly forgets about its earlier states. This property is illustrated in Example 3.5 for an LGSS model.

3.5 Example: Mixing property in the LGSS model

Consider the LGSS model using the same setup as in Example 2.6, where 20 different state processes are simulated during 8 time steps. Here, each state process has a randomly selected initial states distributed as $x_0 \sim \mathcal{N}(0, 20^2)$.

In Figure 3.5, we present the evolution of the state processes in three different LGSS models. We note that the value of ϕ determines the rate at which the processes converge to a stationary phase. A larger value of ϕ gives the process a longer memory and this results in that it requires longer time to forget its initial condition. That is, the state process mixes slowly and therefore future observations contain useful information about the current state.

The observation that some SSMs mixes quickly is the basis for the assumption that

$$p_\theta(x_{0:t}|y_{1:T}) \approx p_\theta(x_{0:t}|y_{1:\kappa_t}), \tag{3.23}$$

where $\kappa_t = \min\{T, t + \Delta\}$ and Δ denotes some lag determined by the user. This means that future observations contain a decreasing amount of information about the current state. The rate of this decrease is determined by the mixing of the model as discussed in Example 3.5. If the model mixes quickly, future observations have a limited amount of information about the current state as the state process

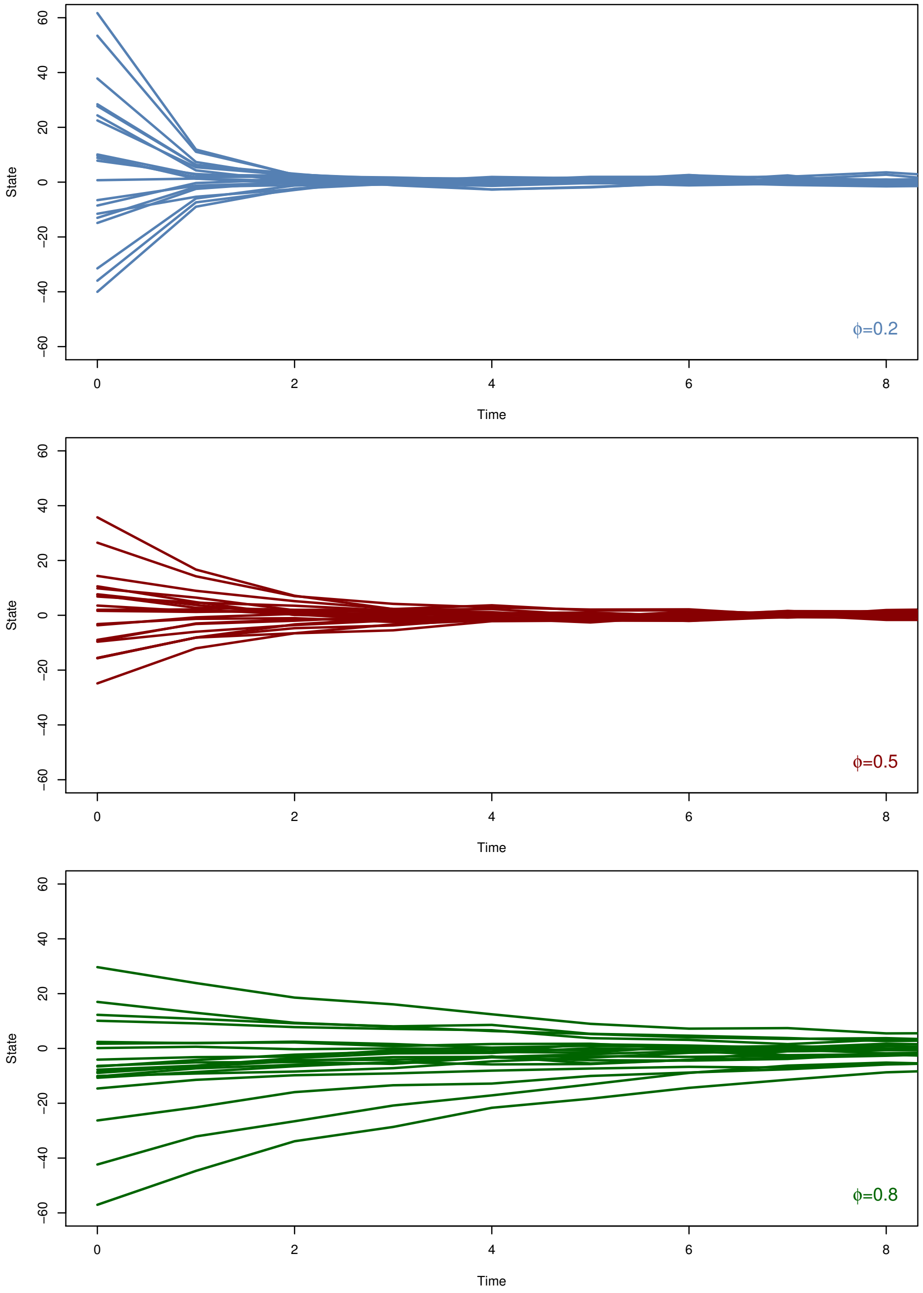

Figure 3.5: *The evolution of 20 different state processes in the LGSS model from Example 2.6 using different initial values. Three different values of ϕ are used in the LGSS model: $\phi = 0.2$ (upper), $\phi = 0.5$ (middle) and $\phi = 0.8$ (lower).*

Algorithm 3 Two-step fixed-lag (FL) particle smoother

INPUTS: $y_{1:T}$ (observations), $R_\theta(x_t|x_{0:t-1}, y_t)$ (propagation kernel), ν_t (auxiliary weights), $N > 0$ (no. particles) and Δ (lag).
OUTPUTS: $\widehat{p}_\theta(x_{t-1:t}|y_{1:T})$ for $t = 1$ to T (empirical two-step smoothing dist.).

```
Initialise each particle $x_0^{(i)}$.
for $t = 1$ to $T$ do
    Sample new the ancestor indices to obtain $\{a_t^{(i)}\}_{i=1}^N$ using (3.10).
    Propagate the particles to obtain $\{x_t^{(i)}\}_{i=1}^N$ using (3.12).
    Calculate new importance weights to obtain $\{w_t^{(i)}\}_{i=1}^N$ using (3.14).
    if $t > \Delta + 1$ then
        Compute $\kappa_t = \min\{t + \Delta, T\}$ and recover $\{x^{(i)}_{\kappa_t,t-1:t}\}_{i=1}^N$.
        Compute (3.24) to obtain $\widehat{p}_\theta(x_{t-1:t}|y_{1:T})$.
    end if
end for
```

quickly forgets its past. Hence, the lag Δ can be selected to be rather small and still make (3.23) a valid approximation. This is the main intuition behind the FL smoother and the procedure follows directly from the APF in Algorithm 2.

In the following, we require estimates of the two-step smoothing distribution $\widehat{p}_\theta(\mathrm{d}x_{t-1:t}|y_{1:\kappa_t})$ to estimate the score function and the information matrix for an SSM. By using the FL smoother assumption, we can compute the required estimate by a marginalisation over the joint smoothing distribution,

$$p_\theta(x_{t-1:t}|y_{1:\kappa_t}) = \int p_\theta(x_{0:\kappa_t}|y_{1:\kappa_t})\, \mathrm{d}x_{0:t-2}\, \mathrm{d}x_{t+1:\kappa_t}.$$

By inserting the empirical joint smoothing distribution $p_\theta(x_{1:\kappa_t}|y_{1:\kappa_t})$ from (3.9), an estimate of the two-step smoothing distribution is obtained as

$$\widehat{p}_\theta(\mathrm{d}x_{t-1:t}|y_{1:\kappa_t}) = \sum_{i=1}^{N} \widetilde{w}^{(i)}_{\kappa_t} \delta_{x^{(i)}_{\kappa_t,t-1:t}}(\mathrm{d}x_{t-1:t}), \tag{3.24}$$

where $x^{(i)}_{\kappa_t,t}$ denotes the ancestor particle from which the particle $x^{(i)}_{\kappa_t}$ originated from at time t. This ancestor is obtained by tracing the linage backwards similar to Figure 3.3, where the same is done for the particles starting at time T.

We summarise the procedure for estimating the two-step smoothing distribution using the FL-smoother in Algorithm 3. The marginal smoothing distribution can be computed using an analogue expression and this results in a similar procedure. Finally, we present an application of these methods for volatility estimation in Example 3.6.

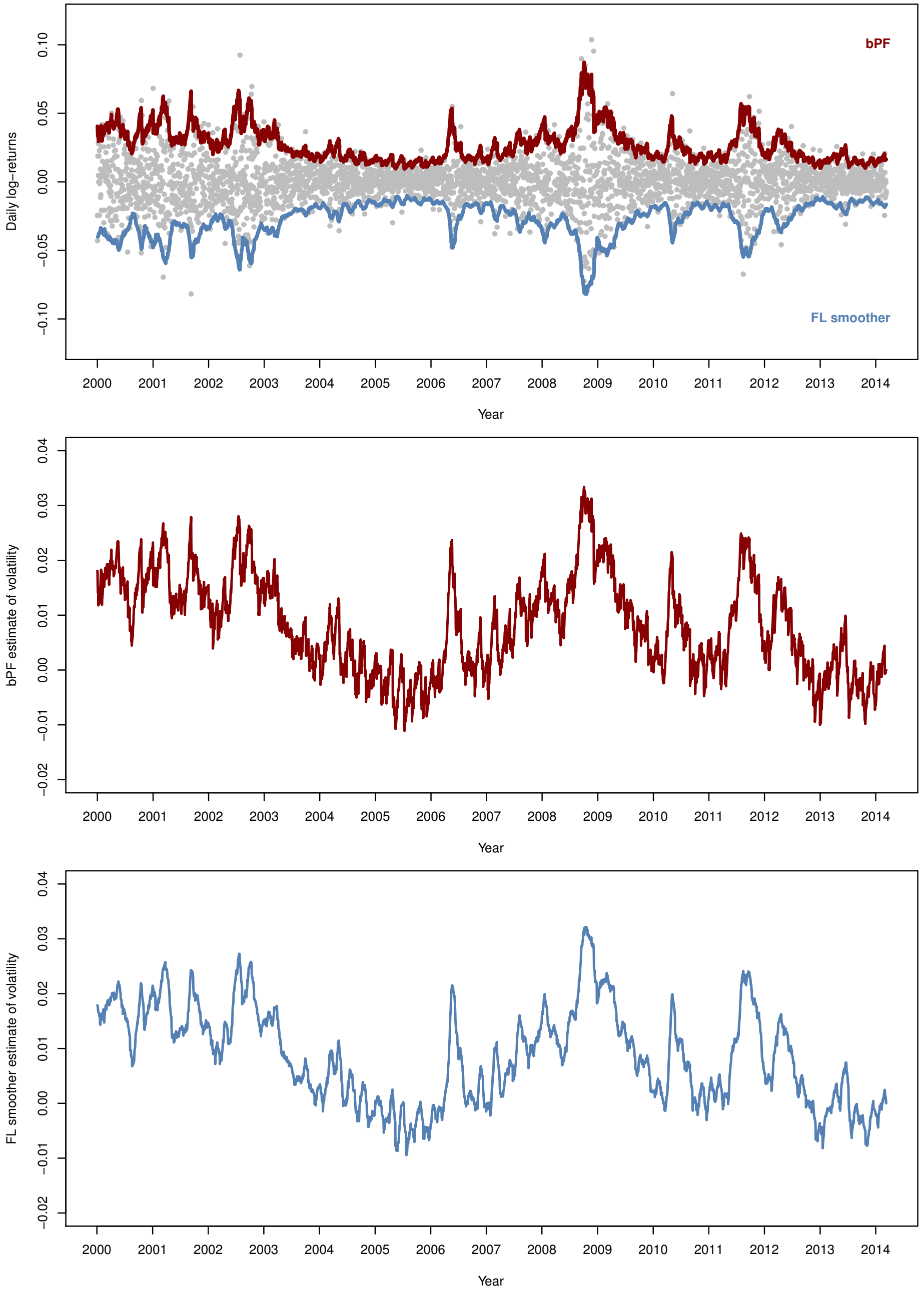

Figure 3.6: *Upper: the log-returns (gray dots) from the Nasdaq OMX Stockholm 30 Index presented in Figure 2.2. The estimated 95% CIs are also presented for the bPF (red) and the FL smoother (blue). Middle: the estimated latent volatility obtained from the bPF. Lower: the estimated latent volatility obtained from the FL smoother.*

3.6 Example: State inference in the Hull-White SV model

Consider the Hull-White SV model (2.4) with data from Section 2.2.2 and the parameter vector estimate from Example 3.1. To infer the latent volatility (the state), we apply the bPF using Algorithm 2 and the FL smoother according to Algorithm 3 using $N = 5\,000$ particles and $\Delta = 12$.

In the upper part of Figure 3.6, we present the filtered log-returns with a 95% CI for the bPF (red) and the FL smoother (blue). Here, we present the upper CI for the bPF and the lower CI for the FL smoother so that the two lines do not overlap. Note that the mean of the log-returns is zero and therefore the CIs are symmetric around zero. We conclude that the CIs seem reasonable as they cover most of the log-returns except at the financial crises. Also, the CI computed from the bPF is a bit rougher than the CI computed using the FL smoother.

In the middle and lower parts of Figure 3.6, we present the estimated volatilities using both methods. We see that the estimates correspond reasonable well with variation of the log-returns. Note that, the periods with larger volatility are connected with the financial crises.

3.4.2 Estimation of additive state functionals

In this section, we consider the use of particle smoothing to estimate the expected value of an additive functional given the observations. This is in analogue with the Monte Carlo estimate of the expectation of a function in (3.3). An *additive functional* satisfies the expression

$$S_\theta(x_{0:T}) = \sum_{t=1}^{T} s_{\theta,t}(x_{t-1:t}), \tag{3.25}$$

which means that we can decompose a function that depends on the entire particle trajectory into several functionals. Here, $s_{\theta,t}(x_{t-1:t})$ denotes some general functional that depends on only two states of the trajectory. This type of additive functionals occurs frequently in nonlinear SSMs when we would like to compute functions that depend on the kernels $f_\theta(x_{t+1}|x_t)$ and $g_\theta(y_t|x_y)$. In the following, we give some concrete examples of these functionals connected with parameter inference in SSMs. In these problems, we would like to compute the expected value of the additive functional given the observations,

$$\begin{aligned}\mathbb{E}\Big[S_\theta(x_{0:T})\big|y_{1:T}\Big] &= \int S_\theta(x_{0:T}) p_\theta(x_{0:T}|y_{1:T})\,\mathrm{d}x_{0:T} \\ &= \sum_{t=1}^{T} \int s_{\theta,t}(x_{t-1:t}) p_\theta(x_{t-1:t}|y_{1:T})\,\mathrm{d}x_{t-1:t}.\end{aligned} \tag{3.26}$$

This can be done by inserting the two-step smoothing distribution estimated by any particle filter or smoothing algorithm. Examples of some different approaches for this are found in Poyiadjis et al. (2011) using the APF and in Del Moral et al. (2010) using a forward smoother based on the FFBSm. In this section and in

Paper A, we discuss the use of the FL smoother for this application. The resulting estimate is obtained by inserting (3.24) into (3.26),

$$\widehat{S}_\theta(x_{0:T}) = \sum_{t=1}^{T} \sum_{i=1}^{N} \widetilde{w}_{\kappa t}^{(i)} s_{\theta,t}(x_{\kappa_t,t-1:t}^{(i)}), \tag{3.27}$$

which can be computed by some minor modifications of Algorithm 3.

We encounter this type of additive functionals are encountered in two common problems concerning SSMs: when estimating the score function and parts of the information matrix. In this thesis, we make use of the material in this section for estimating the latter two quantities in Papers A and D. To derive the expressions for the additive functions related to these problems, we consider the logarithm of the joint distribution of states and observations in an SSM,

$$\log p_\theta(x_{0:T}, y_{1:T}) = \log \mu(x_0) + \sum_{t=1}^{T} \Big[\log f_\theta(x_t|x_{t-1}) + \log g_\theta(y_t|x_t) \Big]. \tag{3.28}$$

By using this quantity, the score function can be estimated using *Fisher's identity* (Fisher, 1925; Cappé et al., 2005),

$$\mathcal{S}(\theta') = \nabla \ell(\theta)\big|_{\theta=\theta'} = \int \Big[\nabla \log p_\theta(x_{0:T}, y_{1:T})\big|_{\theta=\theta'} \Big] p_\theta(x_{0:T}|y_{1:T}) \, \mathrm{d}x_{0:T},$$

which results in the functional

$$\xi_{\theta,t}(x_{t-1:t}) = \nabla \log f_\theta(x_t|x_{t-1})\big|_{\theta=\theta'} + \nabla \log g_\theta(y_t|x_t)\big|_{\theta=\theta'}, \tag{3.29}$$

corresponding to the gradient of (3.28) evaluated at $\theta = \theta'$. An estimator of the score function is obtained by inserting the additive function (3.29) into the empirical distribution obtained by the FL smoother (3.27) as

$$\widehat{\mathcal{S}}(\theta') = \sum_{t=1}^{T} \sum_{i=1}^{N} \widetilde{w}_{\kappa t}^{(i)} \xi_{\theta,t}(x_{\kappa_t,t-1:t}^{(i)}).$$

The observed information matrix can be estimated using *Louis' identity* (Louis, 1982; Cappé et al., 2005). However, only some parts of the identity can be directly estimated using the FL smoother. The remaining parts must be estimated using the APF or a more advanced smoother like the FFBSm. To see this, we rewrite the observed information matrix using Louis' identity,

$$\mathcal{J}(\theta') = -\nabla^2 \ell(\theta)\big|_{\theta=\theta'} = \Big[\nabla \ell(\theta)\big|_{\theta=\theta'} \Big]^2 - \Big[\nabla^2 \mathcal{L}(\theta)\big|_{\theta=\theta'} \Big] \Big[\mathcal{L}(\theta) \Big]^{-1}.$$

Here, the second term of the identity can be written as

$$\begin{aligned} \Big[\nabla^2 \mathcal{L}(\theta)\big|_{\theta=\theta'} \Big] \Big[\mathcal{L}(\theta) \Big]^{-1} &= \int \Big[\nabla \log p_\theta(x_{0:T}, y_{1:T})\big|_{\theta=\theta'} \Big]^2 p_\theta(x_{0:T}|y_{1:T}) \, \mathrm{d}x_{0:T} \\ &+ \int \Big[\nabla^2 \log p_\theta(x_{0:T}, y_{1:T})\big|_{\theta=\theta'} \Big] p_\theta(x_{0:T}|y_{1:T}) \, \mathrm{d}x_{0:T}. \end{aligned} \tag{3.30}$$

The first term in (3.30) cannot be directly estimated by the FL smoother, as it does not provide approximations of the densities needed. Instead, it can be estimated using the APF directly as proposed by Poyiadjis et al. (2011) or by the use of a combination of the FL smoother and the APF. The latter alternative is discussed in Paper A. However, the second term in (3.30) can be estimated using the FL smoother directly as it can be written as an additive functional given by

$$\zeta_{\theta,k}(x_{t-1:t}) = \nabla^2 \log f_\theta(x_t|x_{t-1})\big|_{\theta=\theta'} + \nabla^2 \log g_\theta(y_t|x_t)\big|_{\theta=\theta'}. \tag{3.31}$$

corresponding to the Hessian of (3.28) evaluated at $\theta = \theta'$. An estimator for this part of Louis' identity is obtained by inserting the additive function (3.31) into the empirical distribution obtained by the FL smoother (3.27). The expected information matrix can be estimated as the sample covariance matrix of a large number of score functions estimated using different data sets. We discuss this method further in Paper D.

We have now seen two examples of additive functionals in connection with parameter inference in SSMs and how to estimate them using the FL smoother. The same setup can also be used in the expectation maximisation algorithm (Dempster et al., 1977; McLachlan and Krishnan, 2008) as discussed in Del Moral et al. (2010) using a forward smoother. We return to this problem in Section 4.1, where we discuss different methods for parameter inference. We end this section with Example 3.7, where we estimate the log-likelihood, score and observed information matrix for a nonlinear SSM using the FL smoother.

3.7 Example: Score and information matrix in the Hull-White SV model

Consider the problem of estimating the log-likelihood, the score function and the natural gradient (the score function divided by the observed information matrix) for $\theta = \phi$ in the HWSV model (2.4). We again make use of the data from Section 2.2.2 and the parameter vector estimate from Example 3.1. For this model, the additive functional connected with the score function (3.29) is

$$\begin{aligned}\xi_{\theta,t}(x_{t-1:t}) &= \nabla\Big[-\frac{1}{2}\log\big(2\pi\sigma_v^2\big) - \frac{1}{2\sigma_v^2}\big(x_{t+1}-\phi x_t\big)^2\Big]\Big|_{\phi=\phi'} \\ &\quad + \nabla\Big[-\frac{1}{2}\log\Big(2\pi\beta^2\exp(x_t)\Big) - \frac{y_t^2\exp(-x_t)}{2\beta^2}\Big]\Big|_{\phi=\phi'} \\ &= x_t\big(x_{t+1}-\phi' x_t\big),\end{aligned}$$

and similar for the observed information matrix (3.31),

$$\zeta_{\theta,t}(x_{t-1:t}) = \nabla\Big[x_t\big(x_{t+1}-\phi x_t\big)\Big]\Big|_{\phi=\phi'} = -x_t^2.$$

Here, we make use of the FL smoother in Algorithm 3 for estimating (3.27) with lag $\Delta = 12$ and $N = 5\,000$ particles. We vary the parameter on a grid within the interval $\phi \in [0.70, 0.99]$ and estimate the required quantities at each grid point. Here, we fix $\{\sigma_v, \beta\}$ to their estimated value. The observed information matrix is estimated using the combination of the APF and the FL smoother introduced in Paper A.

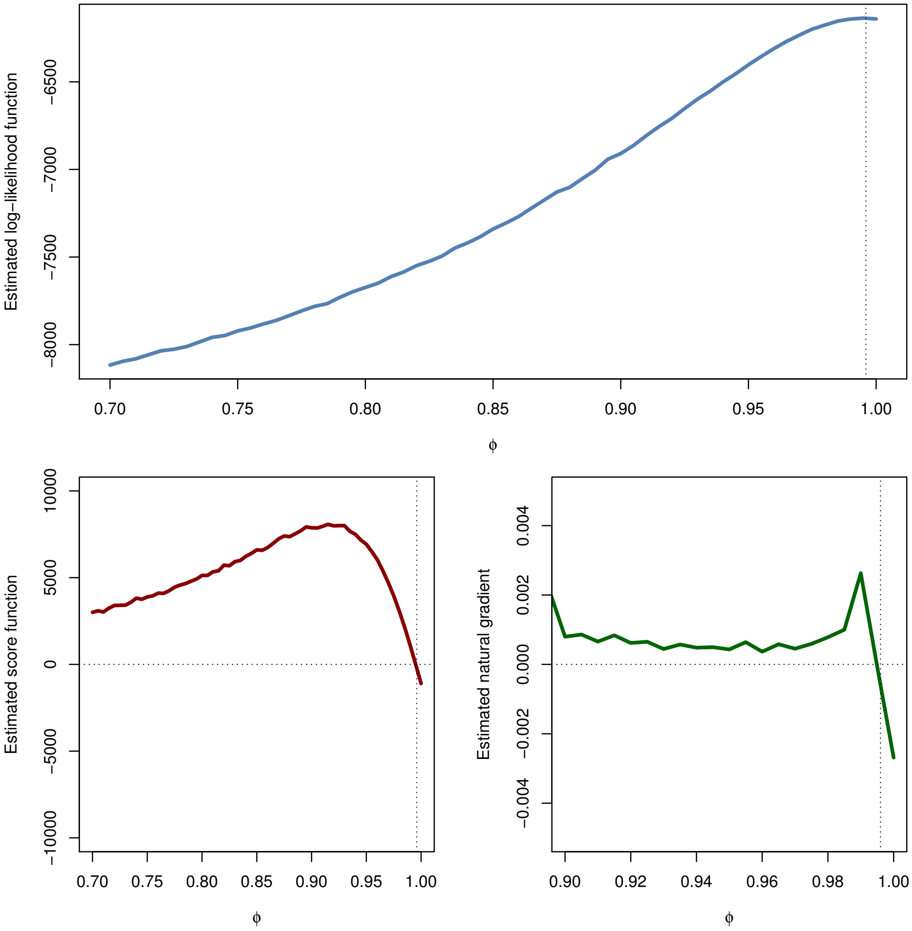

Figure 3.7: *The estimates of the log-likelihood function (upper) the score function (lower left) and the natural gradient (lower right) of the HWSV model in Example 3.7. The dotted lines indicate the true parameters and the zero level.*

The resulting estimates are presented in Figure 3.7. We see that the log-likelihood and score estimates seems reasonable and have a rather low variance. The natural gradient is more difficult to estimate due to that the observed information matrix estimates are noisy. We see that the sign of the estimate is correct but the gradient is noisy and rather small.

3.4.3 Statistical properties of the particle fixed-lag smoother

We conclude the discussion of particle smoothing by reviewing some results regarding the statistical properties of the estimates obtained by the FL smoother for the additive functionals. In Olsson et al. (2008), the authors analyse the bias and variance of the estimates from the APF and the FL smoother, respectively. The variances of the estimates are concluded (under some regularity conditions) to be upper bounded by quantities proportional to $T^2/\sqrt{N}$ and $(T \log T)/\sqrt{N}$ for the APF and the FL smoother, respectively.

The bias is also analysed in Olsson et al. (2008) and the authors conclude (under some regularity conditions) that the bias is upper bounded by quantities proportional to T^2/N and $\lambda + (T \log T)/N$, respectively. Here, λ denotes a quantity which is independent of the number of particles. Hence, the FL smoother gives a biased estimate for all N whereas the bias of the estimate obtained from the APF decreases as $1/N$.

In the analysis, it is assumed that the lag is selected according to $\Delta \propto c \log T$, where $c > -1/\log \rho$ and $\rho \in [0, 1]$ denotes a measure of the mixing of the model (Olsson et al., 2008). If Δ is too small, the underlying approximation of the FL smoother (3.23) and the accuracy of the estimate are rather poor. This is the result of that more information about the current state is available in future observations that are not taken into account by the smoother. If instead Δ is too large, we get problems with path degeneracy and in the limit when $\Delta = T$, the APF estimate is recovered. The choice of Δ is therefore crucial for obtaining good estimates from the FL smoother and its optimal value depends on both the number of the observations and the mixing properties of the model. Hence, it must be tailored for each application individually as is further discussed in Paper A.

Finally, we mention that even better accuracy can be obtained by using other more advanced smoothing algorithms. The main drawback with these algorithm is that their computational cost is larger than for the APF and the FL smoother, which have a computational cost that is proportional to $\mathcal{O}(NT)$. For example, the FFBSm algorithm (Doucet et al., 2000) has a computational cost that is proportional to $\mathcal{O}(N^2T)$. The benefit of using this smoother is that the variance of the estimate grows proportional to $\mathcal{O}(T)$, instead of as $\mathcal{O}(T^2)$ and $\mathcal{O}(T \log T)$ for the APF and the FL smoother, respectively. These properties are discussed in more detail in Del Moral et al. (2010) and Poyiadjis et al. (2011). There are also some new promising particle smoothers with a computational complexity proportional to $\mathcal{O}(TN \log N)$. For more information, see Klaas et al. (2006), Taghavi et al. (2013) and Gray (2003).

3.5 SMC for Image Based Lighting

In Sections 3.3 and 3.4, we have discussed the application of SMC to the filtering and smoothing problems in SSMs. However, the SMC algorithm can be applied to other problems that are sequential in nature and where the target distribution $\pi(x_{0:k})$ grows over some index $k = 1, \ldots, K$. It is also possible to make use of SMC methods for static problems by defining an artificial target that grows over k even if the original target does not. This approach is discussed by Del Moral et al. (2006) and opens up for using SMC for a wide range of problems.

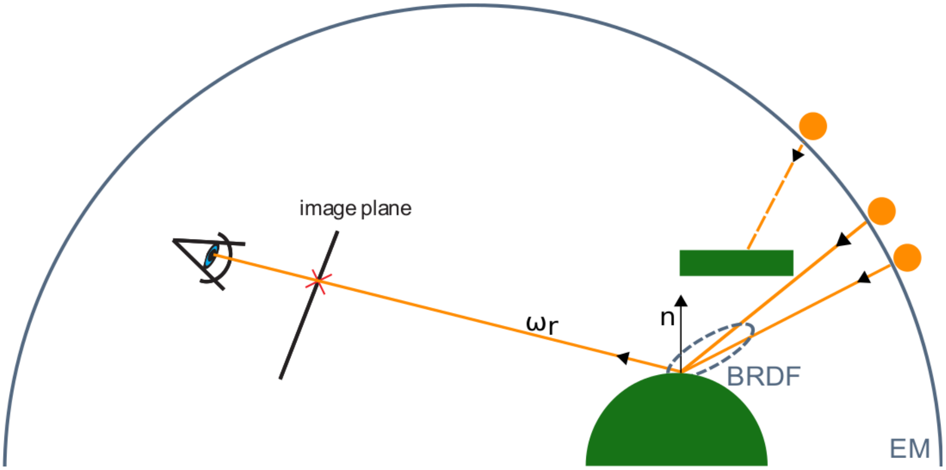

Figure 3.8: *The setup used in the IBL approach, where the LTE gives the outgoing radiance at the angle ω_r that hits the image plane. This radiance is computed by taking the EM, the BRDF and the visibility into account. Note, that one of the light sources are occluded in this scene and does not contribute to the outgoing radiance.*

We exemplify the usefulness of this general class of SMC algorithm by returning to the problem of rendering a sequence of photorealistic images using IBL (Debevec, 1998; Pharr and Humphreys, 2010) as discussed in Section 1.1.2. In Figure 3.8, we present a cartoon of the setup of the LTE. Here, we are interested in calculating the outgoing radiance $L_{r,k}(\omega_r)$ from a point at the outgoing angle ω_r at frame k, which is given by the LTE as

$$L_{r,k}(\omega_r) = \int f_r(\omega_i \to \omega_r) L_k(\omega_i) V(\omega_i)(\omega_i \cdot \mathbf{n}) \, \mathrm{d}\omega_i, \tag{3.32}$$

where $\mathbf{n}$ denotes the surface normal and ω_i denotes incoming angles of the light rays that hits the object at n. Here, f_r, L_k and V denote the *bidirectional reflectance distribution function* (BRDF), the EM in frame k and the binary visibility function, respectively. Furthermore, we assume that these functions can be evaluated point-wise. In this problem, the BRDF describes the optical properties of the object and can be seen as a distribution over incoming angles.

To see how SMC can be used to solve (3.32), we define this as a sequence of normalisation problems, where kth unnormalised target distribution is given by

$$\gamma_k(\omega_i) = f_r(\omega_i \to \omega_r) L_k(\omega_i)(\omega_i \cdot \mathbf{n}),$$

where we neglect the visibility term for computational convenience. Calculating the outgoing radiance (without regarding visibility) $L_{r,k}(\omega_r)$ can therefore be seen as normalisation,

$$L_{r,k}(\omega_r) = \int \gamma_k(\omega_i) \, \mathrm{d}\omega_i,$$

for a sequence of EMs indexed by k. From this setup, we see that the desired particles corresponds to incoming angles ω_i of the light that hits the object and contributes to the outgoing radiance. As previously discussed, the challenge is to select only a few important angles to take into account when solving the LTE. Otherwise, the problem becomes computational infeasible.

In the filtering problem, this normalisation factor is the marginal likelihood of the problem, which can be computed using the particle weights. In a similar manner, we can compute $L_r(\omega_r)$ and the only difference between the APF in Algorithm 2 is the choice of propagation kernel. Here, we do not have an SSM to describe the dynamical behaviour of the EM between frames. Instead, we make use of an MCMC kernel to adapt the particles between EMs. Details of this approach are found in Kronander et al. (2014a) and Ghosh et al. (2006).

Another approach to solve (3.32) is to use the IS algorithm directly on $L_{r,k}(\omega_r)$ for each frame individually. This can be done with an algorithm similar to SIR, where we first sample from the EM and compute the particle weights using the BRDF. The resulting particles are obtained after a resampling step. This method is referred to as the *bidirectional importance sampling* (BIS) (Burke et al., 2005) algorithm. This approach is computationally cheap, but cannot make use of information from the previous frame when estimating $L_{r,k}(\omega_r)$ in the current frame k. As previously discussed, this problem can be solved with the SMC setup that we consider here. We conclude this section by Example 3.8 in which we compare these two approaches.

3.8 Example: SMC for direct illumination

Consider a simple setup, where we would like to render a sphere given a sequence of HDR images. This is similar to the problem considered in Section 1.1.2 but simpler since we do not consider multiple bounces. Hence, this problem is referred to as *direct illumination*. Here, we compare four different approaches to render the sphere in the image. The first and second methods are to make use of the IS algorithm to sample directly from the EM and the BRDF, respectively. Hence, we disregard the information in the BRDF and the EM, respectively. We also make use of the BIS algorithm and the SMC algorithm previously outlined.

In Figure 3.9, we compare the four different approaches at frame $k = 10$ in a sequence of EMs. The first three methods solves the problem for each frame individually and the SMC approach makes use of the previous information. We see that using the IS algorithm directly to sample from the EM does not perform well. The reason is that we consider a mirror sphere, where the BRDF is rather narrow, which means that only light from a few incoming angles contribute to the outgoing radiance. The remaining three methods gives comparable results. Therefore, we conclude that there could be advantages by using the SMC algorithm for this application. See Kronander et al. (2014a) for a more detailed discussion and comparison of these approaches.

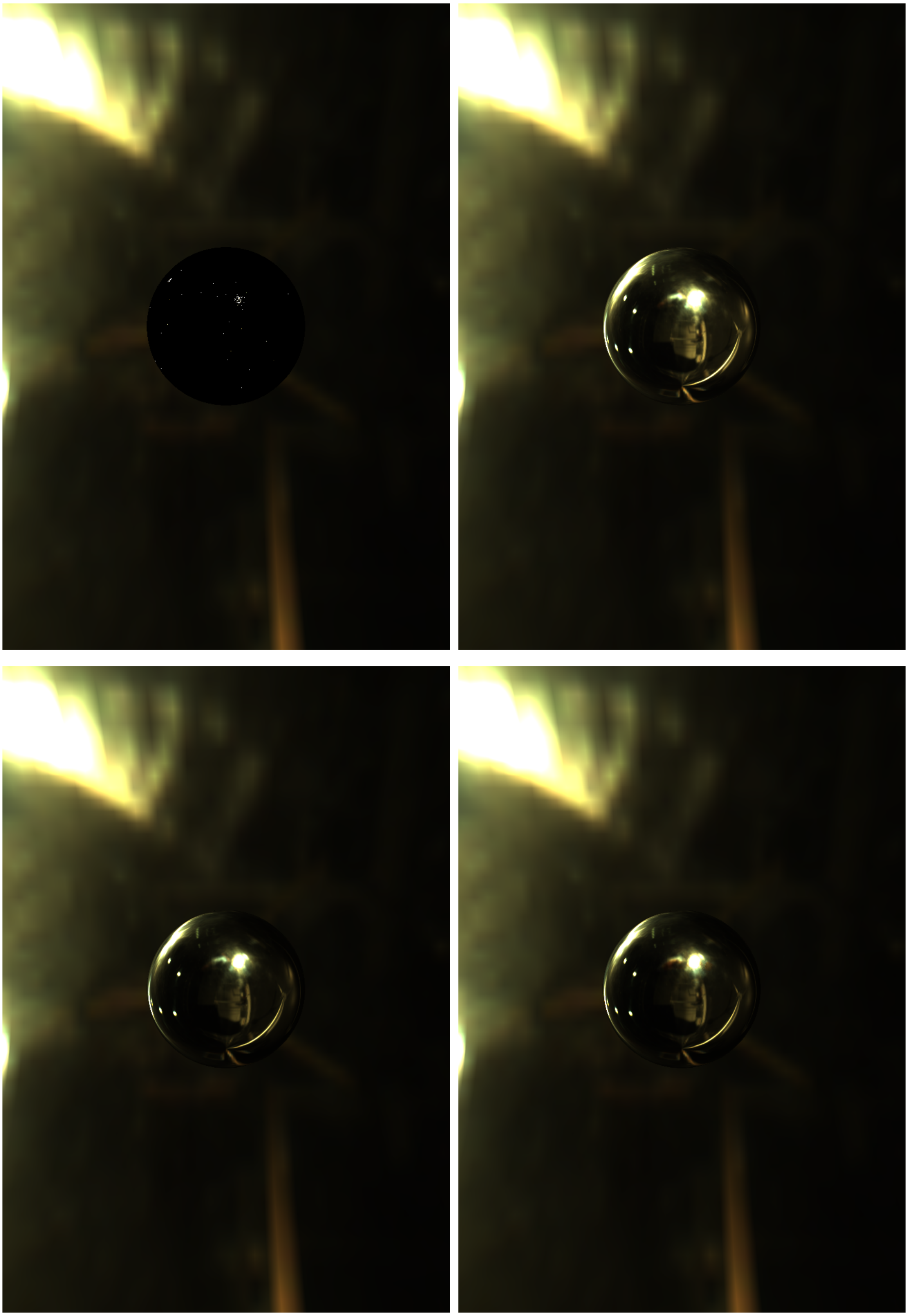

Figure 3.9: *Mirror sphere, frame 10 with different sampling techniques: IS the EM (upper left), IS the BRDF (upper right), BIS (lower left), and the SMC algorithm (lower right).*

4

Parameter inference using sampling methods

In this chapter, we consider some sampling based methods for ML and Bayesian parameter inference in SSMs. We start by giving a small overview of some of the current algorithms in the field. Later, we introduce and discuss the use of MCMC (Robert and Casella, 2004) and particle MCMC (PMCMC) (Andrieu et al., 2010) approaches to Bayesian parameter inference. Here, we also discuss the use of Langevin and Hamiltonian dynamics to improve the efficiency of the method. This material is later used in Paper A to construct a particle version of these MCMC algorithms.

The second part of this chapter is devoted to discussing the BO algorithm (Jones, 2001; Boyle, 2007; Lizotte, 2008; Osborne, 2010) and how it can be used together with GPs (Rasmussen and Williams, 2006) for ML parameter inference and input design. This approach is used in Papers B and C for ML based parameter inference and MAP based parameter inference, respectively.

4.1 Overview of computational methods for parameter inference

There are many different methods developed for parameter inference in SSMs using both ML and Bayesian methods. Here, we review some of these methods to give an overview of the area and discuss some of the more popular methods. For more exhaustive accounts of different methods, see Douc et al. (2014), Kantas et al. (2009), Cappé et al. (2007) and Cappé et al. (2005).

4.1.1 Maximum likelihood parameter inference

Most parameter inference problems in the ML framework cannot be solved by analytical calculations. Instead popular approaches are based on optimisation and other iterative algorithms to solve the inference problem in Definition 2.2. From Chapter 3, we know that the APF can be used to calculate the log-likelihood and that the FL smoother can be used to estimate the score function (the gradient of the log-likelihood). Hence, we can make use of a *gradient-based optimisation algorithm* to maximise the log-likelihood and thereby estimate the parameters of the model. This method operates by an iterative application of

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \epsilon_k \widehat{\mathcal{S}}(\theta_k), \tag{4.1}$$

where $\widehat{\theta}_k$ and ϵ_k denote the current estimate of the parameter vector and the step length, respectively. This approach is used by Poyiadjis et al. (2011) and Yildirim et al. (2013) for parameter inference in SV models with Gaussian returns and α-distributed returns, respectively. Furthermore, we can estimate the Hessian information by a particle smoother and make use of it in (4.1). This results in a *Newton optimisation algorithm* which operates by an iterative application of

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \epsilon_k \widehat{\mathcal{J}}^{-1}(\theta_k) \widehat{\mathcal{S}}(\theta_k). \tag{4.2}$$

Another approach is to estimate the Hessian on-the-fly in a Quasi-Newton algorithm using finite difference approximation. A popular member of this class of algorithms is the *simultaneous perturbation stochastic approximation* (SPSA) algorithm discussed in Spall (1987). The SPSA algorithm is used for inference in the αSV model in Ehrlich et al. (2012).

The *expectation maximisation* algorithm (Dempster et al., 1977; McLachlan and Krishnan, 2008) is another popular alternative for ML inference. In this method, the latent states are seen as missing information and are estimated together with the parameter vector of the model. The quantity required by the expectation maximisation algorithm is an additive functional that can be estimated using a particle smoother. Many different versions of the expectation maximisation algorithm are available in the literature using different particle smoothers. Some examples are Del Moral et al. (2010) using a forward smoother, Olsson et al. (2008) using the FL smoother, Lindsten (2013) using the conditional particle filter with ancestor sampling and Schön et al. (2011) using the FFBSm.

Finally, we mention another approach based on *Bayesian optimisation* (Jones, 2001; Boyle, 2007; Lizotte, 2008), introduced in Papers B and C for ML parameter inference in SSMs where the likelihood can be intractable. We return to this approach in Section 4.4, where we introduce the method in detail.

4.1.2 Bayesian parameter inference

As for the ML parameter inference approach, most interesting Bayesian parameter inference problems are analytically intractable. Some problems can be solved in closed-form using conjugate priors as discussed in Section 2.4. However, in general we require approximate methods to solve expectation, marginalisation and normalisation problems in Bayesian inference. Previously, we discussed the sampling based approach in which we make use of statistical simulations and computers to approximate the posterior. Examples of some sampling based approaches are MCMC methods or SMC methods. Some recent methods that makes us of the latter method are *SMC-squared* (SMC^2) (Chopin et al., 2013) and *particle learning* (Carvalho et al., 2010). An alternative class of methods are based on approximate analytical computations, which often makes use of iterated approximations to approximate the posterior. Examples of analytical approximations are the *Laplace approximation*, the *integrated nested Laplace approximation* (INLA) (Rue et al., 2009), *variational Bayes* (VB) (Bishop, 2006) and *expectation propagation* (EP) (Minka, 2001).

In this thesis, we are primarily interested in sampling based inference using MCMC (Robert and Casella, 2004) for Bayesian parameter inference. MCMC is a family of algorithms that all are based on simulating a Markov chain with the target as its stationary distribution. After some *burn-in*, the chain reaches stationarity and the samples obtained from the chain can be treated as independent samples from the parameter posterior by the *ergodic theorem* discussed in Section 4.2.1. Here, we consider two instances of MCMC algorithms, the *Metropolis-Hastings* (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) and the *Gibbs sampler* (Geman and Geman, 1984). The Gibbs sampler can be seen as a special case of the more general MH algorithm. As such it can only be used in models with a certain conditional structure of the joint parameter posterior. We return to the Gibbs sampler in Paper E for parameter inference in a non-Gaussian ARX model.

The MH algorithm is a general method for sampling from intractable distributions and is applied in many different fields. Some applications of the MH algorithm are found in biology (Wilkinson, 2011), system identification (Ninness and Henriksen, 2010), finance (Johannes and Polson, 2009) and statistics (Robert and Casella, 2004). A major problem with this algorithm is that it requires evaluations of the likelihood of the SSMs. As discussed in Section 3.3.4, we can only obtain an unbiased estimator of this quantity, but it turns out that this is enough. As a result of the unbiasedness property, the resulting Markov chain keeps the target as its stationary distribution. This type of methods are known as *psuedo-marginal* algorithms (Andrieu and Roberts, 2009; Andrieu et al., 2010) and this family includes the *particle MH* (PMH) and *particle Gibbs* (PG) algorithms. This opens up for Bayesian parameter inference in SSMs. We return to discussing the PMH algorithm in Section 4.3 and propose refinements to it in Paper A.

4.2 Metropolis-Hastings

In this section, we discuss the use of the MH algorithm for sampling from the parameter posterior $p(\theta|y_{1:T})$. Here, we give a short introduction to the MH algorithm and also briefly discuss why it works and discuss some interesting extensions of the algorithm based on random walks on Riemann manifolds. Interested readers are referred to Robert and Casella (2004), Gelman et al. (2013) and Liu (2008) for more detailed accounts of the algorithm and its general application.

The MH algorithm samples the *target distribution* $\pi(\theta) = p(\theta|y_{1:T})$ by simulating a carefully constructed Markov chain on the parameter space Θ. The Markov chain is constructed in such a way that it admits the target as its unique stationary distribution. The algorithm consists of two steps: (i) a new parameter θ'' is sampled from a *proposal distribution* $q(\theta''|\theta')$ given the current state θ' and (ii) the current parameter is changed to θ'' with *acceptance probability* $\alpha(\theta'', \theta')$, otherwise the chain remains at the current parameter. The acceptance probability is given by

$$\alpha(\theta'', \theta') = 1 \wedge \frac{\pi(\theta'')}{\pi(\theta')} \frac{q(\theta'|\theta'')}{q(\theta''|\theta')} = 1 \wedge \frac{p(\theta'')}{p(\theta')} \frac{p_{\theta''}(y_{1:T})}{p_{\theta'}(y_{1:T})} \frac{q(\theta'|\theta'')}{q(\theta''|\theta')}, \tag{4.3}$$

where $a \wedge b \triangleq \min\{a, b\}$ and $p_\theta(y_{1:T})$ denotes the likelihood. The resulting procedure is outlined in Algorithm 4 for parameter inference in models where we can compute the likelihood exactly. Here, we have used the form of the parameter posterior from (2.14), where the marginal likelihoods cancel in the acceptance probability. Therefore, the algorithm only requires that we can point-wise evaluate the unnormalised target distribution.

Note that the IS algorithm could be used to estimate the parameter posterior using e.g. a multivariate Gaussian distribution as the proposal as in Example 3.1. However, if θ is high dimensional this approach would require many samples to accurately estimate the posterior. In the MH algorithm, we could instead construct a Markov chain that explores the posterior distribution by local moves thus exploiting the previously accepted parameter. Hence, it focuses its attention to areas of the parameter space in which the posterior assigns a relatively large probability mass. This makes sampling of high dimensional target more efficient, as less iterations of the algorithm are required to obtain an accurate estimate.

To see this, assume that we have a *symmetric* proposal $q(\theta''|\theta') = q(\theta'|\theta'')$, so that the ratio between the proposals cancel in (4.3). The remaining part is the ratio between the target evaluated at θ'' and θ'. If the proposed parameter θ'' results in that the target assumes a larger value than in θ', it is always accepted. Also, we accept a proposed parameter with some probability if this results in a small decrease of the target compared with the previous iteration. This results in that the MH sampler both allows for the algorithm to climb the posterior to its mode and explore the area surrounding the mode. Hence, the MH algorithm can possibly escape local extrema which is a problem for many local optimisation algorithms used in numerical ML parameter inference.

The performance of the MH algorithm is dependent on the choice of the proposal distribution. A poor proposal leads to a poor exploration of the posterior distribution, which results in that many iterations of the algorithm are required to obtain a good approximation of the posterior. Here, we discuss some common choices of proposals which are needed for the discussion in Section 4.2.2 and in Paper A. The perhaps simplest example is the *independent* proposal in which $q(\theta''|\theta') = q(\theta'')$, i.e. a parameter is proposed without taking the previously accepted parameter into

Algorithm 4 Metropolis-Hastings (MH) for Bayesian inference in SSMs

INPUTS: $M > 0$ (no. MCMC steps), $q(\,\cdot\,)$ (proposal) and θ_0 (initial parameter).
OUTPUT: $\theta = \{\theta_1, \ldots, \theta_M\}$ (samples from the parameter posterior).

```
Initalise using $\theta_0$.
for $k = 1$ to $M$ do
    Sample $\theta'$ from the proposal $\theta' \sim q(\theta' | \theta_{k-1})$.
    Calculate the likelihood $p_{\theta'}(y_{1:T})$.
    Sample $\omega_k$ from $\mathcal{U}(0, 1)$.
    if $\omega_k < \alpha(\theta', \theta_{k-1})$ given by (4.3) then
        $\theta_k \leftarrow \theta'$. {Accept the parameter}
    else
        $\theta_k \leftarrow \theta_{k-1}$. {Reject the parameter}
    end if
end for
```

account. This proposal cannot exploit the previous accepted parameters and this could be a difficulty in inference problems when the posterior is quite complicated. However, if the proposal is similar to the posterior, this leads to a good mixing of the Markov chain as the proposed parameters are uncorrelated. This insight have been used in Giordani and Kohn (2010) to construct a proposal distribution based on a mixture of Gaussian distributions, which results in an efficient MH sampler in some applications.

Another popular choice is the (symmetric) Gaussian random walk (RW) with some step length ϵ, which results from selecting $q(\theta'' | \theta') = \mathcal{N}(\theta''; \theta', \epsilon^2)$. The choice of the step length in the RW proposal determines the mixing of the Markov chain and thereby the efficiency of the exploration of the parameter posterior. If the step length is too small, the exploration is poor since that it takes a long time to explore the target. If it is too large, we seldom accept the proposed parameter as the difference in the values that the posterior assume is too large, resulting in a small acceptance probability. This is often referred to as the *mixing of the Markov chain* (c.f. Example 3.5). That is we would like to balance the mixing of the Markov chain to get a reasonable exploration of the posterior and acceptance rate. This problem is illustrated in Example 4.1, where we make use of Algorithm 4 to infer the parameters in an LGSS model for which we can compute the likelihood exactly. We return to the choice of proposal and the mixing of the Markov chain in following sections.

4.1 Example: Parameter inference in the LGSS model

Consider the parameter inference problem in the LGSS model (2.2) with the unknown parameter $\theta = \phi$. We use the parameter $\theta^\star = 0.5$ together with $\{\sigma_v, \sigma_e\} = \{1.0, 0.1\}$ to generate a realisation from the model with $T = 250$ and known initial state $x_0 = 0$. Here, we use the MH algorithm defined in Algorithm 4 together with the Kalman filter for estimating the likelihood. A Gaussian random walk proposal is used with zero mean and variance ϵ. For simplicity, we use a uniform prior over

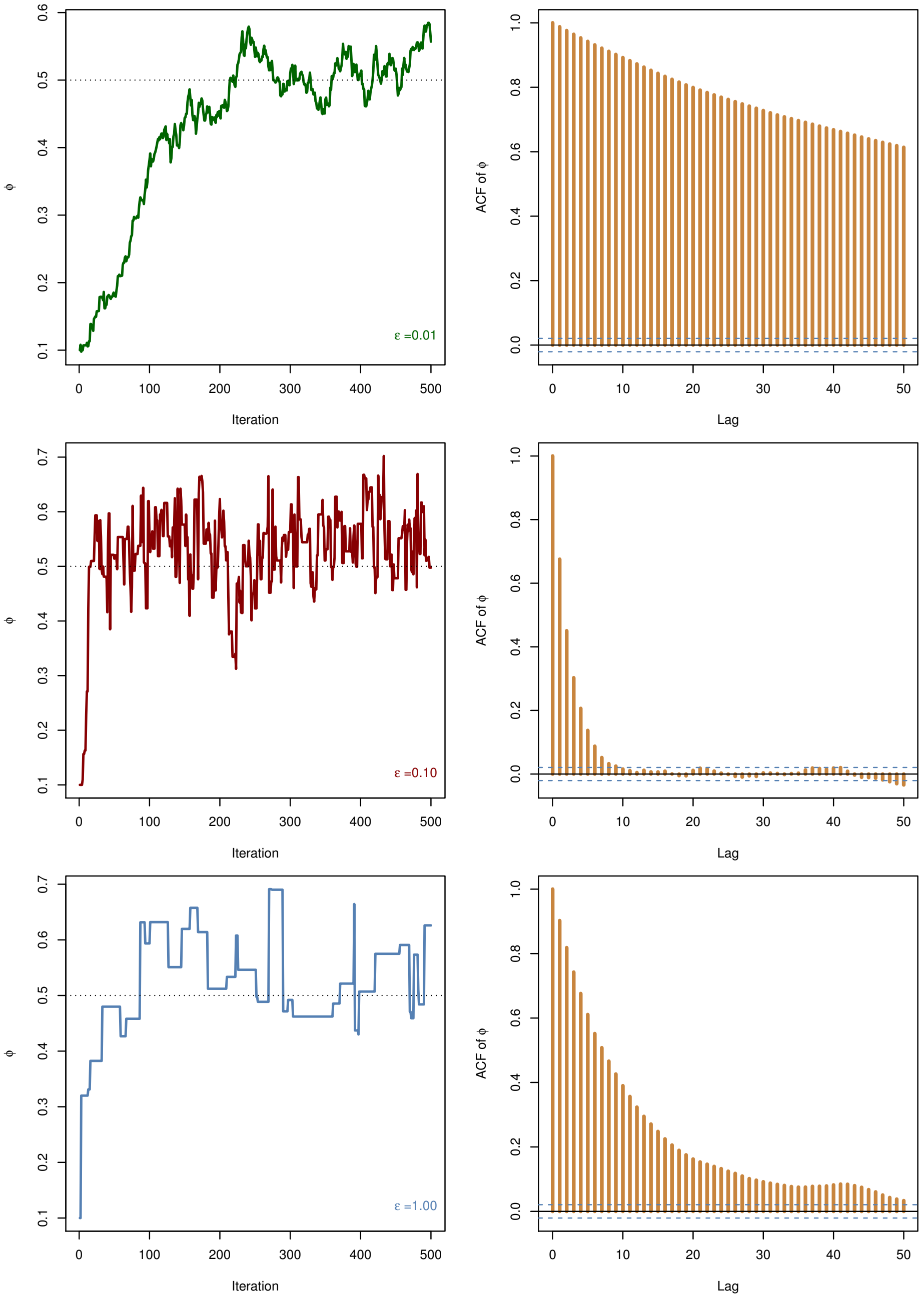

Figure 4.1: *The traces (left) of the first 500 iterations of ϕ in the LGSS model (2.2) using the RW proposal with three different step lengths $\epsilon = 0.01$ (upper), $\epsilon = 0.10$ (middle) and $\epsilon = 1.00$ (lower). The estimated autocorrelations function (right) using 9 000 iterations corresponding to each step length in the RW proposal. The dotted lines show the true parameters from which the data was generated.*

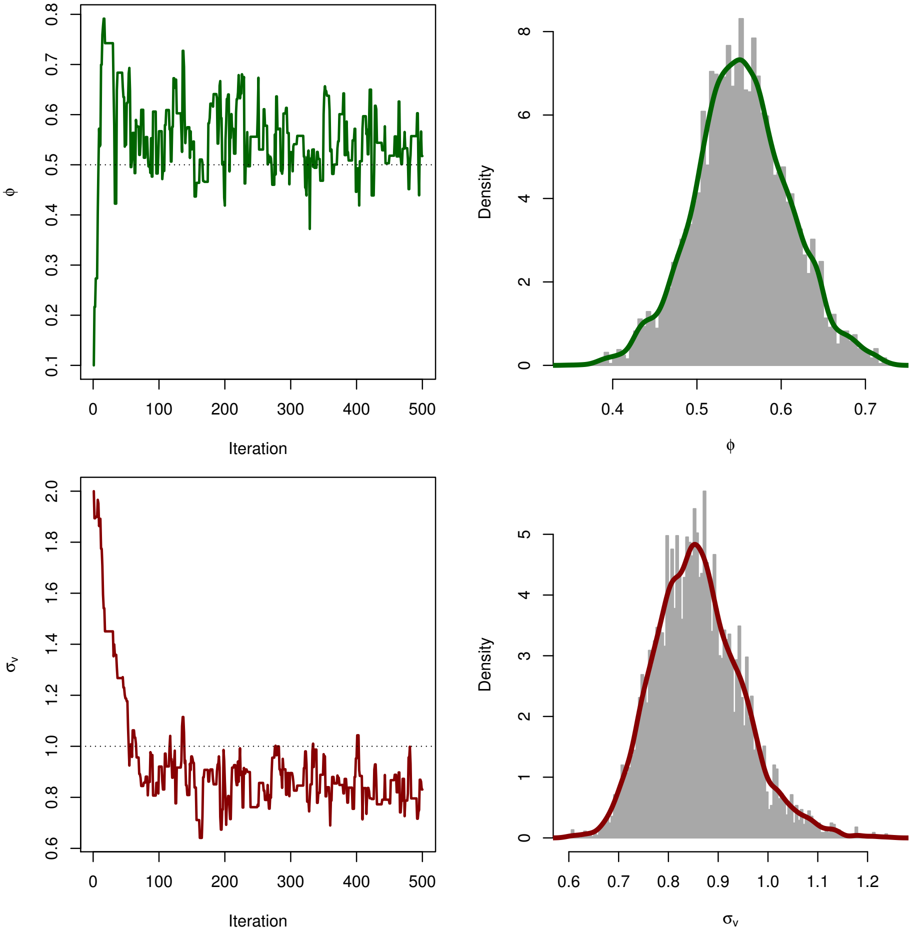

Figure 4.2: *The trace of the first 500 iterations of ϕ (upper left) and σ_v (lower left) in the LGSS model in Example 4.1. The estimated parameter posteriors (right) obtained from 9 000 iterations of the MH algorithm. The dotted lines show the true parameters from which the data was generated.*

$\phi \in (-1, 1) \subset \mathbb{R}$ to ensure that the system is stable at all times.

In Figure 4.1, we present the trace plot of the Markov chain together with the estimated autocorrelation function from $M = 10\,000$ iterations (discarding the first 1 000 iterations as burn-in) for the RW proposal with three different step lengths. We see that the mixing is rather poor for the smallest step size $\epsilon = 0.01$, as this results in a poor exploration of the posterior and therefore a large correlation in the Markov chain. This is also the case if the step length is too large as illustrated by the proposal with step length $\epsilon = 1.00$. Here, the large step length results is a small acceptance probability and therefore a larger correlation in the Markov chain.

We now add the parameter σ_v to our inference problem and would therefore like to infer the parameters $\{\phi, \sigma_v\}$ using the same setup as before. Here, we also add an uniform prior over $\sigma_v \in \mathbb{R}_+$ as it corresponds to a standard deviation. We use the step length $\epsilon = 0.1$ with the RW proposal and simulate the Markov chain for $M = 10\,000$ iterations (discarding the first 1 000 iterations as burn-in). In Figure 4.2, we present the trace of each parameter and the resulting posterior density estimate. We see that the Markov chain mixes well in both parameters and that the posterior estimates rather close to the *true* parameters.

4.2.1 Statistical properties of the MH algorithm

In this section, we discuss some of the statistical results that the MH algorithm relies upon and briefly mention their underlying assumptions. For more information about the properties of the MH algorithm and MCMC algorithms in general, see e.g. Tierney (1994), Robert and Casella (2004) and Meyn and Tweedie (2009). The core result that is used in the MH algorithm is the *ergodic theorem* (Tierney, 1994; Robert and Casella, 2004). Given a well-behaved test function φ, we can construct a Monte Carlo estimator,

$$\widehat{\varphi}_{\text{MH}} = \frac{1}{N} \sum_{i=1}^{N} \varphi(\theta_i),$$

where $\{\theta_i\}_{i=1}^{M}$ denotes the samples obtain from the MH algorithm. If the Markov chain is ergodic, then by the ergodic theorem this estimator is *strongly consistent*, i.e.

$$\widehat{\varphi}_{\text{MH}} \xrightarrow{a.s.} \mathbb{E}\big[\varphi(\theta)\big], \qquad N \to \infty.$$

Note that this property does not follow directly from the SLLN as the samples obtained for the posterior are not IID, as the proposed parameters are correlated. Hence, the ergodic theorem can be seen as the SLLN for correlated samples. Also, a CLT for the error in the estimator (under some conditions) is given by

$$\sqrt{N}\Big[\widehat{\varphi}_{\text{MH}} - \mathbb{E}\big[\varphi(\theta)\big]\Big] \xrightarrow{d} \mathcal{N}\big(0, \sigma_{\varphi}^2\big),$$

where σ_φ^2 denotes the variance of the estimator, which is proportional to the mixing properties of the Markov chain. In fact, this variance is proportional to the *integrated autocorrelation time* (IACT), which is given by

$$\mathsf{IACT}(\theta_{1:M}) = 1 + 2 \sum_{k=1}^{\infty} \rho_k(\theta_{1:M}).$$

The IACT cannot be computed analytically in practice as we do not know the true autocorrelation function for many models. Therefore, we often approximate it by

$$\widehat{\mathsf{IACT}}(\theta_{1:M}) = 1 + 2 \sum_{k=1}^{K} \widehat{\rho}_k(\theta_{1:M}),$$

where $\widehat{\rho}_k(\,\cdot\,)$ denotes the empirical autocorrelation function at lag k and K denotes some maximum lag for which to compute the IACT. This value can be selected as a fixed value or by the first index at which the ACF becomes statistically insignificant, i.e. the first index K such that $|\widehat{\rho}_K(\theta_{1:M})| < 2/\sqrt{M}$. Another related measure is the *effective sample size* (ESS) defined as

$$\mathsf{ESS}(\theta_{1:M}) = \frac{M}{\mathsf{IACT}(\theta_{1:M})} = \frac{M}{1 + 2\sum_{k=1}^{\infty} \rho_k(\theta_{1:M})}, \tag{4.4}$$

which can be approximated in the same manner as the IACT. The IACT and ESS have the interpretations as the number of iterations between each independent sample and the number of independent samples obtained from the posterior, respectively. Hence, we would like to minimise the IACT and maximise the ESS to get an optimal mixing in the Markov chain and many uncorrelated samples from the posterior. We illustrate the ESS values for parameter inference in an LGSS model in Example 4.2.

4.2 Example: ESS values for inference in the LGSS model

We return to Example 4.1 and calculate the ESS values for the three different Markov chains considered in Figure 4.1. The ESS values (4.4) are $\{22, 1292, 353\}$ when calculated using the adaptive method for the RW proposal with each of the three step lengths, respectively. These number validates the previous discussion that the proposal with $\epsilon = 0.10$ is preferable for this problem, as this maximises the number of uncorrelated samples from the posterior.

The consistency results and the CLT relies on the ergodic theorem, which assumes that the Markov chain is *irreducible* and *aperiodic*. Irreducible means that we should be able to get from any state to any state in the state space. Aperiodicity means that the Markov chain does not return to a state with regular intervals, i.e. no loops that the chain can get stuck in. These requirements need to be check for each MH algorithm to validate its assumptions. However, often in practice these assumptions hold for at least unimodal parameter posteriors.

Another property that is important for a Markov chain used in MCMC is *reversible*.

This property holds if the chain satisfies the *detailed balance* equation,

$$\pi(\theta'')K(\theta'',\theta') = \pi(\theta')K(\theta',\theta''),$$

where $K(\theta'',\theta')$ denote the Markov transition kernel. This results in that we can write

$$\int \pi(\theta'')K(\theta'',\theta')\,\mathrm{d}\theta'' = \int \pi(\theta')K(\theta',\theta'')\,\mathrm{d}\theta'' = \pi(\theta')\int K(\theta',\theta'')\,\mathrm{d}\theta'' = \pi(\theta'),$$

which shows that π is an invariant distribution of K, i.e. admits $\pi(\theta)$ as its stationary distribution. To show that the MH algorithm, satisfies the detailed balance equation we follow Liu (2008) and write the transition kernel of the Markov chain as,

$$K(\theta'',\theta') = q(\theta'',\theta')\min\left\{1, \frac{\pi(\theta')q(\theta',\theta'')}{\pi(\theta'')q(\theta'',\theta')}\right\}, \qquad \theta'' \neq \theta'$$

which gives

$$\pi(\theta'')K(\theta'',\theta') = \min\{\pi(\theta'')q(\theta'',\theta'), \pi(\theta')q(\theta',\theta'')\},$$

which is a symmetric function in θ'' and θ'. This results in that the detailed balances is fulfilled and the resulting Markov chain obtained from the MH algorithm is reversible.

4.2.2 Proposals using Langevin and Hamiltonian dynamics

We have previously discussed two different proposals for the MH algorithm and that these could perform better than the IS algorithm when the target is high dimensional. It turns out that the RW proposal in the MH algorithm still scales rather poorly as the dimension of the parameter space d increases, see Roberts et al. (1997). That is, the mixing of the Markov chain decreases and more iterations are required to maintain the number of independent samples from the target. This is a result of that the random walk does not explore the parameter space well when the dimension increases.

A modification to the random walk proposal that could increase the mixing in the Markov chain is to add a drift term that is proportional to the gradient of the log-target. This leads to a proposal which is the noisy equivalent of (4.1). In statistics, the resulting algorithm is known as the *Metropolis adjusted Langevin algorithm* (MALA) (Roberts and Rosenthal, 1998; Neal, 2010), where the proposal is said to follow a Langevin dynamics. This means that the proposal can be seen as the discrete version of a continuous time *Langevin diffusion* process (Øksendal, 2010; Kloeden and Platen, 1992). The Langevin diffusion with stationary distribution $p(\theta|y_{1:T})$ is given by the stochastic differential equation,

$$\mathrm{d}\theta(\tau) = \frac{1}{2}\Big[\nabla \log p(\theta|y_{1:T})\big|_{\theta=\theta(\tau)}\Big]\,\mathrm{d}\tau + \mathrm{d}B(\tau),$$

where $B(\tau)$ denotes a Brownian motion. To obtain samples from the parameter posterior, we can simulate the Langevin diffusion to stationarity, which is useful in the proposal as it guides the process towards the mode of the posterior. The discrete time proposal used in MALA follows from a first order Euler discretisation of (4.5) as

$$q(\theta''|\theta') = \mathcal{N}\left(\theta''; \theta' + \frac{\epsilon^2}{2}\mathcal{S}(\theta'), \epsilon^2 I_d\right),$$

where ϵ denotes a step length of the discretisation and the corresponding proposal. The proposal can also be derived using a Laplace approximation of the log-posterior as discussed in Paper A. Another version of this algorithm is the manifold MALA (mMALA) (Neal, 2010; Girolami and Calderhead, 2011) which also includes the Hessian information of the log-posterior in analogue with Newton algorithms (c.f. (4.2)). This proposal can be derived similar to the MALA proposal and has the form

$$q(\theta''|\theta') = \mathcal{N}\left(\theta''; \theta' + \frac{\epsilon^2}{2}\mathcal{J}^{-1}(\theta')\mathcal{S}(\theta'), \epsilon^2\mathcal{J}^{-1}(\theta')\right),$$

where we include the observed information matrix $\mathcal{J}(\theta)$ into the proposal. We make use of the MALA and mMALA proposals in Paper A to construct particle versions of the algorithms for parameter inference in SSMs. Alternative algorithms that solves the same problem are proposed by Girolami and Calderhead (2011), where the MALA and mMALA are used for parameter inference in the SV model.

In this thesis, we adopt an optimisation mind set and refer to the MALA and mMALA algorithm as first order and second order proposals, respectively. The names of the proposals refer to the use of first order information (the gradient) and the second order information (the Hessian) in the proposals. In the optimisation literature, this corresponds to the first order gradient-based search method and and second order Newton method, respectively. The corresponding MH algorithms are therefore called MH1 and MH2, respectively.

Another perspective of the second order proposal is that the added gradient and Hessian information about the log-target is used to construct a random walk on a Riemann manifold (Livingstone and Girolami, 2014; Girolami and Calderhead, 2011). For this end, we require that the negative Hessian is a PD matrix which is not always the case when we use the observed information matrix as the second order information. In Girolami and Calderhead (2011), it is proposed that the expected information matrix should be used instead, but this is computational costly to estimate for an SSM. Therefore, we make use of methods from optimisation (Nocedal and Wright, 2006) to make the observed information matrix PD if necessary. This is done by *regularisation*, i.e. adding an appropriate diagonal matrix to make the negative eigenvalues positive, see Paper A for more information.

Another related method is called manifold Hamiltonian MCMC (mHMC) and it can improve the mixing of the Markov chain further. This method originates from physics and is one instance of *hybrid MC* (Duane et al., 1987) algorithms, which are used in statistical physics to simulate from high dimensional targets. Here, we shortly discuss their use in a statistical setting for parameter inference. Interested readers are referred to Liu (2008), Neal (2010) and Girolami and Calderhead (2011) for more information. The mHMC algorithm is based on simulating a physical system with the *Hamiltonian* (the total energy),

$$H(\theta, p) = U(\theta) + K(p), \tag{4.5}$$

where $p \sim \mathcal{N}(0, \mathcal{J}^{-1}(\theta))$ denotes a random fictitious momentum for each parameter. Here, the *potential energy function* and the *kinetic energy function* are given by $U(\theta) = -\log \pi(\theta)$ and $K(p) = p^\top \mathcal{J}^{-1}(\theta)p/2$, respectively. The proposal simulates this Hamiltonian system for a number of steps L using the so-called *leap-frog algorithm* (Neal, 2010) with some step size ϵ. The result from this procedure is the proposed parameter and the resulting momentum $\{\theta'', p''\}$. This pair is accepted/rejected using a similar procedure as in the MH algorithm. The acceptance probability compares the Hamiltonian of the system at the last accepted parameter and the proposed parameter by

$$\alpha(\theta'', \theta') = 1 \wedge \exp\big(H(\theta'', p'') - H(\theta', p')\big). \tag{4.6}$$

As the accept/reject decision is delayed for L steps, this proposal allows for the chain to move a larger distance between the iterations of the algorithm. This increases the mixing of the Markov chain and it also allows for the Markov chain to visit isolated modes in the posterior. This leads to a better exploration of the posterior as well as more effective samples. The mHMC algorithm is used in many applications and is currently a popular algorithm in statistics and machine learning, see e.g. Chen et al. (2014), Beam et al. (2014) and Betancourt and Girolami (2013). In Girolami and Calderhead (2011), the authors make use of the mHMC algorithm for parameter inference in a SV model with impressive results. To adapt the mHMC algorithm to the PMCMC framework is therefore an exciting opportunity for future research.

We conclude this section with an comparison between the three MCMC algorithms previously discussed. In Example 4.3, we replicate and extend an illustration from Neal (2010) to compare the different methods when sampling from a high-dimensional target distribution.

4.3 Example: Parameter inference in a 100-dimensional Gaussian dist.

Consider the problem of sampling from a non-isotropic 100-dimensional Gaussian distribution with zero-mean and covariance matrix $\Sigma = \mathsf{diag}(1.00, 0.99, \ldots, 0.01)$. In this problem, we use random step sizes where $\epsilon_{\text{RW}} \sim \mathcal{U}(0.018, 0.026)$ for MH-RW and $\epsilon_{\text{HMC}} \sim \mathcal{U}(0.010, 0.016)$ for HMC with $L = 150$. These values follow the suggestions by Neal (2010). For MALA, we us the step length $\epsilon_{\text{LMC}} \sim \mathcal{U}(0.008, 0, 013)$, which results in an acceptance rate of about 60%. The HMC algorithm is executed for 1 000 iterations and the RW and MALA algorithms are executed for 150 000 iterations but only every 150th iteration is considered to make a fair comparison with the HMC. The resulting acceptance probabilities are 0.26, 0.60 and 0.88, for each proposal respectively.

The trace plots and ACF for the first coordinate (with standard deviation 1) are presented in Figure 4.3. The estimated mean and standard deviations of the target distribution are presented in Figure 4.4. The HMC algorithm gives almost

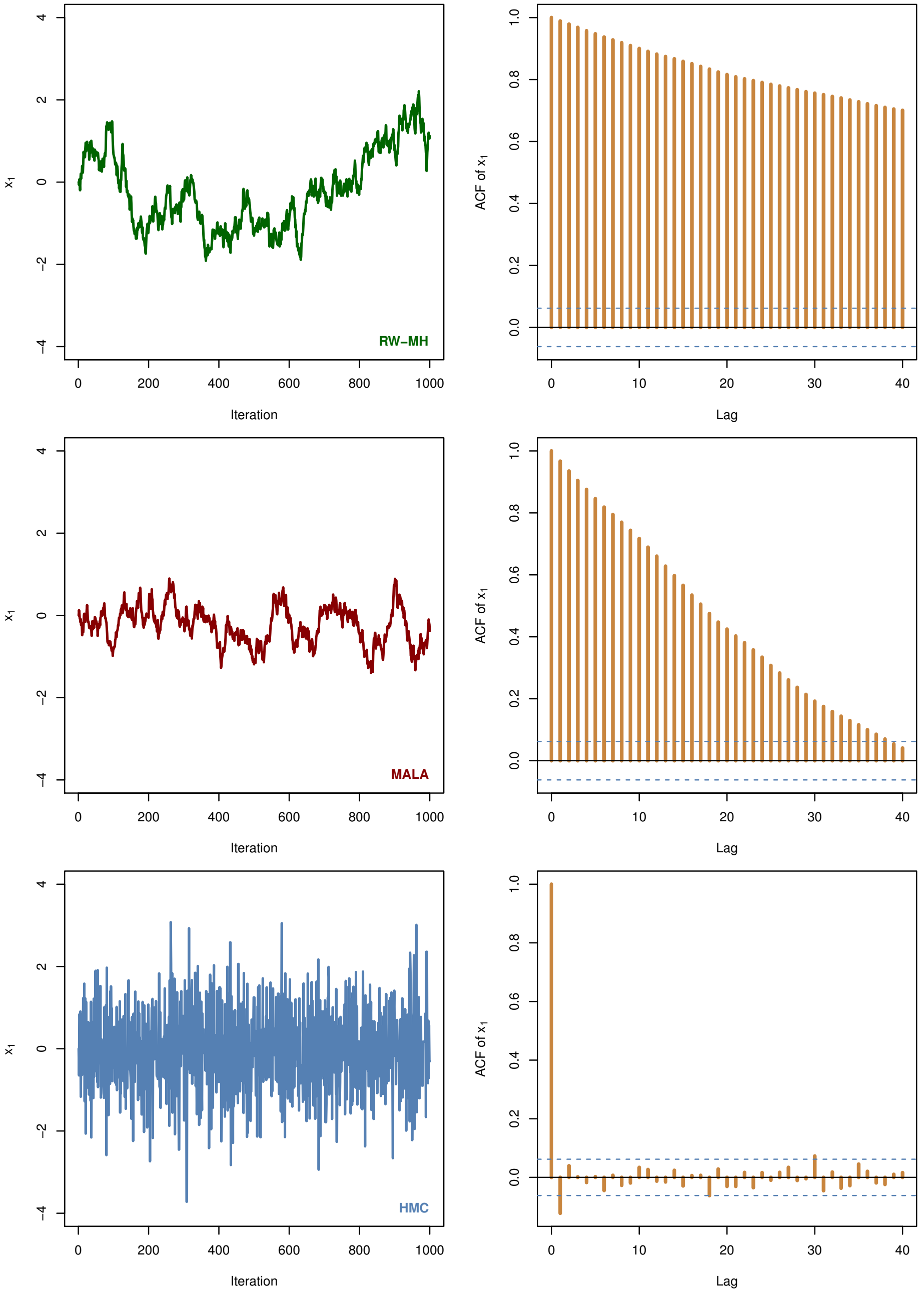

Figure 4.3: *The trace of x_1 (left) and the corresponding estimated ACF (right) for the RW-MH, MALA and HMC algorithms.*

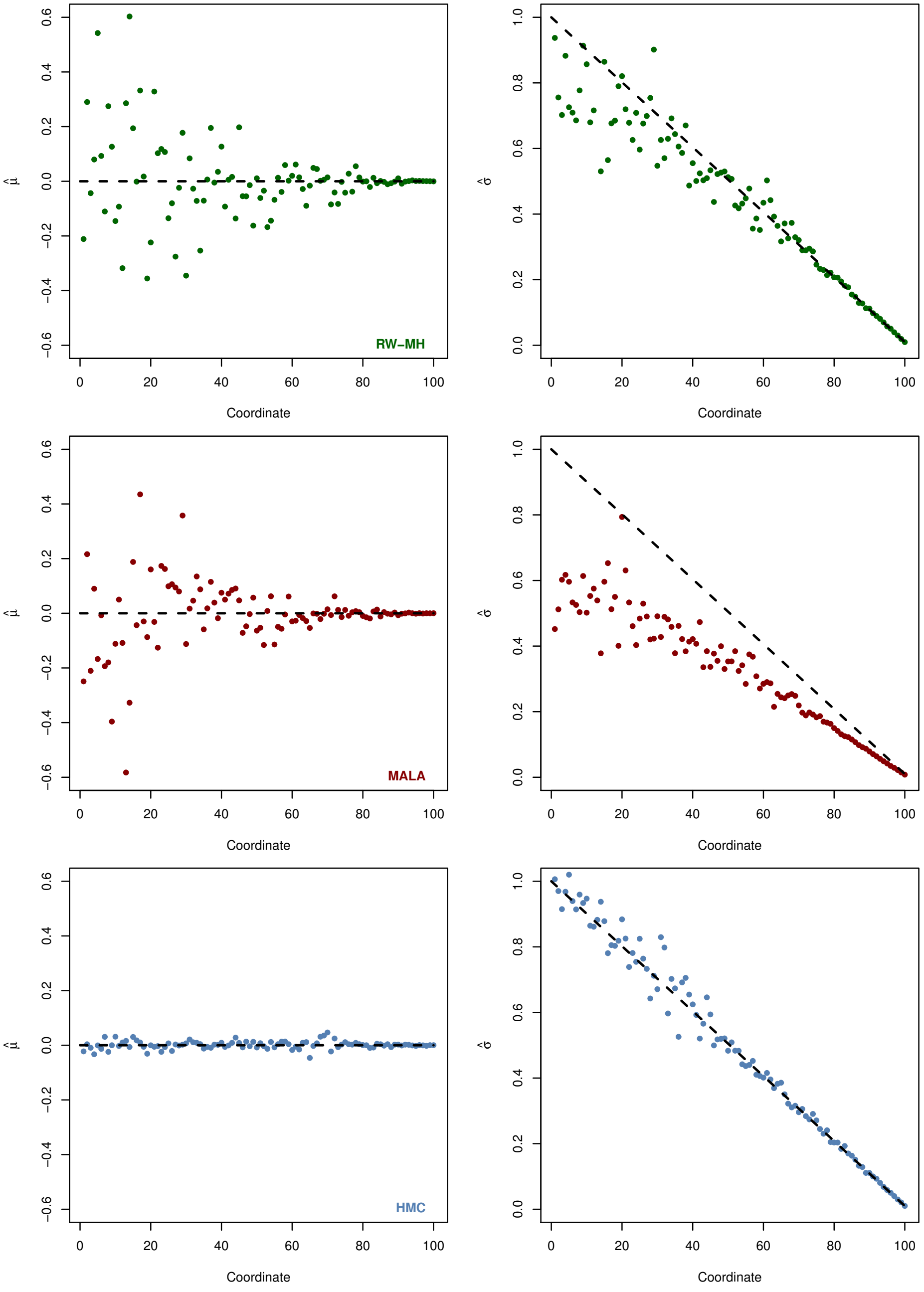

Figure 4.4: *The estimated parameters obtained for each coordinate from the RW-MH, MALA and HMC algorithms. The estimates of the mean vector (left) and the diagonal of the covariance matrix (right) are presented for each coordinate. The dashed lines indicate the true parameter values for each coordinate.*

independent samples from the target. The ACF falls off quicker for the MALA compared with the RW-MH, which indicates a more efficient exploration of the posterior. However, the estimates of the covariance matrix are biased for the MALA, which could be the result of that the Markov chain gets stuck somewhere in the parameter space. We conclude that there is a large gain in using the HMC algorithm for inference in models with a high-dimensional parameter vector.

4.3 Particle Metropolis-Hastings

As previously discussed in Chapter 2, the likelihood of an SSM is analytically intractable and therefore we cannot make use of the MH algorithm directly for parameter inference. In Section 3.3.4, we reviewed how to construct an unbiased estimator for the likelihood based on the APF. A natural solution to this problem could therefore be to replace the intractable quantities in the acceptance probability with unbiased estimates.

This idea is first used by Beaumont (2003), where the authors replace an analytical intractable likelihood with an unbiased estimate for a genetics application of the MH algorithm. The first use of this idea for parameter inference in nonlinear SSMs is found in Fernández-Villaverde and Rubio-Ramírez (2007), where the intractable likelihood is replaced with an estimate from the APF. Subsequent work by Andrieu and Roberts (2009) and Andrieu et al. (2010) analyse the resulting PMH algorithm and prove that this is a valid approach to solve the problem. To see why this method works, we review the derivation of the PMH algorithm following the presentation in Flury and Shephard (2011).

Consider the problem of using the MH algorithm to sample the parameter posterior (2.14), i.e. $\pi(\theta) = p(\theta|y_{1:T})$. This implies that the acceptance probability (4.3) depends explicitly on the intractable likelihood $p_\theta(y_{1:T})$, preventing direct application of the MH algorithm to this problem. Instead, assume that there exists an unbiased, non-negative estimator of the likelihood $\widehat{p}_\theta(y_{1:T}|u)$, i.e.

$$\mathbb{E}_{u|\theta}\big[\widehat{p}_\theta(y_{1:T}|u)\big] = \int \widehat{p}_\theta(y_{1:T}|u) m_\theta(u)\, \mathrm{d}\theta = p_\theta(y_{1:T}) \tag{4.7}$$

where $u \in \mathsf{U}$ denotes the multivariate random variable (vector) used to construct this estimator. Here, $m_\theta(u)$ denotes the probability density of u on U. When the APF is used to construct the estimator of the likelihood, the random variable u is the particles and their ancestors $\{x_{0:T}^{(i)}, a_{1:T}^{(i)}\}_{i=1}^N$.

The PMH algorithm can then be seen as a standard MH algorithm operating in a non-standard extended space $\Theta \times \mathsf{U}$, with the *extended target* given by

$$\pi(\theta, u|y_{1:T}) = \frac{\widehat{p}_\theta(y_{1:T}|u) m_\theta(u) p(\theta)}{p(y_{1:T})} = \frac{\widehat{p}_\theta(y_{1:T}|u) m_\theta(u) p(\theta|y_{1:T})}{p_\theta(y_{1:T})},$$

and the proposal distribution $m_{\theta''}(u'')q(\theta''|\theta')$. As a result, we can recover the

Algorithm 5 Particle Metropolis-Hastings (PMH) for Bayesian inference in SSMs

INPUTS: Algorithm 2, $M > 0$ (no. MCMC steps), $q(\theta'', \theta')$ (proposal) and θ_0 (initial parameters).
OUTPUT: $\theta = \{\theta_1, \ldots, \theta_M\}$ (samples from the parameter posterior).

```
Initalise using $\theta_0$.
for $k = 1$ to $M$ do
    Sample $\theta'$ from the proposal $\theta' \sim q(\theta' | \theta_{k-1})$.
    Estimate the likelihood $\widehat{p}_{\theta'}(y_{1:T})$ using Algorithm 2 and (3.19).
    Sample $\omega_k$ from $\mathcal{U}(0, 1)$.
    if $\omega_k < \alpha(\theta', \theta_{k-1})$ given by (4.8) then
        $\{\theta_k, \widehat{p}_{\theta_k}(y_{1:T})\} \leftarrow \{\theta', \widehat{p}_{\theta'}(y_{1:T})\}$. {Accept the parameter}
    else
        $\{\theta_k, \widehat{p}_{\theta_k}(y_{1:T})\} \leftarrow \{\theta_{k-1}, \widehat{p}_{\theta_{k-1}}(y_{1:T})\}$. {Reject the parameter}
    end if
end for
```

parameter posterior by marginalisation of the extended target,

$$\int \pi(\theta, u | y_{1:T}) \, \mathrm{d}u = \frac{p(\theta | y_{1:T})}{p_\theta(y_{1:T})} \underbrace{\int \widehat{p}_\theta(y_{1:T} | u) m_\theta(u) \, \mathrm{d}u}_{= p_\theta(y_{1:T})} = p(\theta | y_{1:T}),$$

using the unbiasedness property (4.7) of the likelihood. Samples from the parameter posterior can therefore be obtained as a byproduct by simulating from $\pi(\theta, u | y_{1:T})$. By selecting the proposal distribution as $q(\theta'' | \theta', u)$, the acceptance probability is given by

$$\alpha(\theta'', \theta') = 1 \wedge \frac{\widehat{p}_{\theta''}(y_{1:T} | u'')}{\widehat{p}_{\theta'}(y_{1:T} | u')} \frac{p(\theta'')}{p(\theta')} \frac{q(\theta'' | \theta', u')}{q(\theta' | \theta'', u'')}. \tag{4.8}$$

Note, that the acceptance probability is the same as for the MH algorithm, but replacing the intractable likelihood with an unbiased estimator and including an *extended proposal*. As previously discussed, the random variable u contains the entire particle system generated by the APF algorithm. From Section 3.4.2, we know that this information can be used in combination with the FL smoother to estimate the score and information matrix, which can be used to construct particle versions of the MH1 and MH2 algorithms. This is the main idea behind the proposed PMH1 and PMH2 algorithms in Paper A, to which we refer interested readers for more information.

It turns out that the acceptance rate of the PMH algorithm is closely connected with the number of particles used to estimate the log-likelihood. If N is too small, then the variance of the log-likelihood estimates are large and therefore we often get stuck with the Markov chain with a resulting low acceptance rate. We also know that N is connected with the computational cost of the APF algorithm. Therefore, we have a trade-off between the number of MCMC iterations and the number of particles in the APF, where we would like to minimise the total computational cost of the PMH algorithm. This problem is analysed and discussed by Doucet et al. (2012), Pitt et al. (2010) and Pitt et al. (2012). From this work, it is recommended to use a value of N such that the variance of the log-likelihood estimates is between 0.25 and 2.25. Consequently, we can determine the *optimal* number of particles by a pilot run.

Two other common versions of the PMH algorithm are the *particle independent MH* (PIMH) algorithm and the *particle marginal MH* (PMMH or PMH0) algorithm. The Gaussian versions of these proposals are obtained by using $q(\theta'') = \mathcal{N}(\theta''; 0, \epsilon^2)$ and $q(\theta''|\theta') = \mathcal{N}(\theta''; \theta', \epsilon^2)$, respectively. The resulting general version of the PMH algorithm that incorporates the PMH0 and PIMH as special cases is presented in Algorithm 5. The full procedure for PMH1 and PMH2 are found in Algorithm 1 in Paper A.

4.4 Example: PMH0 for parameter inference in the GARCH(1,1) model

Consider the parameter inference problem for the GARCH(1,1) model (2.3) using the NASDAQ OMX Stockholm 30 Index data from Section 2.2.2. We make use of the PMH0 algorithm and the RW proposal with the step length $\epsilon = 0.005$ for all elements in the parameter vector. We run the algorithm for $M = 20\,000$ iterations (discarding the first $5\,000$ as burn-in) with the faPF algorithm and $N = 3\,000$ particles for estimating the likelihood.

The resulting acceptance rate is 0.06 (after burn-in) with ESS $\{39, 21, 21, 11\}$ for the four parameters, respectively. Note that the poor ESS are due to that we for simplicity make use of the same step length for all parameters. In Figure 4.5, we present the trace plots and posterior estimates obtain from the run. We see that the mixing is rather poor and longer runs with smaller step lengths are needed. Also, we could make use of PMH1 or PMH2 from Paper A to improve the mixing. The posterior means are $\widehat{\theta}_{\text{PMH}} = \{0.0088, 0.14, 0.86, 0.63\}$, which can be used as point estimates of the parameters in the model.

4.4 Bayesian optimisation

BO (Jones, 2001; Boyle, 2007; Lizotte, 2008; Osborne, 2010) is a popular (global) derivative-free optimisation method, which is currently studied extensively in the machine learning community. Here, we consider the use of BO to solve

$$x_{\max} = \operatorname*{argmax}_{x \in \mathcal{B}} f(x), \tag{4.9}$$

where $\mathcal{B} \subset \mathbb{R}^d$ denotes some compact set. Here, we assume that we cannot directly evaluate the real-valued function $f(x)$, but can obtain noisy estimates (samples) modelled as

$$y_k = f(x_k) + z_k, \qquad z_k \sim \mathcal{N}(0, \sigma_z^2), \tag{4.10}$$

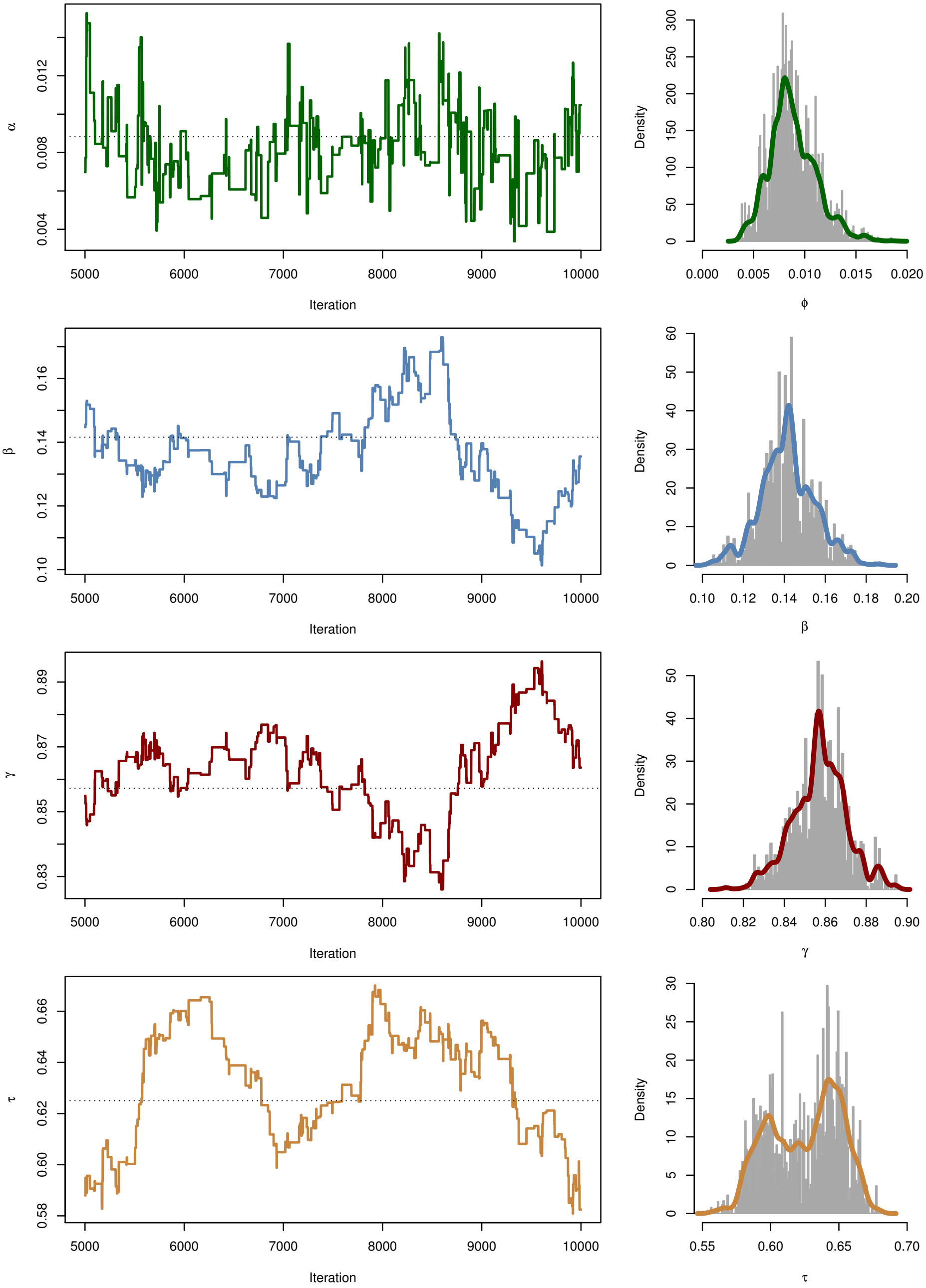

Figure 4.5: *The trace plots (left) and the corresponding posterior estimates (right) of the GARCH(1,1) model using the Nasdaq OMX Stockholm 30 Index data from Section 2.2.2. The estimates are computed using the PMH0 algorithm with* 20 000 *iterations and discarding the first* 5 000 *as burn-in.*

where z_k denotes some zero-mean Gaussian noise with unknown variance σ_z^2. Therefore, the BO algorithm is useful when we can only estimate the value of the objective function with some simulation based algorithm. It also turns out that the BO algorithm requires less estimates of the objective function than other optimisation algorithms (Brochu et al., 2010). As a consequence, BO is useful when the noisy estimates of the objective function are computationally costly to obtain. In the following, we make use of BO for ML based parameter inference and for input design in nonlinear SSMs. In these cases, the objective function corresponds to the log-likelihood and the logarithm of the determinant of the expected information matrix, respectively. In both these cases, these quantities are analytically intractable but we can obtain noisy estimates from the objective function by the use of computationally costly particle filtering and smoothing. Hence, these are two applications which fits the BO algorithm well.

The BO algorithm operates by constructing a *surrogate function* also called a *response surface* that emulates the objective function. In BO, this surrogate function is modelled as a probabilistic function with some prior form selected by the user. The name comes from that the samples from the objective function are used together with the prior to update the model using Bayes' theorem. Using the updated posterior, we can predict the value of the objective function anywhere in the space of interest and also obtain an uncertainty of the predicted value.

Using the predictive distribution, the algorithm can analyse where the optimum of the objective function could be located and focus the sampling to that area. Also, the algorithm can choose to explore areas where there are large uncertainties in the predicted value of the objective function. We refer to these two situations as *exploitation* and *exploration*, respectively. In the following, we discuss this part of the algorithm in more detail and the *acquisition rules* that is used to make the decisions regarding exploitation and exploration. To conclude this overview of the BO algorithm, we present the three steps that are carried out during the kth iteration:

(i) Sample the objective function in x_k to obtain y_k.

(ii) Use the data collected $\mathcal{D}_k = \{x_i, y_i\}_{i=1}^k$ to construct a *surrogate function.*

(iii) Use an *acquisition rule* and the surrogate function to select x_{k+1}, i.e. where to sample the objective function in the next iteration of the algorithm.

We now proceed by discussing Steps (ii) and (iii) in more detail. Step (i) depends on the specific optimisation problem that we would like to solve and for the two previously discussed applications correspond to the APF and the FL smoother in Algorithms 2 and 3, respectively.

In this thesis, we make use of GPs (Rasmussen and Williams, 2006) as the surrogate function. Therefore, we devote the following section to introducing the GP and discuss its structure and how to combine it with the obtained samples from the objective function. Then, we discuss some different acquisition functions and compare their properties. Finally, we combine the GP with the BO algorithm to obtain the *Gaussian process optimisation* (GPO) algorithm. We conclude this section by discussing some applications of GPO in connection with SSMs. For more information regarding BO, see Lizotte (2008), Boyle (2007), Brochu et al. (2010) and Snoek et al. (2012).

4.4.1 Gaussian processes as surrogate functions

GPs are an instance of *Bayesian nonparametric models* and has its origins in *kriging* (Cressie, 1993; Matheron, 1963) methods from spatial statistics. An application of kriging is to interpolate between elevation measurements sampled in some terrain to build a map of the elevation in an area. The underlying assumption is that the elevation should vary smoothly between sampled points. This is the property of the GP that we would like to use for interpolating the objective function between the values in which we sample it.

A GP (Rasmussen and Williams, 2006) can be seen as a generalisation of the multivariate Gaussian distribution to an infinite dimension. As such, it is a collection of random variables, where each finite subset is jointly distributed according to a Gaussian distribution. A realisation drawn from a GP can therefore be seen as an infinitely long vector of values, which can be seen as a function over the real space $\mathbb{R}^d$. This is why the GP is considered by some to be a *prior over functions* on $\mathbb{R}^d$.

As the GP is a Gaussian distribution of infinite dimension, we cannot characterise it using a mean vector and covariance matrix. Instead, we introduce a *mean function* $m(x)$ and a *kernel* (or covariance function) $\kappa(x, x')$ defined as

$$m(x) = \mathbb{E}[f(x)], \tag{4.11a}$$

$$\kappa(x, x') = \mathbb{E}\Big[\big(f(x) - m(x)\big)\big(f(x') - m(x')\big)\Big]. \tag{4.11b}$$

To construct the surrogate function, we assume a priori that the objective function can be modelled as a GP,

$$f(x) \sim \mathcal{GP}\big(m(x), \kappa(x, x')\big), \tag{4.12}$$

with the mean function and kernel defined by (4.11). Here, the mean function specifies the average value of the process and the kernel specifies the correlation between (nearby) samples. Both functions are considered to be prior choices to the algorithm and are used to encode the beliefs about the data before it is observed.

Consequently, we have that both the prior (4.12) and the data likelihood (4.10) are distributed according to Gaussian distributions. Hence, the posterior resulting from Bayes' theorem is a Gaussian distribution with some mean and covariance that can be calculated using standard results. From this posterior, we can construct the *predictive distribution* at some *test point* $x_\star$ given the data $\mathcal{D}_k$ by

$$f(x_\star)\big|\mathcal{D}_k \sim \mathcal{N}\Big(x_\star; \mu_f(x_\star|\mathcal{D}_k), \sigma_f^2(x_\star|\mathcal{D}_k)\Big), \tag{4.13a}$$

$$\mu_f(x_\star|\mathcal{D}_k) = \kappa_\star^\top \Big[\kappa\big(x_{1:k}, x_{1:k}\big) + \sigma_z^2 I_k\Big]^{-1} y_{1:N}, \tag{4.13b}$$

$$\sigma_f^2(x_\star|\mathcal{D}_k) = \kappa\big(x_\star, x_\star\big) - \kappa_\star^\top \Big[\kappa\big(x_{1:k}, x_{1:k}\big) + \sigma_z^2 I_k\Big]^{-1} \kappa_\star + \sigma_z^2, \tag{4.13c}$$

where $\kappa_\star = \kappa\big(x_\star, x_{1:k}\big)$ denotes the covariance between the test value and the sampling points. Here, $\kappa\big(x_{1:k}, x_{1:k}\big)$ denotes a matrix where the element at (i,j) is given by $\kappa(x_i, x_j)$ for $i = 1, \ldots, k$ and $j = 1, \ldots, k$.

To obtain the GP posterior, we need to select a kernel function. Note that, it is possible to include the assumption of a non-zero mean function into the kernel function by adding an appropriate constant kernel. Therefore, we only make use of a zero mean function in this thesis and focus on the *kernel design problem*, where several kernels can be combined by different operations to encode the prior beliefs of the structure in the data. Here, we only consider the combination of a constant covariance function and three different popular kernels: the squared exponential (SE), the Matérn 3/2 and the Matérn 5/2. See Rasmussen and Williams (2006) for other kernels and for a discussion of how they can be combined.

The SE kernel is also known as radial basis function (RBF) and has the form

$$\kappa_{\text{SE}}(x, x') = \sigma_\kappa^2 \exp\left(\frac{(x - x')^2}{2l^2}\right), \tag{4.14}$$

where the hyperparameters are $\alpha = \{\sigma_\kappa^2, l\}$. Here, l is called the *characteristic length scale* as it scales the Euclidean distance between the two points x and x'. Two other kernels are the Matérn 3/2 and the Matérn 5/2 with the form

$$\kappa_{3/2} = \sigma_\kappa^2 \bigg(1 + \frac{\sqrt{3}(x - x')}{l}\bigg) \exp\bigg(- \frac{\sqrt{3}(x - x')}{l}\bigg), \tag{4.15a}$$

$$\kappa_{5/2} = \sigma_\kappa^2 \bigg(1 + \frac{\sqrt{5}(x - x')}{l} + \frac{5(x - x')^2}{3l^2}\bigg) \exp\bigg(- \frac{\sqrt{5}(x - x')}{l}\bigg), \tag{4.15b}$$

where the hyperparameters are $\alpha = \{\sigma_\kappa^2, l\}$. The main difference between the three kernels are their smoothness properties. The SE kernel is the smoothest and it has infinitely many continuous derivatives. The $\kappa_{3/2}$ and $\kappa_{5/2}$ kernels only have one and two continuous derivatives, respectively. To illustrate the different kernels, we present some simulated realisations from each in Example 4.5.

4.5 Example: GP kernels

In Figure 4.6, we present realisations from the GP prior (4.12) using three different kernels with two different length scales. We see that the smoothness of the prior decreases from top to bottom and this verifies the previous discussion. Also, we see that the length scale determines the rate of change in the realisations.

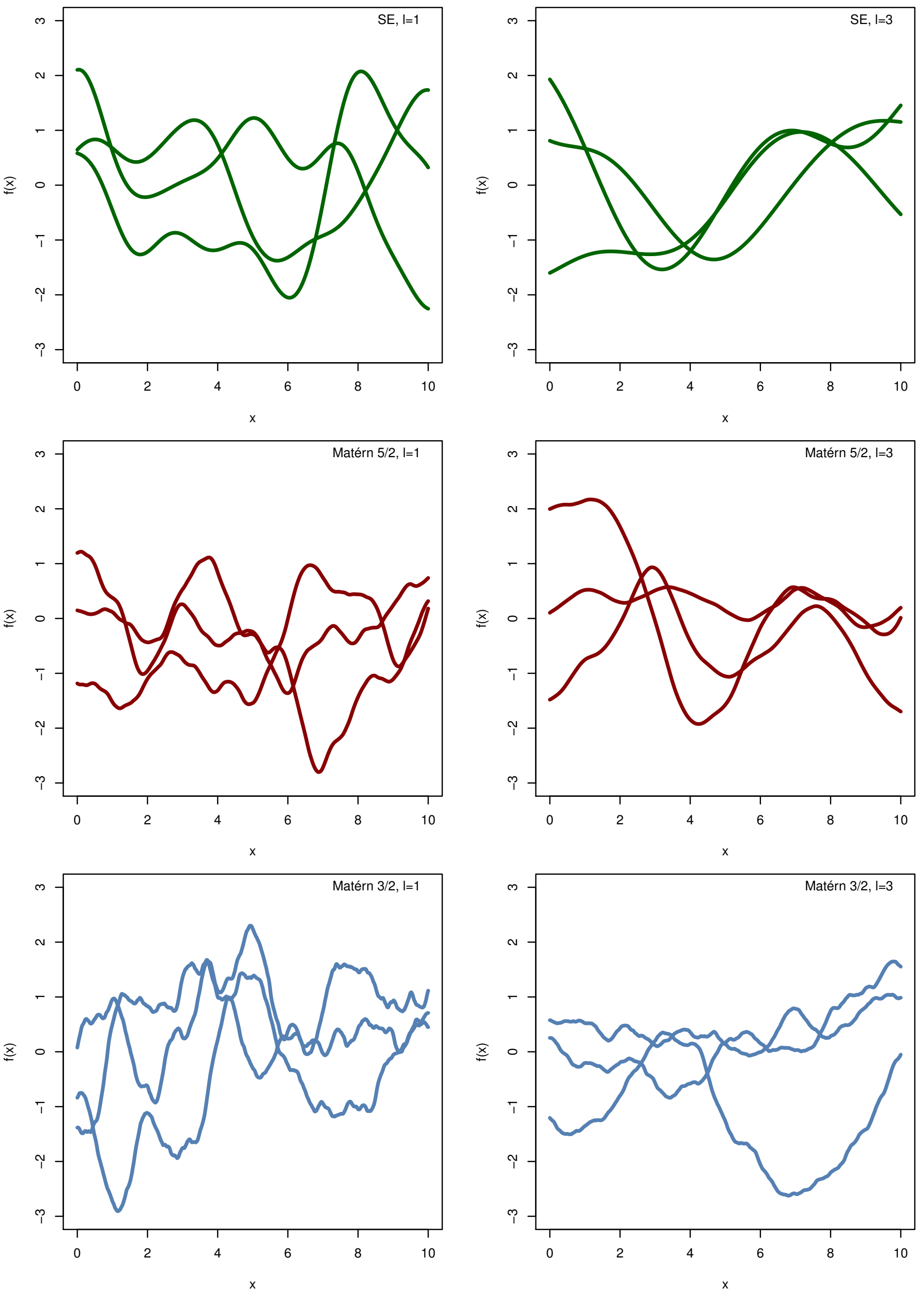

Figure 4.6: *Realisations simulated from the GP prior using three different kernels: SE (upper), Matérn 5/2 (middle) and Matérn 3/2 (lower) and length scales 1 (left) and 3 (right).*

Finally, we need to determine suitable values for the hyperparameters in the kernel. This can be done using an ML procedure called *emperical Bayes* (EB), where the marginal likelihood of the data is optimised with respect to the hyperparameters α. Note that this is not a pure Bayesian approach as the data is used to determine the properties of the prior. Nevertheless, it is a popular approach in the GP literature and therefore we make use of it here. The marginal likelihood can be computed by a marginalisation,

$$p(y|x,\alpha) = \int p(y|f,x,\alpha)p(f|x,\alpha)\,\mathrm{d}f,$$

where we drop the subscript on $y_{1:k}$, $x_{1:k}$ and $f_{1:k}$ for brevity. The *log-marginal likelihood* can be obtained in closed form using results for the Gaussian distribution as

$$\log p(y|x,\alpha) \propto -\frac{1}{2}y^\top \Big[\kappa\big(x,x\big) + \sigma_z^2 I_N\Big]^{-1} y - \frac{1}{2}\log\Big|\kappa\big(x,x\big) + \sigma_z^2 I_N\Big|,$$

where we have neglected the terms that are independent of α (independent of the kernel). The gradient of the log-marginal likelihood for α_j can be computed using

$$\frac{\partial}{\partial \alpha_j}\log p(y|x,\alpha) = \frac{1}{2}\mathsf{tr}\bigg(\Big(\beta\beta^\top - \kappa\big(x,x\big)^{-1}\Big)\frac{\partial}{\partial \theta_j}\kappa\big(x,x\big)\bigg), \qquad \beta = \kappa\big(x,x\big)^{-1}y.$$

Therefore, the hyperparameters can be estimated by maximising the log-marginal likelihood using a gradient-based search algorithm (4.1). In Example 4.6, we make use of the EB method for GP regression using the different kernels discussed in this section.

4.6 Example: GP regression

Consider the GP regression problem where we would like to recover the underlying function $f(x)$ given by

$$f(x) = \Big[3\cos(0.5x) + \sin(2 + 0.25x)\Big]^2,$$

from the noisy measurements y generated by (4.10) with $\sigma_z^2 = 2$. Here, we make use of the three kernels in (4.14) and (4.15) with an added constant kernel to account for the non-zero mean of the data. The hyperparameters of the resulting kernels are estimated using the EB procedure.

In the left part of Figure 4.7, we use $N = 5$ observations and present the resulting predictive mean (solid line), the underlying function $f(x)$ (dashed) and the 95% CI of the predictive distribution (gray area). We see that most of the samples are located in the left part of the region and as a result the predictive means of the GPs follows the underlying function.

In the right part of Figure 4.7, we present the same setup but using $N = 15$ sampled points instead. Here, the overall fit is much better and the three GP predictive distributions recover the underlying function pretty well across the region. As the underlying function is smooth (have infinity many continuous derivatives), it is well captured by the SE kernel.

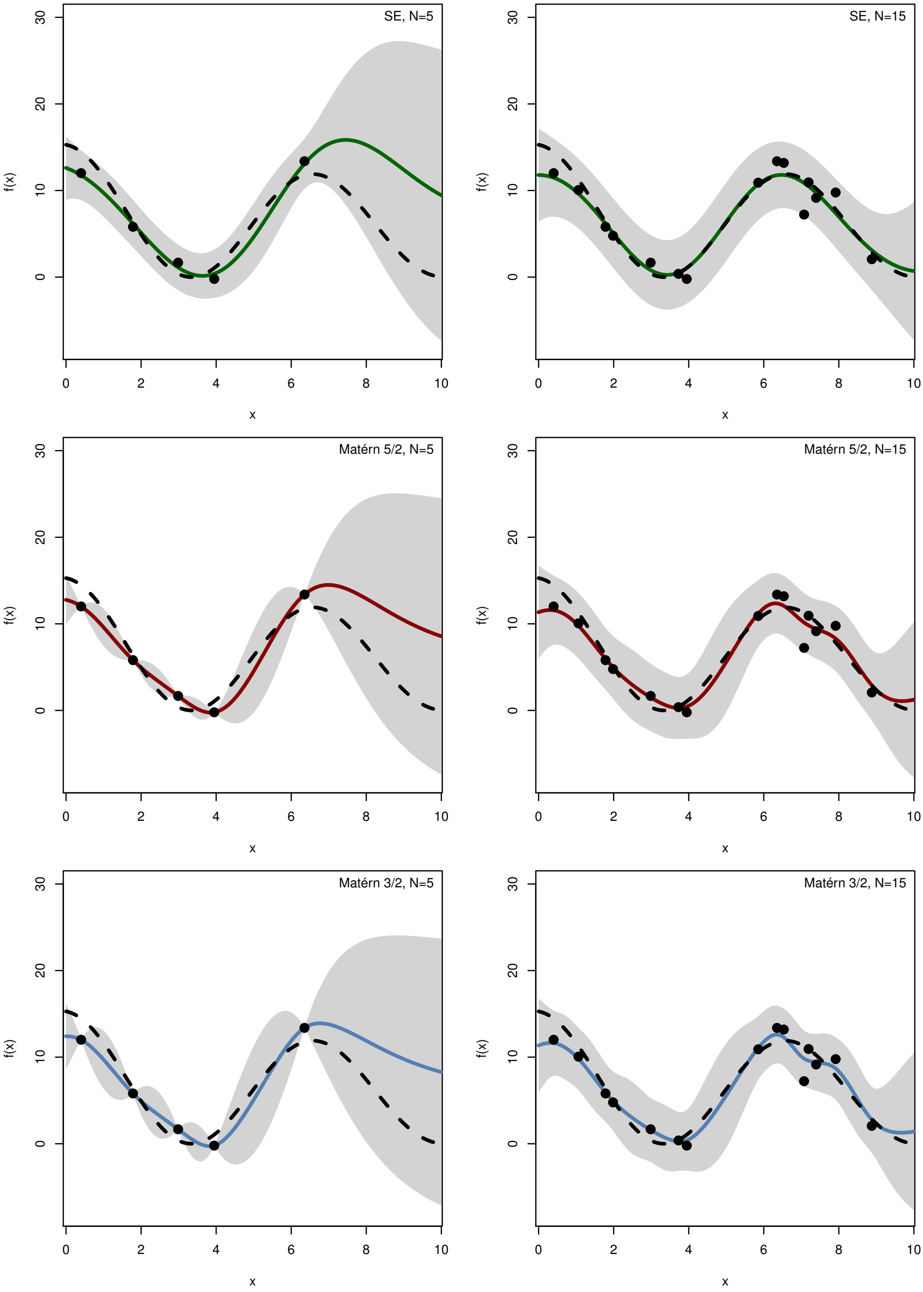

Figure 4.7: *The GP regression problem in Example 4.6 using $N = 5$ (left) and $N = 15$ (right) data points with three different kernels. The mean of the predictive distributions (solid lines) and the corresponding 95% CI (gray area) are presented together with the true function (dashed line) and the noisy observations (black dots).*

4.4.2 Acquisition rules

We now proceed by discussing Step (iii) of the BO algorithm, i.e. the acquisition rule and how it operates. The main idea with this rule is to make use of the predictive mean and its uncertainty to decide on a good parameter to sample the objective in during the next iteration. We would like the algorithm to explore the parameter space to find all the peaks, but still focus the samples around the maximum to decrease the number of samples required to solve the problem. This is called the *exploration and exploitation trade-off* as we would like to explore the space, but also exploit the information encoded in the surrogate function about about where the maxima are located.

Using a general acquisition rule $\mathsf{AQ}(x|\mathcal{D}_k)$, we would like to select x_{k+1} in Step (iii) as the maximising argument,

$$x_{k+1} = \operatorname*{argmax}_{x \in \mathcal{B}} \mathsf{AQ}(x|\mathcal{D}_k), \tag{4.16}$$

which in itself is an optimisation problem. We discuss some different methods for solving this in Paper B and now instead proceed with discussing three different acquisition rules that are popular and widely used in GPO (Brochu et al., 2010). These are: *the probability of improvement* (PI), *the expected improvement* (EI) and *the upper confidence bound* (UCB).

The PI and EI make use of the fact that the predictive distribution is Gaussian and that we can use the predictive mean and covariance. From this, we can use the probability density function (PDF) and the cumulative distribution function (CDF) of the Gaussian distribution to calculate the PI and EI. This can be done by introducing the highest predicted value of the surrogate function (or the incumbent),

$$\mu_{\max} = \max_{x_i \in x_{1:k}} \mu_f(x_i|\mathcal{D}_k), \tag{4.17}$$

then the PI (Kushner, 1964) can be computed using

$$\mathsf{PI}(x|\mathcal{D}_k) = \mathbb{P}\big(\mu_f(x|\mathcal{D}_k) \geq \mu_{\max} + \xi\big) = \Phi(Z_k), \tag{4.18a}$$

$$Z_k = \frac{\mu_f(x|\mathcal{D}_k) - \mu_{\max} - \xi}{\sigma_f(x|\mathcal{D}_k)}, \tag{4.18b}$$

where $\Phi(\,\cdot\,)$ denotes the Gaussian CDF. Here, $\xi \geq 0$ denotes a user-defined coefficient proposed in Lizotte (2008) to balance the exploitation and exploration behaviour. From the form of the PI expression, we note that the variable Z_k can be seen as a standard Gaussian random variable. Hence, Z_k assumes a (large) positive value if the predictive mean is close to or larger than $\mu_{\max}$. Therefore, we obtain a value of the PI close to one and it is probable that the GPO algorithm will sample the objective function in this region during its next iteration. Instead, we obtain a small value of the PI if the predictive mean is much smaller than $\mu_{\max}$ and/or the uncertainty is very large. A small value of the PI results in that it is unlikely that the GPO algorithm will sample the objective function in this region during its next iteration.

However, the PI only takes into account the probability of an improvement and not its size. To include this information, we consider the EI rule (Mockus et al., 1978; Jones et al., 1998) of the form

$$\mathsf{EI}(x|\mathcal{D}_k) = \big(\mu_f(x|\mathcal{D}_k) - \mu_{\max} - \xi\big)\Phi(Z_k) + \sigma_f(x|\mathcal{D}_k)\phi(Z_k), \tag{4.19}$$

where $\phi(\,\cdot\,)$ denotes the Gaussian PDF. The interpretation of the EI follows in analogue with the PI rule. If the predictive mean is close to or larger than $\mu_{\max}$ then $\Phi(Z_k)$ assumes a value close to one, which scales the expected gain in the objective function. Here, we also take the uncertainty into account by the second term in (4.19). This term can be seen as a scaling of the uncertainty in the predictive distribution. The scaling is large if Z_k is close to zero and decreases for larger value. This means that we get an extra contribution to the EI if the predictive mean is close to $\mu_{\max}$, which means that it could be interesting to explore that area. For more information, see Paper B where we review the derivation of the EI rule.

The third acquisition rule, the UCB rule follows from that we can construct a CI using the predictive distribution. The intuition for this is that if the predictive mean is high in an area of the parameter space, the resulting UCB is also large. Moreover, uncertainty in a region increases the predictive covariance, which also increases the value of the UCB. By this rule, we therefore explore areas where peaks have been found and where the uncertainty is large. As the name suggests, we are interested in the upper bound which for a Gaussian distribution is given by

$$\mathsf{UCB}(x|\mathcal{D}_k) = \mu_f(x|\mathcal{D}_k) + \epsilon\sigma_f(x|\mathcal{D}_k). \tag{4.20}$$

Here, $\epsilon \geq 0$ denotes a coefficient determining the confidence level of the interval. As the predictive distribution is Gaussian, we would choose $\epsilon = 1.96$ to obtain a 95% CI and $\epsilon = 2.58$ to obtain a 99% CI.

4.7 Example: GPO using different acquisition rules

Consider again the problem in Example 4.6 using the Matérn 5/2 kernel with $N = 3$, $N = 5$ and $N = 15$ samples. In the left of Figure 4.8, we present the predictive distributions together with the underlying function as before. In the right part of the figure, we present the normalised value of the three different acquisition functions previously discussed: the PI (green), the EI (red) and the UCB (blue). Here, we use the recommended value of $\xi = 0.01$ in Lizotte (2008) and $\epsilon = 1.96$, resulting in that the UCB corresponds to a 95% CI.

We note that the three acquisition functions have quite different behaviours in the three situations. In the first situation (upper), the PI and the UCB have two peaks that are located in the left and right part of the region to exploit the current information and to explore the region better, respectively. The EI would like to sample the left end of the region to exploit the current information and to reduce the uncertainty in that area.

In the second situation (middle), the EI again would like to exploit the current information by placing a sample near the peak of the predictive mean. The other two acquisition functions would like to explore the right part of the region better.

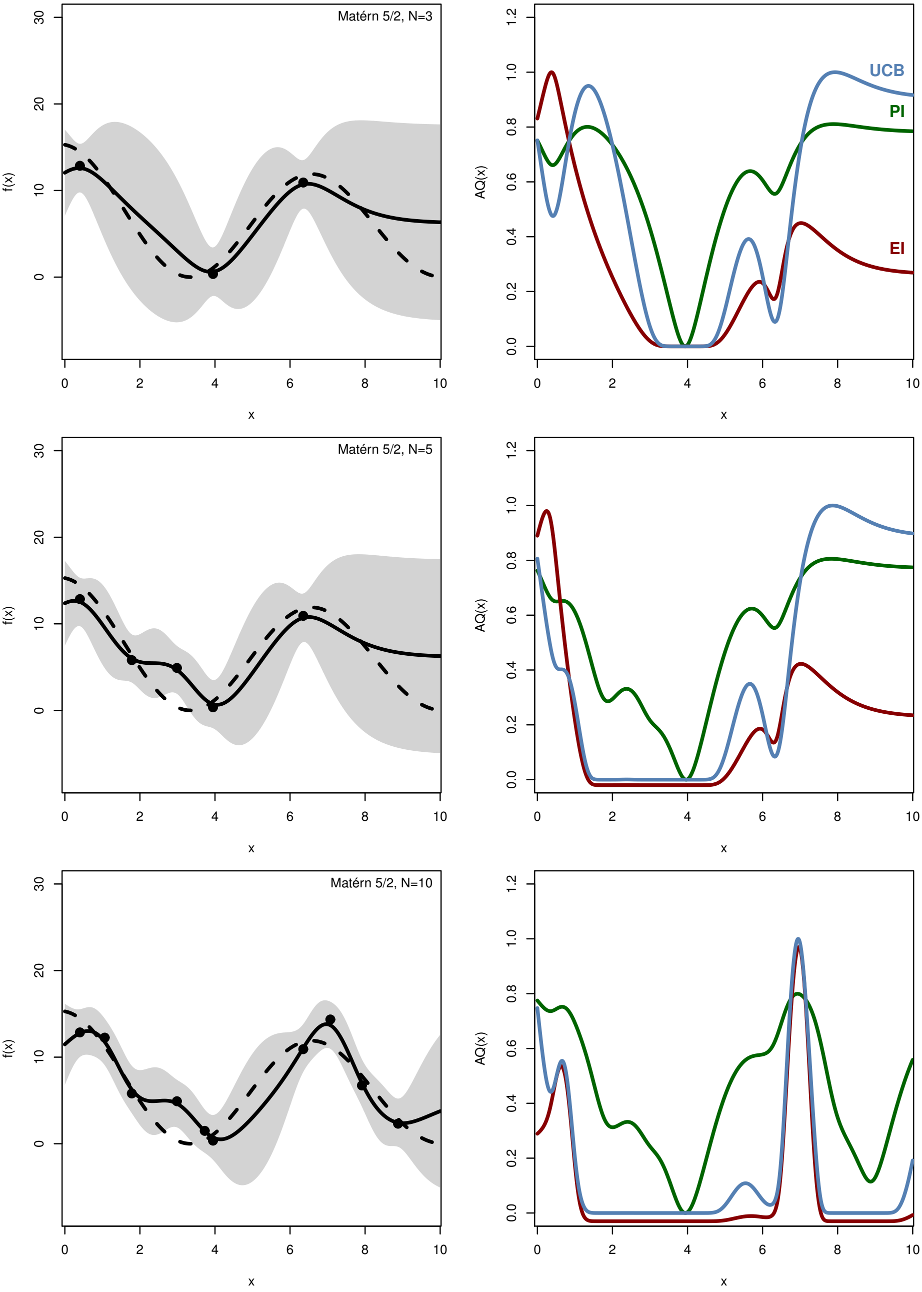

Figure 4.8: *The GP regression problem in Example 4.6 using $N = 3$ (upper), $N = 5$ (middle) and $N = 10$ (lower) data points with the Matérn 5/2 kernel. In the left part, we present the mean of the predictive distributions (solid lines) and the corresponding 95% CI (gray area) together with the true function (dashed line) and the noisy observations (black dots). In the right part, we present normalised values of three different acquisition functions for each situation.*

Finally, in the third situation (lower) the three functions agree and would like to exploit the current peak in the predictive mean. Hence, we conclude that the three acquisition functions have rather different behaviour and we return in a following example to investigate how this affects the resulting GPO algorithm.

4.4.3 Gaussian process optimisation

The full procedure for using GPO to estimate the solution to (4.9) is presented in Algorithm 6. As previously discussed, the user choices include the kernel for the GP prior and the acquisition function. Again, we remind the reader that the choice of kernel is crucial for the performance of the algorithm. Furthermore, an optimisation method is needed to optimise the acquisition function in (4.16). Here, we make use of a global derivative-free optimisation algorithm called DIRECT (Jones et al., 1993), but we discuss other possible choices in Paper B.

The GPO algorithm is often initialised by some randomly selected parameters sampled from the parameter prior. These samples are used to estimate the hyperparameters of the GP kernels so that the AQ function can operate. The number of these samples varies with the dimension of the problem but between 5 and 50 are reasonable numbers for small problems. Also, as the EB procedure is computationally costly, it is beneficial to estimate the GP hyperparameters every 5th or 10th iteration to save computations.

We end this section by discussing three different examples of where GPO and/or surrogate function modelling is useful. In Examples 4.8 and 4.9, we use the GPO algorithm for solving the ML parameter inference problem in an SSM and compare the different possible choices of AQs. Note that more examples of this application are found in Papers B and C.

Finally in Example 4.10, we illustrate the use of GPO for creating an input that maximises the logarithm of determinant of the expected information in an SSM. Remember that this corresponds to maximising the accuracy of the ML parameter estimate, which is the objective in input design.

Algorithm 6 Gaussian process optimisation (GPO)

INPUTS: $\kappa(\,\cdot\,)$ (GP kernel), AQ (acquisition func.), K (no. iterations) and x_1 (init. value).
OUTPUT: $\widehat{x}_{\max}$ (estimate of the maximising argument of (4.9)).

```
Initialise the algorithm by random sampling.
for $k = 1$ to $K$ do
    Sample $f(x_k)$ to obtain the noisy estimate $y_k$.
    Compute (4.13) to obtain $\mu_f(x|\mathcal{D}_k)$.
    Compute (4.17) to obtain $\mu_{\max}$.
    Compute (4.16) to obtain $x_{k+1}$.
end for
Compute the maximiser of $\mu_f(x|\mathcal{D}_K)$ to obtain the estimate $\widehat{x}_{\max}$ of (4.9).
```

4.8 Example: GPO for ML inference in the GARCH(1,1) model

Consider the ML parameter inference problem in the GARCH(1,1) model (2.3) using synthetic data with $\theta = \tau$ and $\Theta = (0, 0.25) \in \mathbb{R}$. Here, we make use of the GPO algorithm for solving this problem, by first rewriting (4.10) as

$$\widehat{\ell}(\theta_k) = \ell(\theta_k) + z_k, \qquad z_k \sim \mathcal{N}(0, \sigma_z^2),$$

which is similar to the CLT established in Section 3.3.4. As a consequence, (4.9) turns into the maximum likelihood maximisation problem discussed in Section 2.3. The resulting procedure follows from Algorithm 6 by plugging in the APF from Algorithm 2 for log-likelihood estimation.

Here, we use an one dimensional parameter vector to be able to compare the three different AQs discussed in the previous section. We generate $T = 250$ observations from the model using $\{\alpha, \beta, \gamma, \tau\} = \{0.1, 0.8, 0.05, 0.3\}$. We make use of the faPF with $N = 100$ to estimate the log-likelihood during each iteration. Here, we use the recommended value of $\xi = 0.01$ in Lizotte (2008) and $\epsilon = 1.96$ for the acquisition rule.

In Figure 4.9, we present the procedure at five consecutive iterations using three different acquisition rules. The procedure is initialised with two randomly selected samples of the log-likelihood, which are not shown in the figure. Here, we see that the behaviour of the three acquisition rules are rather different but the resulting mode of the log-likelihood is almost the same. The parameter estimate is obtained around $\widehat{\theta}_{\mathrm{ML}} = 0.12$ for all three choices of the acquisition rule. Note, that this small toy example is probably not complex enough to show the real differences between the acquisition rules. Remember, that more extensive evaluations by Lizotte (2008) show that the EI rule is often a good choice in many different applications.

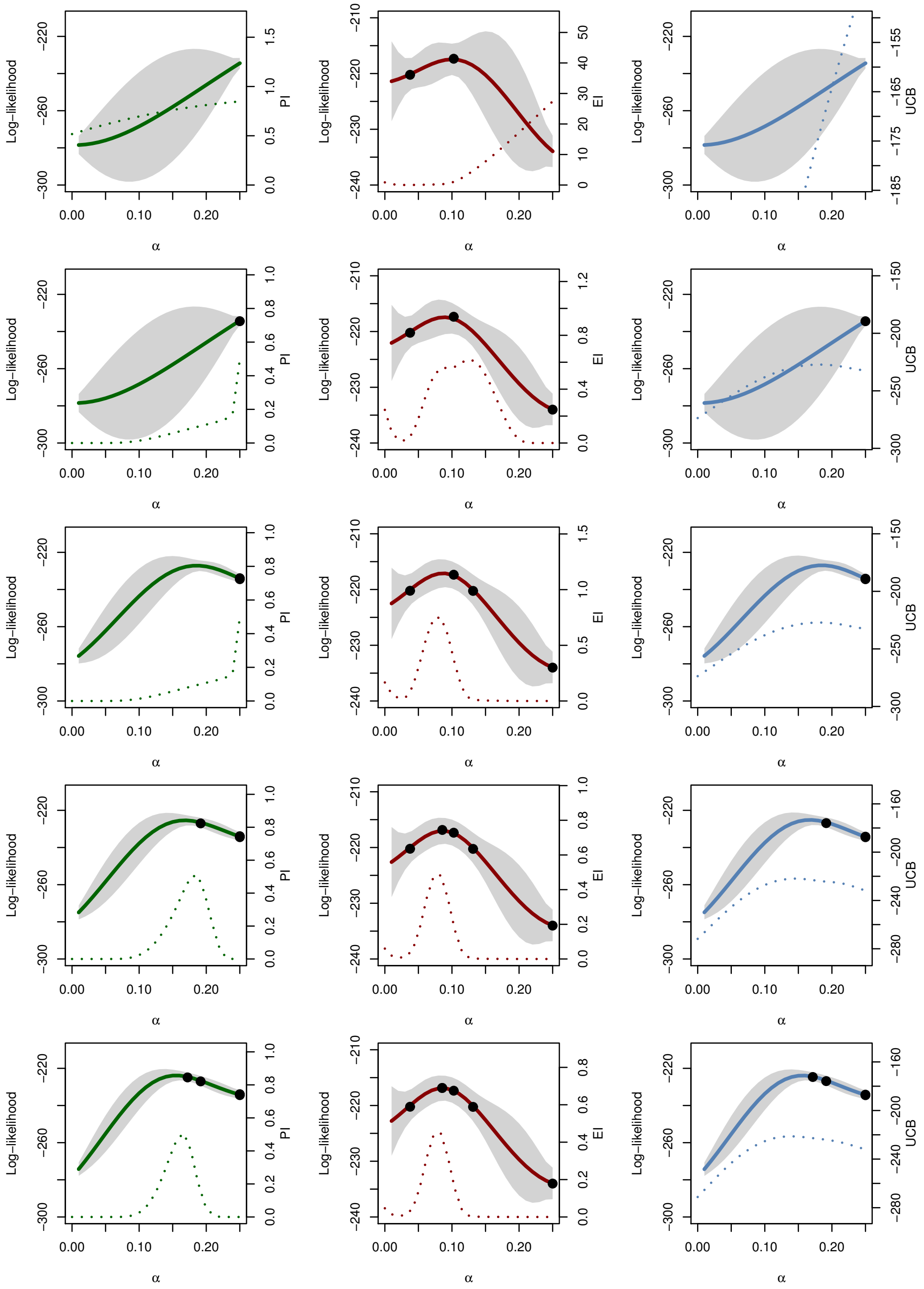

Figure 4.9: *Five steps of the GPO algorithm for ML parameter inference in Example 4.8 using three different acquisition rules: PI (left), EI (center), UCB (right). The predictive mean and the resulting value of the acquisition rule are presented with coloured solid and dotted lines, respectively. The black dots and gray areas indicate the samples obtained from the log-likelihood and the 95% predictive CI, respectively.*

4.9 Example: GPO for ML inference in the earthquake count model

We return to the setting considered in Example 4.9 for ML parameter inference in the earthquake count model (2.6) using the real-world data discussed in Section 2.2.3. Here, the parameter vector is given by $\theta = \{\phi, \sigma_v, \beta\}$ and the parameter space is $\Theta = (0,1) \times (0,1) \times (10,20) \in \mathbb{R}^3$. We make use of the bPF with $N = 1\,000$ particles to estimate the log-likelihood. The procedure is initialised with 50 samples from the log-likelihood at randomly selected parameters and continues with 150 iterations of the GPO algorithm.

In Figure 4.10, we present the current ML parameter estimate at each iteration of the GPO algorithm. We see that the parameter estimates and the predicted log-likelihood stabilise after about 150 iterations. This is rather fast compared with e.g. SPSA which in Paper B requires an order of magnitude more samples of the log-likelihood. The final parameter estimate is obtained as $\widehat{\theta}_{\text{ML}} = \{0.88, 0.15, 17.65\}$. This shows that the underlying intensity is rather slowly varying and that the mean number of major earthquakes each year is 17.65.

4.10 Example: Input design in the LGSS model using GPO

Consider the LGSS model (2.2) with the parameters $\theta^\star = \{0.5, 1, 0.1, 0.1\}$, $T = 250$ and an input $u_{1:T}$ generated by

$$u_t = \begin{cases} -1, & \text{with probability } (1-\alpha_1)(1-\alpha_2), \\ 0, & \text{with probability } \alpha_1, \\ 1, & \text{with probability } (1-\alpha_1)\alpha_2, \end{cases}$$

for $t = 1, \ldots, T$. Here, the aim is to find the input parameters $\alpha^\star = \{\alpha_1^\star, \alpha_2^\star\}$ such that

$$\alpha^\star = \operatorname*{argmax}_{\alpha \in [0,1] \times [0,1]} \log \mathsf{det}\Big(\widehat{\mathcal{I}}(\theta^\star, u_{1:T}(\alpha))\Big),$$

where $\widehat{\mathcal{I}}(\theta^\star, u_{1:T}(\alpha))$ denotes the estimated expected information matrix using the input $u_{1:T}(\alpha)$ generated using the input parameters α. We make use of the second formulation in (2.13) to estimate the expected information matrix. This is done by first estimating the score function with the FL smoother in Algorithm 3 for $M = 100$ different data realisations from the model using the same input $u_{1:T}$ and the faPF with $N = 100$. Finally, the expected information matrix is estimated using the sample covariance matrix of the score function estimates.

We integrate this problem into the GPO procedure outlined in Algorithm 6, where the estimation of the expected information matrix is Step (i). Furthermore, we make use of the EI as the acquisition rule and initialise the procedure with 10 uniform random samples for α. In Figure 4.11, we present the resulting predictive mean for the input parameters and the sampling points for the algorithm. The input parameter estimates are obtained as $\widehat{\alpha} = \{0.26, 0.50\}$. We see that the algorithm converges quickly to the estimates, which this shows that the GPO algorithm can be a useful alternative in input design. This as, it requires a limited number of computationally costly estimates of the expected information matrix.

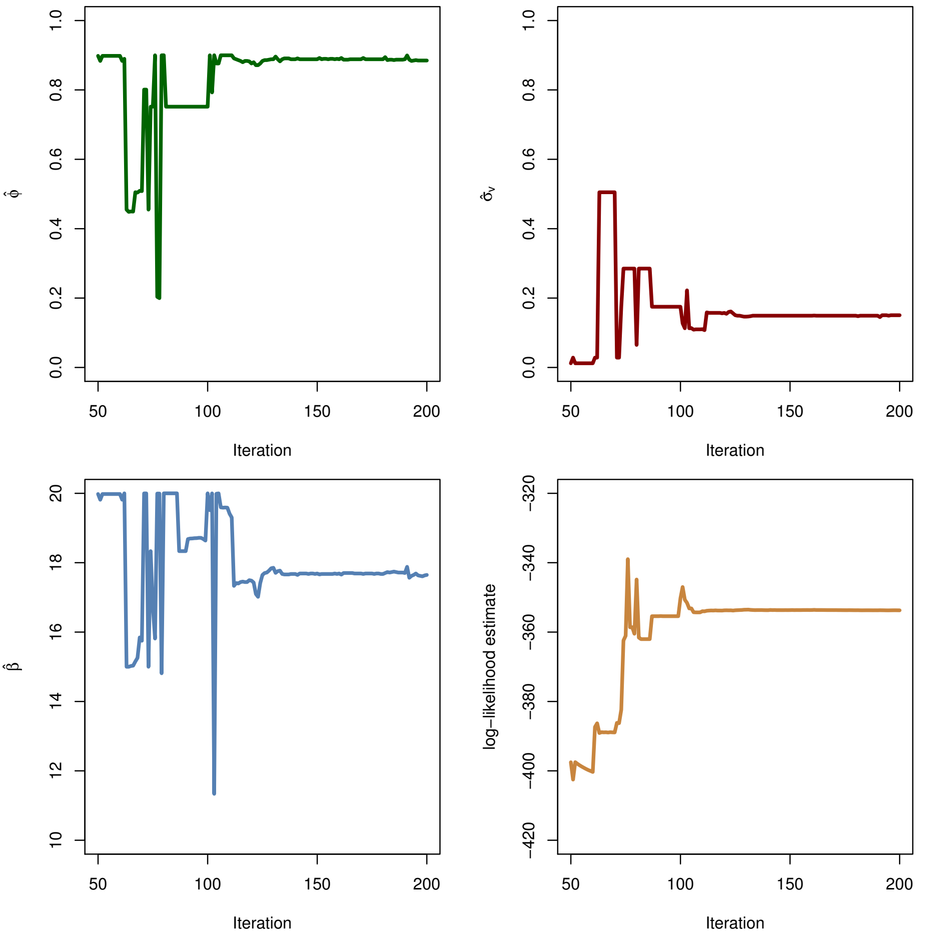

Figure 4.10: *The ML parameter estimate and the resulting predicted log-likelihood at each iteration of the GPO algorithm in Example 4.9.*

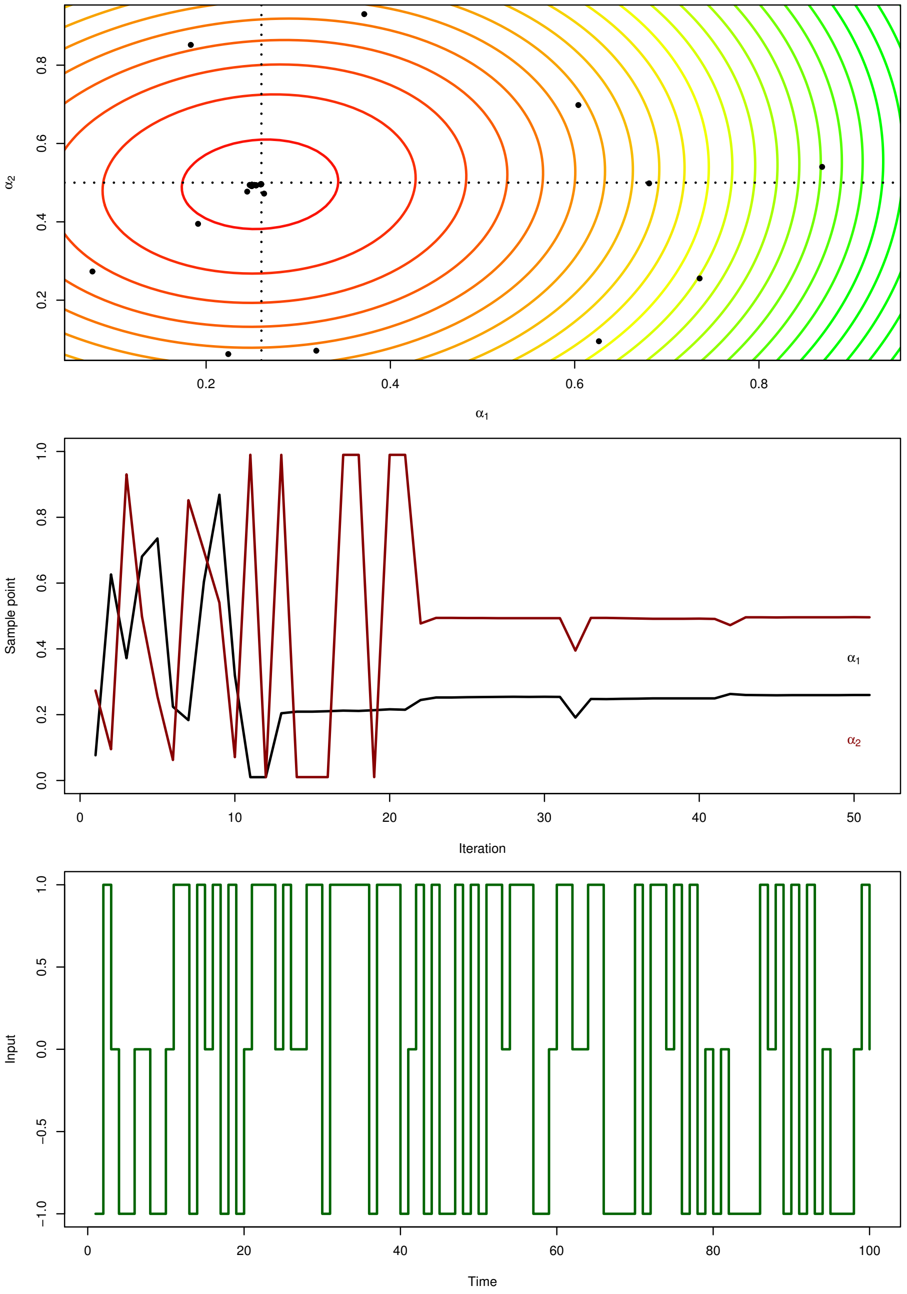

Figure 4.11: *Upper: the estimated mean of the expected information matrix as a function of the input parameters. Middle: the sampling points of the GPO algorithm. Lower: a realisation of the estimated optimal input.*

5 Concluding remarks and future work

In this chapter, we give a summary of the contributions in the papers included in this thesis and discuss some avenues for future work.

5.1 Summary of the contributions

Broadly speaking, the main contributions of this thesis are mainly within two areas. The first contribution is to *develop new methods and improve existing methods for efficient parameter inference in SSMs.* Here, we are concerned with computational efficiency, which means that we would like to reduce the number of particles or iterations needed to reach a certain accuracy in the parameter estimates. This contribution is contained within Papers A-C. The second contribution is to make use of SMC and MCMC to *extend existing methods for parameter inference and input design to nonlinear problems.* This contribution is discussed in Papers D and E.

In Paper A, we develop a novel PMCMC algorithm that combines the PMH algorithm from Section 4.3 with the Langevin dynamics discussed in Section 4.2.2. The key idea here is to include the particle system within the proposal of the PMH algorithm. With this information, we can make use of the FL smoother from Section 3.4.2 to estimate the score function and the observed information matrix. These quantities are then used to construct the PMH1 and PMH2 algorithms in Paper A in analogue with the MH1 and MH2 algorithms discussed in Section 4.2.2. The resulting algorithm is efficient as it explores the posterior distribution better and this results in a higher ESS compared with the PMH0 algorithm. Furthermore, the added information makes the algorithm invariant to affine transformations of the parameter vector and reduces the length of the the burn-in. As a consequence, the proposed algorithm require less iterations than the PMH0 algorithm in some settings, which makes it more computationally efficient as the computational complexity of the two algorithm are the same.

In Paper B, we develop a novel algorithm for ML parameter inference by combining ideas from GPO in Section 4.4 with log-likelihood estimation using the APF from Section 3.3.4. The resulting algorithm is computationally efficient as it requires less samples from the log-likelihood compared with some other optimisation methods. As these estimates are computationally costly to obtain this results in an overall decreased computational cost. Compared with SPSA, the gain is about one order of magnitude, see Paper B for a comparison on the HWSV model. In Paper C, we extend the combination of GPO and SMC to parameter inference in nonlinear SSMs with intractable likelihoods. Computationally costly ABC methods are used to approximate the intractable likelihood. Therefore, there could be substantial gains in using this algorithm for inference in this type of models.

In Paper D, we develop a novel algorithm for input design in nonlinear SSMs, which can handle amplitude constraints on the input. The proposed method combines results from Valenzuela et al. (2013) with SMC from Section 3.4.2 for estimating the expected information matrix. In Paper E, we propose two algorithms for parameter inference in ARX models with Student-t innovations which includes automatic model order selection by two different methods. These methods makes use of the MH algorithm discussed in Section 4.2 with reversible jump and the Gibbs sampler together with sparseness priors.

5.2 Outlook and future work

In this section, we summarise some ideas for future work and extensions of the contributions presented within this thesis. We discuss three different areas; the PMH-algorithm, the GPO-SMC algorithm and input design in SSMs.

5.2.1 Particle Metropolis-Hastings

The proposed contributions to the PMH algorithm are mainly methodological developments of existing methods. Therefore, it would be interesting to examine the theoretical properties of the PMH1 and the PMH2 algorithms. This includes questions regarding the convergence rate of the algorithm and how its properties scale with the dimension of the parameter space. Similar analysis has previously been done for MH0 (Roberts et al., 1997), MH1 (Roberts and Rosenthal, 1998) and PMH0 (Sherlock et al., 2013). A possible first step for the PMH analysis is to consider the situation where the number of observations is large. By the discussion in Section 2.4, we know that the Bayesian CLT would give a roughly Gaussian posterior, which simplifies the analysis.

Further methodological developments could also be interesting in the PMCMC framework. This includes the development of adaptive PMH1 and PMH2, which automatically determines suitable step sizes and the number of particles. It could also be possible to relax the reversibility constraint of the Markov chain during the burn-in phase of the algorithm. This could decrease the hitting time of the posterior mode by the Markov chain. The adaptive mechanism could then decide when the chain has reached the mode and then turn on the reversibility condition, so that the chain admits the target as its stationary distribution. Relevant work for this idea is found in Andrieu and Thoms (2008), Peters et al. (2010) and Pitt et al. (2012). Another approach to reduce the length of the burn-in is to make use of GPO to estimate the location of the posterior mode in a pilot run. This is similar to the work by Owen et al. (2014), where the authors make use of some pilot runs of the ABC algorithm to initialise the PMH0 algorithm in SSMs with intractable likelihoods.

Also, it would be interesting to develop a particle HMC algorithm as suggested in the discussions (Doucet et al., 2011) following Girolami and Calderhead (2011). The challenge with this idea is how to handle that multiple APFs are run within each PMH iteration, which might require some additional developments to the PMCMC framework. Better particle smoothers could also be useful as they could improve the estimates of the score function and the information matrix. This results in that larger step lengths can be used in the PMH2 algorithm and could lead to even larger increases in mixing of the Markov chain.

Online methods for Bayesian inference would also be of great interest, especially for the many big data problems that are likely to be faced in the future. Also, graphical processing units (GPUs) and other multicore architectures can be used to decrease the computational cost as some parts of the SMC and MCMC algorithms are possible to run in parallel. Interested readers are referred to Beam et al. (2014), Neiswanger et al. (2013), Henriksen et al. (2012) and Murray (2012) for more information.

5.2.2 Gaussian process optimisation using the particle filter

As we have demonstrated in this thesis, the performance of the GPO algorithm depends on the kernel function in the GP prior and the choice of acquisition function. Therefore it would be interesting to develop new acquisition functions that could make use of ideas from sparse GPs, Newton methods and/or proximal point algorithms (Rasmussen and Williams, 2006; Nocedal and Wright, 2006). The main challenge is to construct a rule that keeps exploring the objective function, but still keeps the fast convergence that we have illustrated in the examples in Section 4.4.3 and in Papers B and C.

Another possible improvement to the algorithm is to remove the bias in the log-likelihood estimate. This could be done by the bias compensation discussed in Example 3.4. Also, it would be interesting to develop online methods for this algorithm, perhaps by using some kind of stochastic approximation scheme. Finally, we think that there are many interesting applications of GP models for estimating the score and information matrix for an SSM. As these are computationally costly to evaluate with good accuracy, perhaps the ideas from *probabilistic numerics* could be helpful. This is an emerging field in machine learning, where GPs also are used to estimate derivatives and integrals as well as to solve ordinary differential equations. For more information, see Hennig (2013), Osborne et al. (2012), Osborne (2010) and Boyle (2007).

5.2.3 Input design in SSMs

The main drawback of the input design method proposed in Paper D is that the expected information matrix is computationally costly to evaluate. It is possible to make a perfect parallel implementation of this method to reduce the computational cost. However, it would be even more interesting to develop the idea from Example 4.10 and make use of GPO for input design. This could be useful when considering that GPO does not require many evaluations of the objective function, which corresponds to estimates of the expected information matrix.

Also, it would be interesting to consider methods that infer the parameters of the model at the same time as the optimal input. This would relax the unrealistic assumption that we need to know the true parameters to be able to construct an optimal input. There are mainly three approaches for solving this problem. The first is to pose the problem in a robust optimisation setting and only assume that the true parameter is located within some set. By this construction, the resulting optimal input would be an average (in some sense) over this set.

The second approach is to construct a sequential algorithm that first infer the parameters and then the optimal input. This is repeated over many iterations by exciting the system with the input constructed in the last iteration. However, it is difficult to prove that this approach would converge to the true optimal input and the true parameters. Finally, we could pose this problem in a Bayesian manner and marginalise over the parameters of the model. The resulting optimal input would therefore be a marginalisation over the parameter posterior. Therefore, we would not need to know the true parameters of the system. See Müller et al. (2007), Kuck et al. (2006) and Bubeck and Cesa-Bianchi (2012) for more information.

5.3 Source code and data

Source code written in Python and R for recreating most of the examples in Part I are available from the author's homepages at: `http://users.isy.liu.se/en/rt/johda87/` and `http://code.johandahlin.com/`. Furthermore, source code for recreating some of the numerical illustrations from Papers A, B, C and E are also available from the same homepages. The source code and data are provided under the MIT license with no guaranteed support and no responsibility for its use and function.

Bibliography

M. Adolfson, S. Laséen, J. Lindé, and M. Villani. RAMSES – a new general equilibrium model for monetary policy analysis. *Sveriges Riksbank Economic Review*, 2, 2007a.

M. Adolfson, S. Laséen, J. Lindé, and M. Villani. Bayesian estimation of an open economy DSGE model with incomplete pass-through. *Journal of International Economics*, 72(2):481–511, 2007b.

M. Adolfson, S. Laséen, L. Christiano, M. Trabandt, and K. Walentin. RAMSES II – Model Description. *Occasional Paper Series*, 12, 2013.

G. Amisano and O. Tristani. Euro area inflation persistence in an estimated nonlinear DSGE model. *Journal of Economic Dynamics and Control*, 34(10): 1837–1858, 2010.

S. An and F. Schorfheide. Bayesian analysis of DSGE models. *Econometric reviews*, 26(2-4):113–172, 2007.

B. D. O. Anderson and J. B. Moore. *Optimal filtering*. Courier Publications, 2005.

C. Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373, 2008.

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

A. L. Beam, S. K. Ghosh, and J. Doyle. Fast Hamiltonian Monte Carlo Using GPU Computing. *Pre-print*, 2014. arXiv:1402.4089v1.

M. A. Beaumont. Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–1160, 2003.

J. O. Berger. *Statistical decision theory and Bayesian analysis*. Springer, 1985.

M. J. Betancourt and M. Girolami. Hamiltonian Monte Carlo for Hierarchical Models. *Pre-print*, 2013. arXiv:1312.0906v1.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, USA, 2006.

T. Björk. *Arbitrage theory in continuous time*. Oxford University Press, 2004.

F. Black and M. Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, pages 637–654, 1973.

T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

P. Boyle. *Gaussian processes for regression and optimisation*. PhD thesis, Victoria University of Wellington, 2007.

M. Briers, A. Doucet, and S. Maskell. Smoothing algorithms for state-space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61–89, 2010.

E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *Pre-print*, 2010. arXiv:1012.2599v1.

P. J. Brockwell and R. A. Davis. *Introduction to time series and forecasting*. Springer, 2002.

S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

P. Bunch and S. Godsill. Improved particle approximations to the joint smoothing distribution using Markov Chain Monte Carlo. *IEEE Transactions on Signal Processing*, 61(4):956–963, 2013.

D. Burke, A. Ghosh, and W. Heidrich. Bidirectional importance sampling for direct illumination. In *Proceedings of the 16th Eurographics Symposium on Rendering Techniques*, pages 147–156, Konstanz, Germany, June 2005.

O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models*. Springer, 2005.

O. Cappé, S. J Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 95(5): 899–924, 2007.

C. M. Carvalho, M. S. Johannes, H. F. Lopes, and N. G. Polson. Particle learning and smoothing. *Statistical Science*, 25(1):88–106, 2010.

R. Casarin. Bayesian inference for generalised Markov switching stochastic volatility models, 2004. CEREMADE Journal Working Paper 0414.

G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Press, 2 edition, 2001.

K. S. Chan and J. Ledolter. Monte Carlo EM estimation for time series models involving counts. *Journal of the American Statistical Association*, 90(429):242–252, 1995.

T. Chen, E. B. Fox, and C. Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. *Pre-print*, 2014. arXiv:1402.4102v1.

S. Chib, F. Nardari, and N. Shephard. Markov chain Monte Carlo methods for stochastic volatility models. *Journal of Econometrics*, 108(2):281–316, 2002.

N. Chopin, P. E. Jacob, and O. Papaspiliopoulos. SMC^2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, 2013.

R. Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1:223–236, 2001.

J-M. Cornuet, J-M. Marin, A. Mira, and C. P. Robert. Adaptive Multiple Importance Sampling. *Pre-print*, 2011. arXiv:0907.1254v5.

N. Cressie. *Statistics for spatial data.* Wiley, 1993.

D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746, 2002.

J. Dahlin and F. Lindsten. Particle filter-based Gaussian process optimisation for parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication).

J. Dahlin and P. Svenson. A Method for Community Detection in Uncertain Networks. In *Proceedings of 2011 European Intelligence and Security Informatics Conference*, Athens, Greece, August 2011.

J. Dahlin and P. Svenson. Ensemble approaches for improving community detection methods. *Pre-print*, 2013. arXiv:1309.0242v1.

J. Dahlin, F. Johansson, L. Kaati, C. Mårtensson, and P. Svenson. A Method for Community Detection in Uncertain Networks. In *Proceedings of International Symposium on Foundation of Open Source Intelligence and Security Informatics 2012*, Istanbul, Turkey, August 2012a.

J. Dahlin, F. Lindsten, T. B. Schön, and A. Wills. Hierarchical Bayesian ARX models for robust inference. In *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*, Brussels, Belgium, July 2012b.

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis Hastings using Langevin dynamics. In *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013a.

J. Dahlin, F. Lindsten, and T. B. Schön. Inference in Gaussian models with missing data using Equalisation Maximisation. *Pre-print*, 2013b. arXiv:1308.4601v1.

J. Dahlin, F. Lindsten, and T. B. Schön. Second-order particle MCMC for Bayesian parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014a. (accepted for publication).

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Pre-print*, 2014b. arXiv:1311.0686v2.

J. Dahlin, T. B. Schön, and M. Villani. Approximate inference in state space models with intractable likelihoods using Gaussian process optimisation. Technical Report LiTH-ISY-R-3075, Department of Electrical Engineering, Linköping University, Linköping, Sweden, April 2014c.

P. Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 189–198, Orlando, FL, USA, jul 1998. ACM.

P. Del Moral. *Feynman-Kac Formulae - Genealogical and Interacting Particle Systems with Applications.* Probability and its Applications. Springer, 2004.

P. Del Moral. *Mean field simulation for Monte Carlo integration.* CRC Press, 2013.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

P. Del Moral, A. Doucet, and S. Singh. Forward smoothing using sequential Monte Carlo. *Pre-print*, 2010. arXiv:1012.5390v1.

M. Del Negro and F. Schorfheide. Priors from General Equilibrium Models for VARS. *International Economic Review*, 45(2):643–673, 2004.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1): 1–38, 1977.

R. Douc and O. Cappé. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 64–69, Zagreb, Croatia, September 2005.

R. Douc, E. Moulines, and D. S Stoffer. *Nonlinear Time Series: theory, methods and applications with R examples.* CRC Press, 2014.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

A. Doucet, P. Jacob, and A. M. Johansen. Discussion on Riemann manifold

Langevin and Hamiltonian Monte Carlo methods. Journal of the Royal Statistical Society: Series B Statistical Methodology, 73(2), p 162, 2011.

A. Doucet, M. K. Pitt, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. arXiv.org, arXiv:1210.1871, October 2012.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

C. Dubarry and R Douc. Particle approximation improvement of the joint smoothing distribution with on-the-fly variance estimation. *Pre-print*, 2011. arXiv:1107.55241.

E. Ehrlich, A. Jasra, and N. Kantas. Static Parameter Estimation for ABC Approximations of Hidden Markov Models. *Pre-print*, 2012. arXiv:1210.4683v1.

R. F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society*, pages 987–1007, 1982.

P. Fearnhead, D. Wyncoll, and J. Tawn. A sequential smoothing algorithm with linear computational cost. *Biometrika*, 97(2):447–464, 2010.

J. Fernández-Villaverde and J. F. Rubio-Ramírez. Estimating macroeconomic models: A likelihood approach. *The Review of Economic Studies*, 74(4):1059–1087, 2007.

R. A. Fisher. Theory of statistical estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 22(05):700–725, 1925.

T. Flury and N. Shephard. Bayesian inference based only on simulated likelihood: particle filter analysis of dynamic economic models. *Econometric Theory*, 27(5): 933–956, 2011.

K. Fokianos, A. Rahbek, and D. Tjøstheim. Poisson autoregression. *Journal of the American Statistical Association*, 104(488):1430–1439, 2009.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. Chapman & Hall/CRC, 3 edition, 2013.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

A. Ghosh, A. Doucet, and W. Heidrich. Sequential sampling for dynamic environment map illumination. In *Proceedings of the 17th Eurographics conference on Rendering Techniques*, pages 115–126, Nicosia, Cyprus, June 2006.

P. Giordani and R. Kohn. Adaptive independent Metropolis-Hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics*, 19(2):243–259, 2010.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):1–37, 2011.

P. Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer, 2004.

S. J. Godsill, A. Doucet, and M. West. Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168, March 2004.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proceedings of Radar and Signal Processing*, 140(2):107–113, 1993.

A. G. Gray. *Bringing tractability to generalized n-body problems in statistical and scientific computation*. PhD thesis, Carnegie Mellon University, 2003.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

E. Hecht. *Optics*. Pearson, 4 edition, 2013.

P. Hennig. Fast probabilistic optimization from noisy gradients. In *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, GA, USA, jun 2013.

S. Henriksen, A. Wills, T. B. Schön, and B. Ninness. Parallel implementation of particle MCMC methods on a GPU. In *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*, Brussels, Belgium, July 2012.

J. D. Hol, T. B. Schön, and F. Gustafsson. On resampling algorithms for particle filters. In *Proceedings of the Nonlinear Statistical Signal Processing Workshop*, Cambridge, UK, September 2006.

J. Hull. *Options, Futures, and other Derivatives*. Pearson, 7 edition, 2009.

J. Hull and A. White. The pricing of options on assets with stochastic volatilities. *The Journal of Finance*, 42(2):281–300, 1987.

D. Hultqvist, J. Roll, F. Svensson, J. Dahlin, and T. B. Schön. Detection and positioning of overtaking vehicles using 1D optical flow. In *Proceedings of the IEEE Intelligent Vehicles (IV) Symposium*, Dearborn, MI, USA, June 2014. (accepted for publication).

E. Jacquier, N. G. Polson, and P. E. Rossi. Bayesian analysis of stochastic volatility models with fat-tails and correlated errors. *Journal of Econometrics*, 122(1):185–212, 2004.

A. H. Jazwinski. *Stochastic processes and filtering theory*, volume 63. Academic press, 1970.

M. Johannes and N. Polson. MCMC Methods for Continuous-time Financial Econometrics. In Y. Ait-Sahalia and L. Hansen, editors, *Handbook of Financial Econometrics, Vol. 1: Tools and Techniques*, volume 2, pages 1–72. North-Holland, 2009.

D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation*. Prentice Hall, Upper Saddle River, NJ, USA, 2000.

N. Kantas, A. Doucet, S.S. Singh, and J.M. Maciejowski. An overview of sequential monte carlo methods for parameter estimation in general state-space models. In *IFAC Symposium on System Identification (SYSID)*, Saint-Malo, France, July 2009.

M. J. Keeling and P. Rohani. *Modeling infectious diseases in humans and animals*. Princeton University Press, 2008.

S. Kim, N. Shephard, and S. Chib. Stochastic volatility: likelihood inference and comparison with ARCH models. *The Review of Economic Studies*, 65(3): 361–393, 1998.

G. Kitagawa and S. Sato. Monte Carlo smoothing and self-organising state-space model. In A. Doucet, N. de Fretias, and N. Gordon, editors, *Sequential Monte Carlo methods in practice*, pages 177–195. Springer, 2001.

M. Klaas, M. Briers, N. de Freitas, A. Doucet, S. Maskell, and D. Lang. Fast particle smoothing: if I had a million particles. In *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, USA, June 2006.

P. E. Kloeden and E. Platen. *Numerical solution of stochastic differential equations*, volume 23. Springer, 4 edition, 1992.

J. Kronander and T. B. Schön. Robust auxiliary particle filters using multiple importance sampling. In *Proceedings of the 2014 IEEE Statistical Signal Processing Workshop (SSP)*, Gold Coast, Australia, July 2014. (accepted for publication).

J. Kronander, J. Dahlin, D. Jönsson, M. Kok, T. B. Schön, and J. Unger. Real-time Video Based Lighting Using GPU Raytracing. In *Proceedings of the 2014 European Signal Processing Conference (EUSIPCO)*, Lisbon, Portugal, September 2014a. (submitted, pending review).

J. Kronander, T. B. Schön, and J. Dahlin. Backward sequential Monte Carlo for marginal smoothing. In *Proceedings of the 2014 IEEE Statistical Signal Processing Workshop (SSP)*, Gold Coast, Australia, July 2014b. (accepted for publication).

H. Kuck, N. de Freitas, and A. Doucet. SMC samplers for Bayesian optimal nonlinear design. In *Proceedings of the 2006 Nonlinear Statistical Signal Processing Workshop*, pages 99–102, Cambridge, UK, sep 2006.

H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1): 97–106, 1964.

R. Langrock. Some applications of nonlinear and non-Gaussian state–space modelling by means of hidden Markov models. *Journal of Applied Statistics*, 38(12): 2955–2970, 2011.

R. Langrock and W. Zucchini. Hidden Markov models with arbitrary state dwell-time distributions. *Computational Statistics & Data Analysis*, 55(1):715–724, 2011.

E. L. Lehmann and G. Casella. *Theory of point estimation.* Springer, 1998.

F. Lindsten. 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, Canada, May 2013.

F. Lindsten and T. B. Schön. Backward simulation methods for Monte Carlo statistical inference. In *Foundations and Trends in Machine Learning*, volume 6, pages 1–143, August 2013.

J. S. Liu. *Monte Carlo Strategies in Scientific Computing.* Springer, 2008.

S. Livingstone and M. Girolami. Information-geometric Markov Chain Monte Carlo methods using Diffusions. *Pre-print*, 2014. arXiv:1403.7957v1.

D. J. Lizotte. *Practical Bayesian optimization.* PhD thesis, University of Alberta, 2008.

L. Ljung. *System identification: theory for the user.* Prentice Hall, 1999.

T. A. Louis. Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 44(02):226–233, 1982.

A. Marshall. The use of multi-stage sampling schemes in Monte Carlo simulations. In M. Meyer, editor, *Symposium on Monte Carlo Methods*, pages 123–140. Wiley, 1956.

G. Matheron. Principles of geostatistics. *Economic Geology*, 58(8):1246–1266, 1963.

G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley-Interscience, second edition, 2008.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability*. Cambridge University Press, 2009.

T. P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th conference on Uncertainty in Artificial Intelligence*, pages 362–369, Seattle, WA, USA, aug 2001.

S. Mitra. A review of volatility and option pricing. *International Journal of Financial Markets and Derivatives*, 2(3):149–179, 2011.

J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. In L. C. W. Dixon and G. P. Szego, editors, *Toward Global Optimization*, pages 117–129. North-Holland, 1978.

P. Müller, D. A. Berry, A. P. Grieve, M. Smith, and M. Krams. Simulation-based sequential Bayesian design. *Journal of Statistical Planning and Inference*, 137 (10):3140–3150, 2007.

K. P. Murphy. *Machine learning: a probabilistic perspective*. The MIT Press, 2012.

L. Murray. GPU acceleration of the particle filter: The Metropolis resampler. *Pre-print*, 2012. arXiv:1202.6163v1.

C. A. Naesseth. Nowcasting using Microblog Data. Bachelor thesis, Linköping University, sep 2012. LiTH-ISY-EX-ET-12/0398.

R. M. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. Jones, and X-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/ CRC Press, June 2010.

W. Neiswanger, C. Wang, and E Xing. Asymptotically Exact, Embarrassingly Parallel MCMC. *Pre-print*, 2013. arXiv:1311.4780v2.

B. Ninness and S. Henriksen. Bayesian system identification via Markov chain Monte Carlo techniques. *Automatica*, 46(1):40–51, 2010.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2 edition, 2006.

J. Nolan. *Stable distributions: models for heavy-tailed data*. Birkhauser, 2003.

B. Øksendal. *Stochastic differential equations*. Springer, 6 edition, 2010.

J. Olsson, O. Cappé, R. Douc, and E. Moulines. Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state space models. *Bernoulli*, 14(1):155–179, 2008.

M. Osborne. *Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature.* PhD thesis, University of Oxford, 2010.

M. A. Osborne, R. Garnett, S. J. Roberts, C. Hart, S. Aigrain, and N. Gibson. Bayesian quadrature for ratios. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 832–840, La Palma, Canary Islands, SP, apr 2012.

J. Owen, D. J. Wilkinson, and C. S. Gillespie. Scalable Inference for Markov Processes with Intractable Likelihoods. *Pre-print*, 2014. arXiv:1403.6886v1.

G. W. Peters, G. R. Hosack, and K. R. Hayes. Ecological non-linear state space model selection via adaptive particle Markov chain Monte Carlo. *Pre-print*, 2010. arXiv:1005.2238v1.

M. Pharr and G. Humphreys. *Physically based rendering: From theory to implementation.* Morgan Kaufmann, 2010.

M. K. Pitt. Smooth Particle Filters for Likelihood Evaluation and Maximisation. Technical Report 651, Department of economics, University of Warwick, Coventry, UK, July 2002. Warwick economic research papers.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

M. K. Pitt, R. S. Silva, P. Giordani, and R. Kohn. Auxiliary particle filtering within adaptive Metropolis-Hastings sampling. *Pre-print*, 2010. arXiv:1006.1914v1.

M. K. Pitt, R. S. Silva, P. Giordani, and R. Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012.

G. Poyiadjis, A. Doucet, and S. S. Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 2011.

C. R. Rao. *Linear Statistical Inference and Its Applications.* Wiley, 1965.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, August 1965.

C. P. Robert. *The Bayesian choice.* Springer, 2007.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer, 2 edition, 2004.

G. O. Roberts and J. S. Rosenthal. Optimal Scaling of Discrete Approximations to Langevin Diffusions. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 60(1):255–268, 1998.

G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7 (1):110–120, 1997.

S. M. Ross. *Simulation*. Academic Press, 5 edition, 2012.

H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):319–392, 2009.

T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

C. Sherlock, A. H. Thiery, G. O. Roberts, and J. S. Rosenthal. On the efficiency of pseudo-marginal random walk Metropolis algorithms. *Pre-print*, 2013. arXiv:1309.7209v1.

R. H. Shumway and D. S. Stoffer. *Time series analysis and its applications*. Springer, 3 edition, 2010.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pages 2951–2959. Curran Associates, Inc., 2012.

J. C. Spall. A stochastic approximation technique for generating maximum likelihood parameter estimates. In *American Control Conference*, pages 1161–1167, Minneapolis, MN, USA, June 1987.

R. Srikanthan and T. A. McMahon. Stochastic generation of annual, monthly and daily climate data: A review. *Hydrology and Earth System Sciences*, 5(4): 653–670, 1999.

E. Taghavi, F. Lindsten, L. Svensson, and T. B. Schön. Adaptive stopping for fast particle smoothing. In *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

R. S Tsay. *Analysis of financial time series*. John Wiley & Sons, 2 edition, 2005.

J. Unger, J. Kronander, P. Larsson, S. Gustavson, J. Löw, and A. Ynnerman. Spatially varying image based lighting using HDR-video. *Computers & graphics*, 37(7):923–934, 2013.

P. E. Valenzuela, C. R. Rojas, and H. Hjalmarsson. Optimal input design for dynamic systems: a graph theory approach. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, Florence, Italy, dec 2013.

P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. A graph/particle-based method for experiment design in nonlinear systems. In *Proceedings of the 19th*

IFAC World Congress, Cape Town, South Africa, August 2014. (accepted for publication).

E. Veach and L. J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics*, pages 419–428, Los Angeles, CA, USA., aug 1995. ACM.

E. Veach and L. J. Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76, 1997.

D. J. Wilkinson. *Stochastic modelling for systems biology*. CRC press, 2 edition, 2011.

D. A. Woolhiser. Modeling daily precipitation – progress and problems. In *Statistics in the Environmental and Earth Sciences 5*, pages 71–89. Halsted Press New York, 1992.

S. Yildirim, S. S. Singh, T. Dean, and A Jasra. Parameter Estimation in Hidden Markov Models with Intractable Likelihoods Using Sequential Monte Carlo. *Preprint*, 2013. arXiv:1311.4117v1.

S. L. Zeger. A regression model for time series of counts. *Biometrika*, 75(4): 621–629, 1988.

Part II

Publications

Paper A

Particle Metropolis-Hastings using gradient and Hessian information

Authors: J. Dahlin, F. Lindsten and T. B. Schön

Supported by the project *Probabilistic modelling of dynamical systems* (Contract number: 621-2013-5524) funded by the Swedish Research Council.

Edited version of the paper:

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Pre-print*, 2014b. arXiv:1311.0686v2.

Parts of the theory presented in this paper have also been presented in:

J. Dahlin, F. Lindsten, and T. B. Schön. Second-order particle MCMC for Bayesian parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014a. (accepted for publication).

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis Hastings using Langevin dynamics. In *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013a.

Particle Metropolis-Hastings using gradient and Hessian information

J. Dahlin⋆, F. Lindsten† and T. B. Schön‡

⋆Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden.
johan.dahlin@isy.liu.se

†Dept. of Engineering,
University of Cambridge,
CB2 1PZ Cambridge, United Kingdom.
fredrik.lindsten@eng.cam.ac.uk

‡Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
thomas.schon@it.uu.se

Abstract

Particle Metropolis-Hastings (PMH) allows for Bayesian parameter inference in nonlinear state space models by combining MCMC and particle filtering. The latter is used to estimate the intractable likelihood. In its original formulation, PMH makes use of a marginal MCMC proposal for the parameters, typically a Gaussian random walk. However, this can lead to a poor exploration of the parameter space and an inefficient use of the generated particles.

We propose two alternative versions of PMH that incorporate gradient and Hessian information about the posterior into the proposal. This information is more or less obtained as a byproduct of the likelihood estimation. Indeed, we show how to estimate the required information using a fixed-lag particle smoother, with a computational cost growing linearly in the number of particles. We conclude that the proposed methods can: (i) decrease the length of the burn-in phase, (ii) increase the mixing of the Markov chain at the stationary phase, and (iii) make the proposal distribution scale invariant which simplifies tuning.

1 Introduction

We are interested in parameter and state inference in nonlinear state space models (SSM) of the form

$$x_t|x_{t-1} \sim f_\theta(x_t|x_{t-1}), \quad y_t|x_t \sim g_\theta(y_t|x_t), \tag{1}$$

where the latent states and the measurements are denoted by $\mathbf{x} = x_{0:T} \triangleq \{x_t\}_{t=0}^T$ and $\mathbf{y} = y_{1:T} \triangleq \{y_t\}_{t=1}^T$, respectively. Here, $f_\theta(\,\cdot\,)$ and $g_\theta(\,\cdot\,)$ denote the transition and observation kernels, respectively, parametrised by the unknown static parameter vector $\theta \in \Theta \subset \mathbb{R}^d$. The initial state is distributed according to $\mu(x_0)$ which, for notational simplicity, is assumed to be independent of θ.

The aim of Bayesian parameter inference in SSMs is to compute the *parameter posterior distribution*

$$p(\theta|\mathbf{y}) = \frac{p_\theta(\mathbf{y})p(\theta)}{p(\mathbf{y})}, \tag{2}$$

where $p(\theta)$ denotes the prior distribution of θ and $p_\theta(\mathbf{y})$ denotes the likelihood function, which can be expressed as

$$p_\theta(\mathbf{y}) = p_\theta(y_1)\prod_{t=2}^{T} p_\theta(y_t|y_{1:t-1}). \tag{3}$$

The one-step ahead predictor $p_\theta(y_t|y_{1:t-1})$, and thus also the likelihood function, is in general not analytically tractable. However, unbiased estimators of the likelihood can be constructed using Sequential Monte Carlo (SMC) methods and these can be used as *plug-in estimators.* This is especially useful in Markov chain Monte Carlo (MCMC) for estimating the parameter posterior in (2).

This combination of MCMC and SMC is known as Particle MCMC (PMCMC) (Andrieu et al., 2010) or *psuedo-marginal* algorithms (Beaumont, 2003; Andrieu and Roberts, 2009). Here, we focus on a specific member of the PMCMC family called the particle Metropolis-Hastings (PMH) algorithm. The theoretical properties of PMH have been analysed in Andrieu and Vihola (2012), Pitt et al. (2012) and Doucet et al. (2012). The method has been used for a number of interesting applications in e.g. economics (Flury and Shephard, 2011), social network analysis (Everitt, 2012) and ecology (Golightly and Wilkinson, 2011).

The PMH algorithm makes use of an MCMC proposal to move the parameter, after which the (intractable) likelihood is estimated using SMC. The likelihood estimate is plugged in to the MH acceptance probability and it is thus used to decide whether or not the proposed parameter value should be accepted (see Section 2). The original PMH algorithm makes use of a marginal proposal for θ, i.e. only the current parameter is used when proposing a new parameter.

In this paper, we show that information such as the gradient and the Hessian (referred to here as the first and second order information) about the posterior can be included in the construction of the PMH proposal. This idea is first suggested in

Doucet et al. (2011) in the discussions following Girolami and Calderhead (2011). In two previous proceedings, we have applied and extend this idea with first order information (Dahlin et al., 2013) and also using second order information (Dahlin et al., 2014a). The present article builds upon, unifies, and extends this preliminary work. A first order PMH method has also been suggested in the recent preprint (Nemeth and Fearnhead, 2014). This method is similar to the one we presented in Dahlin et al. (2013), using a different estimator of the gradient.

In the context of *vanilla* MH sampling, it has been recognised that the gradient and Hessian information can been used to construct efficient proposal distributions. In the Metropolis Adjusted Langevin Algorithm (MALA) (Roberts and Stramer, 2003), a drift term is added to the proposal in the direction of the gradient of the log-posterior. Intuitively, compared to a random walk MH, this will result in a larger proportion of proposed samples in regions of high posterior probability.

This idea has been extended in the manifold MALA (mMALA) (Girolami and Calderhead, 2011), where the Hessian is used to scale the proposal distribution to also take the curvature of the log-posterior into account. Drawing parallels with the optimisation literature, mMALA shares some properties with Newton-type optimisation algorithms (where MALA is more similar to a steepest ascent method). In particular, the method is invariant to affine transformation of the parameters. This can considerably simplify the tedious tuning of the method since it removes the need for running costly pilot runs, which are commonly used to tune the covariance matrices of the random walk MH and the MALA to take the scale of the problem into account.

Related to these methods are the Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2010) and the manifold HMC (Girolami and Calderhead, 2011) algorithms. These methods build upon the same ideas as MALA and mMALA, respectively, but they make use of multiple steps in the proposal constructions. This paper will focus on incorporating the MALA and mMALA proposals into the PMH framework. However, we believe that the proposed methods will also be useful in designing PMCMC-versions of the HMC/mHMC algorithms.

In our problem, i.e. for inference in a nonlinear SSM (1), the gradient and Hessian of the log-posterior cannot be computed analytically. However, in analogue with the intractable likelihood, these quantities can be estimated using SMC algorithms, see e.g. Poyiadjis et al. (2011), Nemeth et al. (2013) and Doucet et al. (2013). This provides us with the tools to construct PMH algorithms in the flavour of the MALA and the mMALA, resulting in the two methods proposed in this paper, PMH1 and PMH2, respectively. In particular, we make use of a fixed-lag particle smoother (Kitagawa and Sato, 2001) and the Fisher (Fisher, 1925; Cappé et al., 2005) and Louis (Louis, 1982; Cappé et al., 2005) identities to estimate the gradient and the Hessian. The motivation for this is that the fixed-lag smoother only makes use of the weighted particles computed by the particle filter. Consequently, we obtain the gradient and Hessian information, basically, as a byproduct of the likelihood computation in the PMH algorithm. This results in a small computational overhead for the proposed methods when compared to the marginal method.

We provide numerical experiments to examine and illustrate the benefits of using the first and second order information and the accuracy of using the fixed-lag particle smoother. We demonstrate some interesting properties of the proposed algorithms, in particular that they enjoy (i) a shorter burn-in compared with the marginal algorithm, (ii) a better mixing of the Markov chain in the stationary phase, and (iii) a simplified tuning of the step length(s), especially when the target is non-isotropic.

2 Particle Metropolis-Hastings

In this section we review the PMH algorithm and show how the random variables used to compute the likelihood estimator can be incorporated in the proposal construction. We then outline the idea of how this can be used to construct the proposed PMH1 and PMH2 algorithms.

2.1 MH sampling with unbiased likelihoods

The MH algorithm (Metropolis et al., 1953; Hastings, 1970) is a member of the MCMC family for sampling from a target distribution $\pi(\theta)$ by simulating a carefully constructed Markov chain on Θ. The chain is constructed in such a way that it admits the target distribution as its unique stationary distribution. The algorithm consists of two steps: (i) a new parameter θ'' is sampled from a proposal distribution $q(\theta''|\theta')$ given the current state θ' and (ii) the current parameter is changed to θ'' with probability $\alpha(\theta', \theta'')$, otherwise the chain remains at the current parameter. The acceptance probability is given by

$$\alpha(\theta', \theta'') = 1 \wedge \frac{\pi(\theta'')}{\pi(\theta')} \frac{q(\theta'|\theta'')}{q(\theta''|\theta')}, \tag{4}$$

where we use the notation $a \wedge b \triangleq \min\{a, b\}$.

In this paper, we have the parameter posterior distribution (2) as the target distribution, i.e. $\pi(\theta) = p(\theta|\mathbf{y})$. This implies that the acceptance probability (4) will depend explicitly on the intractable likelihood $p_\theta(\mathbf{y})$, preventing direct application of the MH sampler to this problem. However, this difficulty can be circumvented by using a *pseudo-marginal* approach (Andrieu and Roberts, 2009).

Assume that there exists an unbiased, non-negative estimator of the likelihood $\widehat{p}_\theta(\mathbf{y}|u)$. We introduce explicitly the (multivariate) random variable $u \in \mathsf{U}$ used to construct this estimator, and we let $m_\theta(u)$ denote the probability density of u on U.

The pseudo-marginal method is then a standard MH algorithm operating in a non-standard extended space $\Theta \times \mathsf{U}$, with the *extended target*

$$\pi(\theta, u|\mathbf{y}) = \frac{\widehat{p}_\theta(\mathbf{y}|u) m_\theta(u) p(\theta)}{p(\mathbf{y})} = \frac{\widehat{p}_\theta(\mathbf{y}|u) m_\theta(u) p(\theta|\mathbf{y})}{p_\theta(\mathbf{y})},$$

and proposal distribution $m_{\theta''}(u'')q(\theta'' \mid \theta')$.

Since the likelihood estimator is unbiased, $\mathbb{E}_{u|\theta}[\widehat{p}_\theta(\mathbf{y}|u)] = p_\theta(\mathbf{y})$, it follows that the extended target admits $p(\theta|\mathbf{y})$ as a marginal. Hence, by simulating from the extended target $\pi(\theta, u|\mathbf{y})$ we obtain samples from the original target distribution $p(\theta|\mathbf{y})$ as a byproduct.

If the likelihood is estimated by using SMC (see Section 3) we obtain the PMH algorithm. The random variable u then corresponds to all the weighted particles generated by the SMC sampler. However, these random variables carry useful information, not only about the likelihood, but also about the geometry of the posterior. Therefore, we suggest to incorporate this information into the proposal, i.e. when proposing a new parameter value θ'', this is done from some distribution $q(\theta''|\theta', u')$. Note that this opens up for using a wide range of adapted proposals, possibly different from the ones considered in this work.

By this choice of proposal distribution, it follows that the resulting acceptance probability is

$$\alpha(\theta'', \theta') = 1 \wedge \frac{\widehat{p}_{\theta''}(\mathbf{y}|u'')}{\widehat{p}_{\theta'}(\mathbf{y}|u')} \frac{p(\theta'')}{p(\theta')} \frac{q(\theta''|\theta', u')}{q(\theta'|\theta'', u'')}. \tag{5}$$

Note, that the acceptance probability is the same as for the MH algorithm, but replacing the intractable likelihood with an unbiased estimator and including the extended proposal.

2.2 Constructing the first and second order proposals

We now turn to the construction of a proposal that makes use of the gradient and Hessian of the log-posterior. Following Robert and Casella (2004), we do this by a Laplace approximation of the parameter posterior around the current state θ'. Hence, consider a second order Taylor expansion of $\log p(\theta''|\mathbf{y})$ at θ':

$$\begin{aligned}\log p(\theta''|\mathbf{y}) \approx \log p(\theta'|\mathbf{y}) &+ (\theta'' - \theta')^\top \Big[\nabla \log p(\theta|\mathbf{y})\Big]_{\theta=\theta'} \\ &+ \frac{1}{2}(\theta'' - \theta')^\top \Big[\nabla^2 \log p(\theta|\mathbf{y})\Big]_{\theta=\theta'} (\theta'' - \theta').\end{aligned}$$

Taking the exponential of both sides and completing the square, we obtain

$$p(\theta''|\mathbf{y}) \approx \mathsf{N}\Big(\theta''; \theta' + \mathsf{I}_T^{-1}(\theta')\mathsf{S}_T(\theta'), \mathsf{I}_T^{-1}(\theta')\Big),$$

where $\mathsf{S}_T(\theta') = \nabla \log p(\theta|\mathbf{y})|_{\theta=\theta'}$ and $\mathsf{I}_T(\theta') = -\nabla^2 \log p(\theta|\mathbf{y})|_{\theta=\theta'}$ denote the gradient and the negative Hessian of the log-posterior, respectively.

As pointed out above, these quantities cannot be computed in closed form, but they can be estimated from the random variable u' (see Section 3). This suggests three different versions of the PMH algorithm, each resulting from a specific choice

Algorithm 1 Second order Particle Metropolis-Hastings

INPUTS: The inputs to Algorithm 2. $M > 0$ (no. MCMC steps), θ_0 (initial parameters), γ (proposal step lengths).
OUTPUT: $\theta = \{\theta_1, \ldots, \theta_M\}$ (samples from the posterior).

```
Run Algorithm 2 to obtain $\widehat{p}_{\theta_0}(\mathbf{y})$, $\widehat{\mathsf{S}}_T(\theta_0)$ and $\widehat{\mathsf{I}}_T(\theta_0)$.
for $k = 1$ to $M$ do
    Sample $\theta' \sim q(\theta'|\theta_{k-1})$ using (6) with $\widehat{\mathsf{S}}_T(\theta_{k-1})$ and $\widehat{\mathsf{I}}_T(\theta_{k-1})$.
    Run Algorithm 2 to obtain $\widehat{p}_{\theta'}(\mathbf{y})$, $\widehat{\mathsf{S}}_T(\theta')$ and $\widehat{\mathsf{I}}_T(\theta')$.
    Sample $\omega_k$ uniformly over $[0, 1]$.
    if $\omega_k < \alpha(\theta', \theta_{k-1})$ given by (5) then
        $\theta_k \leftarrow \theta'$. {Accept the parameter}
        $\{\widehat{p}_{\theta_k}(\mathbf{y}), \widehat{\mathsf{S}}_T(\theta_k), \widehat{\mathsf{I}}_T(\theta_k)\} \leftarrow \{\widehat{p}_{\theta'}(\mathbf{y}), \widehat{\mathsf{S}}_T(\theta'), \widehat{\mathsf{I}}_T(\theta')\}$.
    else
        $\theta_k \leftarrow \theta_{k-1}$. {Reject the parameter}
        $\{\widehat{p}_{\theta_k}(\mathbf{y}), \widehat{\mathsf{S}}_T(\theta_k), \widehat{\mathsf{I}}_T(\theta_k)\} \leftarrow \{\widehat{p}_{\theta_{k-1}}(\mathbf{y}), \widehat{\mathsf{S}}_T(\theta_{k-1}), \widehat{\mathsf{I}}_T(\theta_{k-1})\}$.
    end if
end for
```

of the proposals

$$q(\theta''|\theta', u') = \begin{cases} \mathsf{N}\left(\theta', \Gamma\right), & \text{[PMH0]} \\ \mathsf{N}\left(\theta' + \frac{1}{2}\widehat{\mathsf{S}}_T(\theta'|u'), \Gamma\right), & \text{[PMH1]} \\ \mathsf{N}\left(\theta' + \widehat{\mathsf{G}}(\theta'|u'), \widehat{\mathsf{H}}(\theta'|u')\right). & \text{[PMH2]} \end{cases} \tag{6}$$

Here, we use the notation $\widehat{\mathsf{G}}(\theta|u) = \frac{1}{2}\Gamma\,\widehat{\mathsf{I}}_T^{-1}(\theta|u)\,\widehat{\mathsf{S}}_T(\theta|u)$ and $\widehat{\mathsf{H}}(\theta|u) = \Gamma\,\widehat{\mathsf{I}}_T^{-1}(\theta|u)$ for the natural gradient and scaled inverse Hessian, respectively. Furthermore, Γ denotes a scaling matrix that controls the step-lengths of the proposal. For PMH0 and PMH1, Γ can be chosen as an estimate of the posterior covariance matrix. However, computing this estimate typically requires costly and tedious trial runs. For PMH2, the curvature of the problem is captured by the Hessian matrix, i.e. a single step-length can by used which can significantly simplify the tuning. It is also possible to choose different step-lengths for the drift term and for the covariance matrix of the proposal.

The proposed methods—PMH1 and PMH2—can be seen as PMCMC-analogues of the MALA and the mMALA, respectively. The final PMH2 algorithm is presented in Algorithm 1. It makes use of Algorithm 2, described in the subsequent section, to estimate the quantities needed for computing the proposal and the acceptance probability. Clearly, PMH0 and PMH1 are special cases obtained by using the corresponding proposal from (6) in the algorithm.

2.3 Properties of the first and second order proposals

In the sequel, we use a single step size $\Gamma = \gamma^2 I_d$ for all the parameters in the proposal. This is done to illustrate the advantage of adding the Hessian information, which rescales the step lengths according to the local curvature. Hence, it allows for taking larger steps when the curvature is small and vice verse. This property of PMH2 makes the algorithm scale-free in the same manner as a Newton algorithm in optimisation (see e.g. (Nocedal and Wright, 2006, Chapter 3)). That is, the parameters are dimensionless and invariant to affine transformations. Note that, since the local information is used, this is different from scaling the proposal in PMH0 with the posterior covariance matrix estimated from a pilot run as this only takes the geometry at the mode of the posterior into account.

Some analyses of the statistical properties are available for PMH0 (Sherlock et al., 2013), MH using a random walk (Roberts et al., 1997) and MALA (Roberts and Rosenthal, 1998). It is known from these analyses that adding the gradient into the proposal can increase the mixing of the Markov chain. Note that these results are obtained under somewhat strict assumptions. Also, we know from numerical experiments (Girolami and Calderhead, 2011) that there are further benefits of also adding the Hessian into the proposal.

From previous work, we also know that significant improvements can be obtain using mHMC instead of mMALA (Neal, 2010; Girolami and Calderhead, 2011). Although, we do not pursue this endeavour here, an interesting direction for future work is to extend the material presented in this work to implement a particle version of mHMC. We believe that the present work is an important step in this direction.

3 Estimation of likelihoods, gradients, and Hessians

In this section, we show how to estimate the likelihood together with the gradients and Hessians needed for the different versions of PMH. This is done with a linear computational cost using an auxiliary particle filter (APF) (Pitt and Shephard, 1999) together with the fixed-lag (FL) particle smoother (Kitagawa and Sato, 2001). Both the APF and the FL are instances of SMC algorithms and we refer to Doucet and Johansen (2011) and Del Moral et al. (2006) for a more in-depth presentation.

3.1 Auxiliary particle filter

An APF can be used to approximate the sequence of joint smoothing distributions $p_\theta(x_{1:t}|y_{1:t})$ for $t = 1$ to T. The APF makes use of a particle system consisting of N weighted particles $\{x_{1:t}^{(i)}, w_t^{(i)}\}_{i=1}^N$ to approximate the joint smoothing distribution at time t according to

$$\widehat{p}_\theta(\,\mathrm{d}x_{1:t}|y_{1:t}) \triangleq \sum_{i=1}^{N} \frac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}} \delta_{x_{1:t}^{(i)}}(\mathrm{d}x_{1:t}). \tag{7}$$

Here, $\delta_z(\mathrm{d}x_{1:t})$ denotes the Dirac measure placed at z. The system is propagated from $t-1$ to t by first sampling an *ancestor index* $a_t^{(i)}$, with

$$\mathbb{P}(a_t^{(i)} = j) = \nu_{t-1}^{(j)} \left[\sum_{k=1}^{N} \nu_{t-1}^{(k)} \right]^{-1}, \quad j = 1, \ldots, N. \tag{8}$$

Given the ancestor index, a new particle is generated from some propagation kernel,

$$x_t^{(i)} \sim R_\theta\Big(x_t | x_{1:t-1}^{a_t^{(i)}}, y_t\Big). \tag{9}$$

This sampling is repeated for each particle, i.e. for $i = 1$ to N. In the above, $\nu_{t-1}^{(i)}$ denote the resampling weights, which need not be equal to the particle importance weights in general. Finally, we append the obtained sample to the particle trajectory by $x_{1:t}^{(i)} = \{x_{1:t-1}^{a_t^{(i)}}, x_t^{(i)}\}$ and compute a new importance weight as

$$w_t^{(i)} \triangleq \frac{w_{t-1}^{a_t^{(i)}}}{\nu_{t-1}^{a_t^{(i)}}} \frac{g_\theta\Big(y_t | x_t^{(i)}\Big) f_\theta\Big(x_t^{(i)} | x_{t-1}^{a_t^{(i)}}\Big)}{R_\theta\Big(x_t^{(i)} | x_{1:t-1}^{a_t^{(i)}}, y_t\Big)}. \tag{10}$$

Hence, the empirical approximations of the smoothing distributions (7) can be computed sequentially for $t = 1$ to T by repeating (8)–(10). Note that the random variables u appearing in the extended target of the PMH algorithm correspond to all the random variables generated by the APF, i.e. all the particles and ancestor indices,

$$u = (\{x_t^{(i)}, a_t^{(i)}\}_{i=1}^{N}, t = 1, \ldots, T).$$

Two important special cases of the APF are: the bootstrap particle filter (bPF) (Gordon et al., 1993) and the fully adapted particle filter (faPF) (Pitt and Shephard, 1999). For the bPF, we select the proposal $R_\theta(x_t|x_{1:t-1}, y_t) = f_\theta(x_t|x_{t-1})$ and the auxiliary weights $\nu_t^{(i)} = w_t^{(i)} = g_\theta(y_t|x_t^{(i)})$. The faPF is obtained by choosing $R_\theta(x_t|x_{1:t-1}, y_t) = p_\theta(x_t|y_t, x_{t-1})$ and $\nu_t^{(i)} = p_\theta(y_{t+1}|x_t^{(i)})$, resulting in the weights $w_t^{(i)} \equiv 1$. Note, that the faPF can only be used in models for which these quantities are available, though partially adapted versions also exist, see Pitt and Shephard (1999).

3.2 Estimation of the likelihood

The likelihood for the SSM in (1) can be estimated using (3) by inserting estimated one-step predictors $p_\theta(y_t|y_{1:t-1})$ obtained from the APF. The resulting likelihood estimator is given by

$$\widehat{p}_\theta(\mathbf{y}|u) = \frac{1}{N^T} \sum_{i=1}^{N} w_T^{(i)} \left\{ \prod_{t=1}^{T-1} \sum_{i=1}^{N} \nu_t^{(i)} \right\}. \tag{11}$$

This likelihood estimator has been extensively studied in the SMC literature. In particular, it is known to be unbiased for any number of particles, see e.g. Pitt et al. (2012) and Proposition 7.4.1 in Del Moral (2004). As discussed in the previous section, this is exactly the property that is needed in order to obtain $p(\theta \mid \mathbf{y})$ as the unique stationary distribution for the Markov chain generated by the PMH algorithm.

Consequently, PMH will target the correct distribution for any number of particles $N \geq 1$. However, the variance in the likelihood estimate is connected with the acceptance rate and the mixing of the Markov chain. Therefore it is important to determine the number of particles that balances a reasonable acceptance rate with a reasonable computational cost. This problem is studied for the marginal PMH algorithm in Pitt et al. (2012) and Doucet et al. (2012).

3.3 Estimation of the gradient

As we shall see below, the gradient of the log-posterior can be computed by solving a smoothing problem. The APF can be used directly to address this problem, since the particles $\{x_{1:T}^{(i)}, w_T^{(i)}\}_{i=1}^N$ provide an approximation of the joint smoothing distribution at time T according to (7) (see also Poyiadjis et al. (2011)). However, this method can give estimates with high variance due to the *particle degeneracy* problem. Instead, we make use of the FL smoother (Kitagawa and Sato, 2001) which has the same linear computational cost, but smaller *particle degeneracy* than the APF. Alternative algorithms for estimating this information are also available in Del Moral et al. (2010), Poyiadjis et al. (2011) and Nemeth et al. (2013).

The gradient of the parameter log-posterior (the first order information) is given by

$$\mathsf{S}_T(\theta) = \nabla \log p(\theta) + \nabla \log p_\theta(\mathbf{y}), \tag{12}$$

where it is assumed that the gradient of the log-prior $\nabla \log p(\theta)$ can be calculated explicitly. The gradient of the log-likelihood $\nabla \log p_\theta(\mathbf{y})$, commonly referred to as the *score function*, can using *Fisher's identity* (Fisher, 1925; Cappé et al., 2005) be expressed as,

$$\nabla \log p_\theta(\mathbf{y}) = \mathbb{E}_\theta \left[\nabla \log p_\theta(\mathbf{x}, \mathbf{y}) \middle| \mathbf{y} \right]. \tag{13}$$

The gradient of the, so called, complete data log-likelihood is by (1) given by

$$\nabla \log p_\theta(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} \xi_\theta(x_t, x_{t-1}), \text{ where} \tag{14}$$
$$\xi_\theta(x_t, x_{t-1}) = \nabla \log f_\theta(x_t|x_{t-1}) + \nabla \log g_\theta(y_t|x_t).$$

Combining this with (13) results in

$$\nabla \log p_\theta(\mathbf{y}) = \sum_{t=1}^{T} \int \xi_\theta(x_t, x_{t-1}) p_\theta(x_{t-1:t}|\mathbf{y}) \, \mathrm{d}x_{t-1:t},$$

which depends on the (intractable) two-step smoothing distribution $p_\theta(x_{t-1:t}|\mathbf{y})$. To approximate the quantity above we use the FL smoother which relies on the assumption that there is a decaying influence of future observations $y_{t+\Delta:T}$ on the state x_t. This means that

$$p_\theta(x_{t-1:t}|\mathbf{y}) \approx p_\theta(x_{t-1:t}|y_{1:\kappa_t}),$$

holds for some large enough $\kappa_t = \min\{t+\Delta, T\}$. Here, Δ denotes a pre-determined lag decided by the user, which depends on the forgetting properties of the model. By marginalisation of the empirical smoothing distribution $\widehat{p}_\theta(x_{1:\kappa_t}|y_{1:\kappa_t})$ over $x_{1:t-2}$ and $x_{t+1:\kappa_t}$, we obtain the *empirical fixed-lag smoothing distribution*

$$\widehat{p}_\theta^\Delta(\mathrm{d}x_{t-1:t}|\mathbf{y}) \triangleq \sum_{i=1}^{N} w_{\kappa_t}^{(i)} \delta_{\tilde{x}_{\kappa_t,t-1:t}^{(i)}}(\,\mathrm{d}x_{t-1:t}). \tag{15}$$

Here, we use the notation $\tilde{x}_{\kappa_t,t}^{(i)}$ to denote the ancestor at time t of particle $x_{\kappa_t}^{(i)}$. Furthermore $\tilde{x}_{\kappa_t,t-1:t}^{(i)} = \{\tilde{x}_{\kappa_t,t-1}^{(i)}, \tilde{x}_{\kappa_t,t}^{(i)}\}$. Inserting (14)–(15) into (13) provides an estimate of (12),

$$\widehat{\mathsf{S}}_T(\theta|u) = \nabla \log p(\theta) + \sum_{t=1}^{T}\sum_{i=1}^{N} w_{\kappa_t}^{(i)} \xi_\theta(\tilde{x}_{\kappa_t,t}^{(i)}, \tilde{x}_{\kappa_t,t-1}^{(i)}), \tag{16}$$

which is used in the proposal distributions in (6).

3.4 Estimation of the Hessian

The negative Hessian of the parameter log-posterior (the second order information) is given by

$$\mathsf{I}_T(\theta) = -\nabla^2 \log p(\theta) - \nabla^2 \log p_\theta(\mathbf{y}). \tag{17}$$

It is assumed that the Hessian of the log-prior $\nabla^2 \log p(\theta)$ can be calculated analytically. The negative Hessian of the log-likelihood, also known as the *observed information matrix*, can using *Louis' identity* (Louis, 1982; Cappé et al., 2005) be expressed as

$$\begin{aligned} -\nabla^2 \log p_\theta(\mathbf{y}) = {} & \nabla \log p_\theta(\mathbf{y})^2 \\ & - \mathbb{E}_\theta\left[\nabla^2 \log p_\theta(\mathbf{x},\mathbf{y})|\mathbf{y}\right] \\ & - \mathbb{E}_\theta\left[\nabla \log p_\theta(\mathbf{x},\mathbf{y})^2|\mathbf{y}\right]. \end{aligned} \tag{18}$$

Here, we have introduced the notation $v^2 = vv^\top$ for a vector v. From this, we can construct an estimator of (17) using the estimate of the gradient in (16), of the form

$$\widehat{\mathsf{I}}_T(\theta|u) = -\nabla^2 \log p(\theta) + \widehat{\mathsf{S}}_T(\theta|u)^2 - \widehat{\mathsf{I}}_T^{(1)}(\theta|u) - \widehat{\mathsf{I}}_T^{(2)}(\theta|u), \tag{19}$$

where we have introduced

$$\mathsf{I}_T^{(1)}(\theta) = \mathbb{E}_\theta\left[\nabla^2 \log p_\theta(\mathbf{x},\mathbf{y})|\mathbf{y}\right], \qquad \mathsf{I}_T^{(2)}(\theta) = \mathbb{E}_\theta\left[\nabla \log p_\theta(\mathbf{x},\mathbf{y})^2|\mathbf{y}\right].$$

We obtain the estimator of the first term analogously to (16) as

$$\widehat{\mathsf{I}}_T^{(1)}(\theta|u) = \sum_{T=1}^{T}\sum_{i=1}^{N} w_{\kappa_t}^{(i)} \zeta_\theta(\tilde{x}_{\kappa_t,t}^{(i)}, \tilde{x}_{\kappa_t,t-1}^{(i)}), \text{ where} \tag{20}$$

$$\zeta_\theta(x_t, x_{t-1}) = \nabla^2 \log f_\theta(x_t|x_{t-1}) + \nabla^2 \log g_\theta(y_t|x_t).$$

The estimate of the second term needs a bit more work and we start by rewriting the last term in (18) as

$$\begin{aligned}
&\sum_{t=1}^{T}\sum_{s=1}^{T} \mathbb{E}_\theta \left[\xi_\theta(x_t, x_{t-1}) \xi_\theta(x_s, x_{s-1})^\top \middle| \mathbf{y} \right] \\
&= \sum_{t=1}^{T} \left\{ \mathbb{E}_\theta \left[(\xi_\theta(x_t, x_{t-1}))^2 \middle| \mathbf{y} \right] + \sum_{s=1}^{t-1} \mathbb{E}_\theta \left[(\xi_\theta(x_t, x_{t-1}), \xi_\theta(x_s, x_{s-1}))^\dagger \middle| \mathbf{y} \right] \right\},
\end{aligned} \tag{21}$$

where we have introduced the operator $(a,b)^\dagger = ab^\top + ba^\top$ for brevity. Consider the last term appearing:

$$\begin{aligned}
&\sum_{s=1}^{t-1} \mathbb{E}_\theta \left[\xi_\theta(x_t, x_{t-1}) \xi_\theta(x_s, x_{s-1})^\top \middle| \mathbf{y} \right] \\
&= \mathbb{E}_\theta \left[\xi_\theta(x_t, x_{t-1}) \underbrace{\left\{ \sum_{s=1}^{t-1} \mathbb{E}_\theta \left[\xi_\theta(x_s, x_{s-1}) \middle| x_{t-1}, y_{1:t-1} \right] \right\}^\top}_{\triangleq \alpha_\theta(x_{t-1})^\top} \middle| \mathbf{y} \right].
\end{aligned}$$

From this, we see that (21) can be written as an additive functional of the form

$$\sum_{t=1}^{T} \mathbb{E}_\theta \left[(\xi_\theta(x_t, x_{t-1}))^2 + (\xi_\theta(x_t, x_{t-1}), \alpha_\theta(x_{t-1}))^\dagger \middle| \mathbf{y} \right],$$

which can be estimated using the FL smoother as before. However, for this we need to compute the quantities $\alpha_\theta(x_{t-1})$. One option is to make use of a type of fixed-lag approximation for $\alpha_\theta(x_{t-1})$, by assuming that x_s and x_t are conditionally independent given $y_{1:\kappa_t}$, whenever $|s-t| > \Delta$. This approach has previously been used by Doucet et al. (2013). Alternatively, we can use a filter approximation according to

$$\widehat{\alpha}_\theta(x_t^{(i)}) = \widehat{\alpha}_\theta(x_{t-1}^{a_t^{(i)}}) + \xi_\theta(x_t^{(i)}, x_{t-1}^{a_t^{(i)}}), \tag{22}$$

for each particle from $i = 1$ to N. Note that this filter approximation suffers from the same particle degeneracy as the APF. However, this only affects a small number of terms and in our experience this approximation works sufficiently well to give estimates with reasonably low variance. The resulting estimate inserted

Algorithm 2 Est. of the likelihood, the gradient and the Hessian of the log-posterior

INPUTS: $\mathbf{y}$ (measurements), $R(\,\cdot\,)$ (propagation kernel), $\nu(\,\cdot\,)$ (weight function), $N > 0$ (no. particles), $0 < \Delta \leq T$ (lag).

OUTPUTS: $\widehat{p}_\theta(\mathbf{y})$ (est. of the likelihood), $\widehat{\mathsf{S}}_T(\theta)$ (est. of the gradient), $\widehat{\mathsf{I}}_T(\theta)$ (est. of the negative Hessian).

```
Initialise each particle $x_0^{(i)}$.
for $t = 1$ to $T$ do
    Resample and propagate each particle using (9).
    Calculate the weights for each particle using (10).
end for
Compute $\widehat{p}_\theta(\mathbf{y})$ by (11).
Compute $\widehat{\mathsf{S}}_T(\theta)$ and $\widehat{\mathsf{I}}_T(\theta)$ by (16) and (20), respectively.
Mirror the negative eigenvalues (if any) of $\widehat{\mathsf{I}}_T(\theta)$ by adding a suitable diagonal matrix
to the estimate.
```

into (21) yields

$$\widehat{\mathsf{I}}^{(2)}_T(\theta|u) = \sum_{t=1}^{T}\sum_{i=1}^{N} w^{(i)}_{\kappa t}\,\eta_\theta(\tilde{x}^{(i)}_{\kappa_t,t}, \tilde{x}^{(i)}_{\kappa_t,t-1}), \text{ where} \tag{23}$$

$$\eta_\theta(x_t, x_{t-1}) = \xi_\theta(x_t, x_{t-1})^2 + (\xi_\theta(x_t, x_{t-1}), \widehat{\alpha}_\theta(x_{t-1}))^\dagger \,.$$

Hence, the second order information can be estimated using (19) by inserting the estimates from (20), (22) and (23).

The second order proposal (6) relies on the assumption that the observed information matrix is positive definite. The estimator given in (19) does not always satisfy this, especially when the Markov chain is located far from the posterior mode. Typically, the amount of information is limited in such regions and this results in that the curvature is difficult to estimate. To cope with this issue we regularize the Hessian by adding a diagonal matrix to shift the eigenvalues to be positive (this heuristic is common also for Newton-type optimisation algorithms, see e.g. (Nocedal and Wright, 2006, Chapter 3.4)).

Note, that there are other solutions available that e.g. makes use of matrix decompositions, see (Nocedal and Wright, 2006, Chapter 3) for alternative methods to ensure positive definiteness of the Hessian. We emphasise that this regularization does not alter the stationary distribution of the Markov chain, i.e. the PMH2 algorithm will still target the correct posterior distribution.

3.5 Accuracy of the estimated gradients and Hessians

The APF and the FL smoother are analysed in Olsson et al. (2008), where it is suggested to choose the lag Δ to be proportional to $\log T$. In this case, the biases are $O(T/N)$ and $\lambda + O(T\log[T]/N)$, for the APF and the FL smoother, respectively. Here, λ denotes a term that is independent of the number of particles. The variances for the two smoothers are $O(T^2/\sqrt{N})$ and $O(T\log[T]/\sqrt{N})$, respectively.

Note that a too small lag gives a large bias in the estimate and a too large lag gives a large variance in the estimate. The choice of Δ thus controls the bias-variance trade off the estimates; we return to this choice in Section 4.

The main difference between the two smoothers is that we obtain a smaller variance, but a persisting bias for the FL smoother. With the latter, we mean that the bias does not decrease to zero in the limit when the number of particles tends to infinity. However, this bias is compensated for by the acceptance probability in the PMH algorithm, as it corresponds to a non-symmetric proposal. Hence, the invariance of the Markov kernel is not compromised and the resulting algorithm still targets the correct posterior distribution.

3.6 Resulting SMC algorithm

In Algorithm 2, we present the complete procedure that combines the APF with the FL smoother to compute the estimates need for the second order proposal (6). This corresponds to estimating the likelihood, the gradient and the negative Hessian of the parameter log-posterior.

4 Numerical illustrations

In this section, we provide illustrations of the properties of the proposed algorithms. We start by evaluating the fixed-lag approximation on a model with known parameters, and then turn to the application of the proposed methods for parameter inference.

4.1 Estimation of the log-likelihood and the gradient

We begin by illustrating the use of the FL smoother for estimating the log-likelihood and the gradient. For this end, we consider a linear Gaussian state space (LGSS) model given by

$$x_{t+1}|x_t \sim \mathsf{N}(x_{t+1}; \phi x_t, \sigma_v^2), \quad y_t|x_t \sim \mathsf{N}(y_t; x_t, \sigma_e^2). \tag{24}$$

We generate two data realisations of $T = 100$ time steps using parameters $\theta^{(1)} = \{\phi, \sigma_v^2, \sigma_e^2\} = \{0.5, 1.0, 0.1^2\}$ and $\theta^{(2)} = \{0.5, 1.0, 1.0\}$ with a known initial state $x_0 = 0$. We use the lag $\Delta = 5$ and run the bPF and the faPF with systematic resampling.

For this model, we can compute the true values of the log-likelihood and the gradient by running an RTS smoother (Rauch et al., 1965). In Figure 1, we present boxplots of the L_1-errors in the estimated log-likelihood and the gradient of the log-posterior with respect to ϕ, evaluated at the true parameters. When the observation noise is small ($\sigma_e = 0.1$), we observe that the faPF has a large advantage over the bPF for all choices of N. When the noise is larger ($\sigma_e = 1.0$), we get smaller difference in the error of the gradient estimates, but the log-likelihood estimates are still better for the faPF. Similar results are also obtained for the gradient with respect to σ_v.

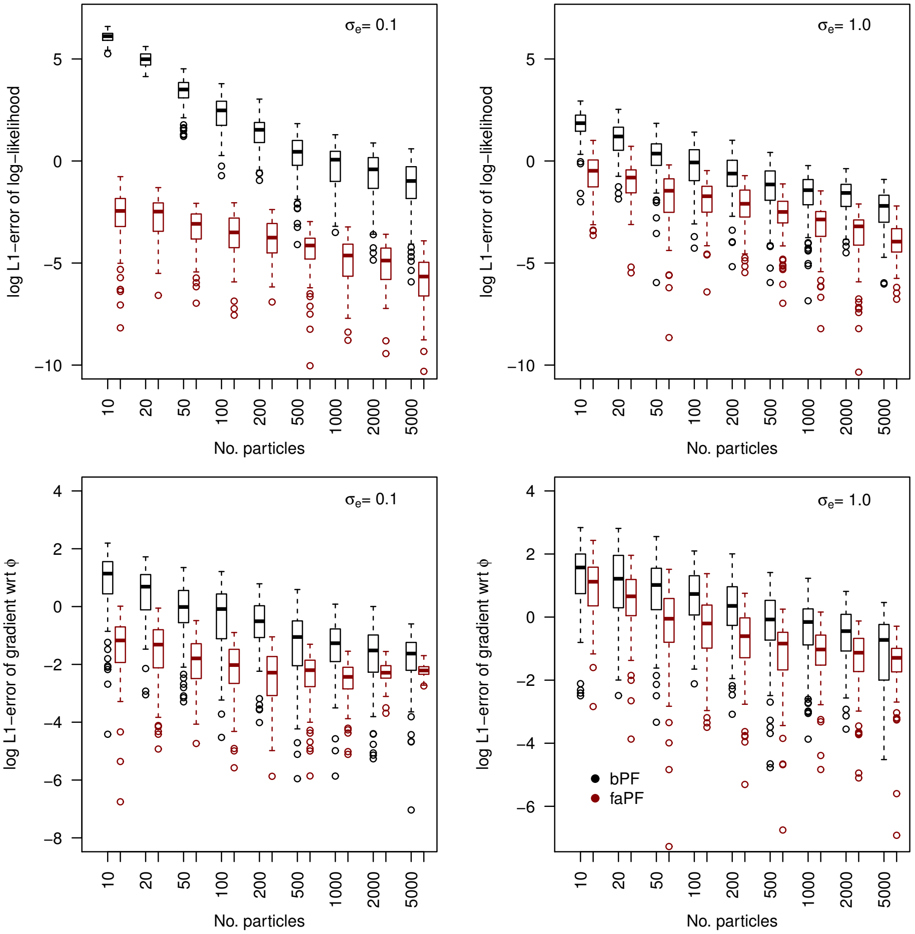

Figure 1: *The log L_1-error in the log-likelihood estimates and the estimates of the gradient with respect to ϕ in the LGSS model with $\sigma_e = 0.1$ (left) and $\sigma_e = 1$ (right). The bPF (black) and faPF (red) are evaluated by* 1 000 *Monte Carlo iterations using a fixed data set with $T = 100$.*

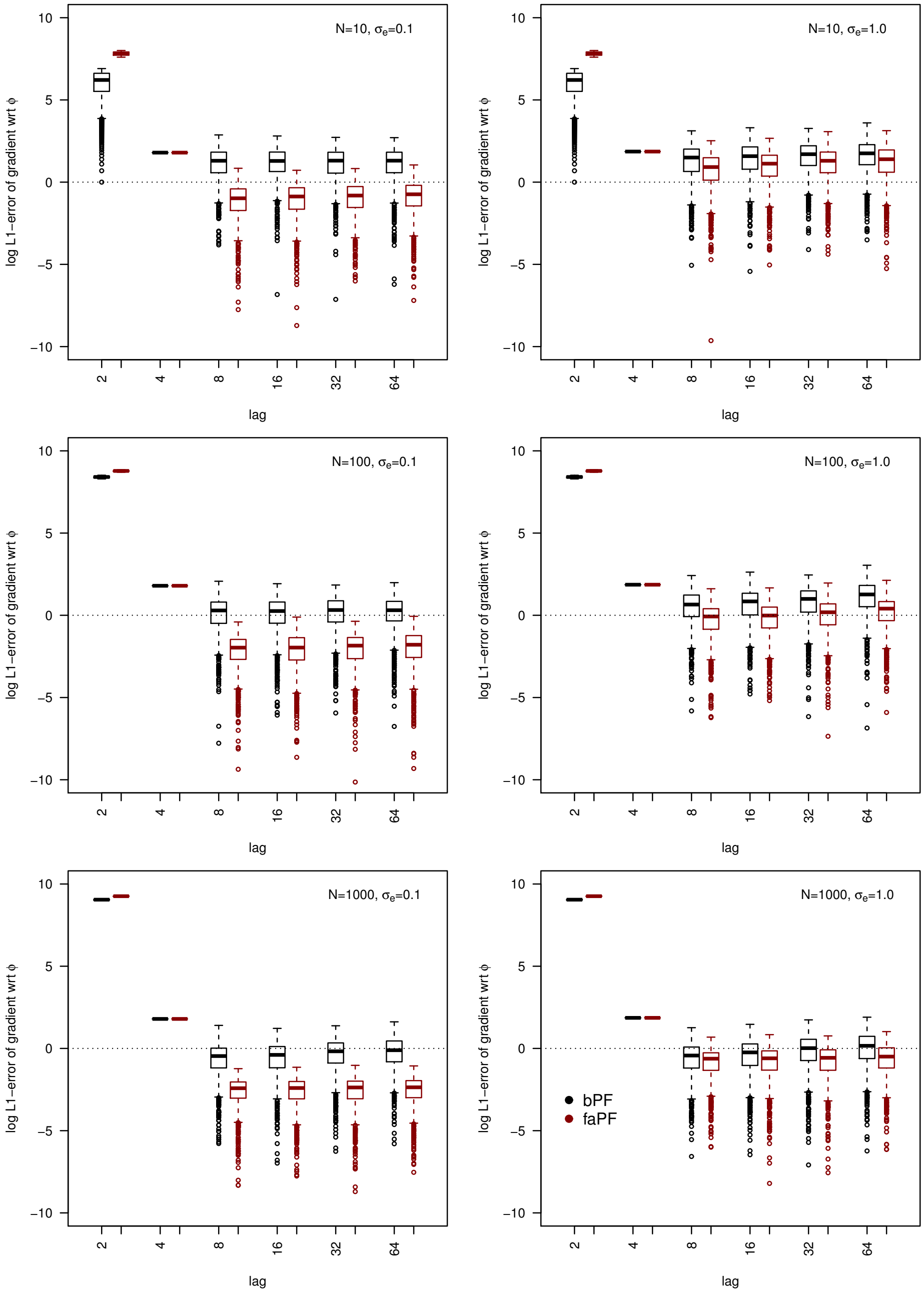

Figure 2: *The log L_1-error in the estimates of the gradient with respect to ϕ in the LGSS model with $\sigma_e = 0.1$ (left) and $\sigma_e = 1$ (right). The bPF (black) and faPF (red) are evaluated by 1 000 Monte Carlo iterations using a fixed data set with $T = 100$.*

In Figure 2, we present the error in the gradient estimates with respect to ϕ using a varying lag Δ and a varying number of particles N. The results are generated using 1 000 Monte Carlo runs on a single data set generated from the previously discussed LGSS model with $T = 100$. We conclude again that faPF is preferable when available. We also see that the lag does not affect the variance in the estimates to any large extent when using the faPF. A lag of about 12 seems to be a good choice for this model when $T = 100$ and when using the faPF with systematic resampling.

4.2 Burn-in and scale-invariance

Consider now the problem of inferring $\{\theta_1, \theta_2\} = \{\phi, \sigma_v\}$ in the LGSS model (24). We simulate a single data set with parameters $\theta^{(1)}$ (as defined in the previous section) of length $T = 250$. We use a uniform parameter prior over $|\phi| < 1, \sigma_v > 0$ and initialise the parameters in $\theta_0 = \{0.1, 2\}$. We use $N = 100$ particles in the faPF with systematic resampling and the lag $\Delta = 12$ for the FL smoother.

We select the step lengths γ to give an acceptance rate between 0.7 and 0.8 in the stationary phase, obtaining $\gamma = \{0.04, 0.065, 1.0\}$ for PMH$\{0, 1, 2\}$, respectively. Note that a single step length is used for each proposal to simplify the tuning. Of course, different step lengths can be used for each parameter, and we could also use different step lengths during the burn-in and the stationary phase of the algorithm. For example, we could have used a pilot run of PMH0 to compute an estimate of the covariance matrix of the posterior distribution and then used this information to precondition the random walk proposal. However, for this approach to be successful, it is necessary that the pilot run in itself is sufficiently well tuned to provide a useful estimate of the posterior covariance. As previously mentioned, the PMH2 algorithm avoids this (potentially difficult and time-consuming) procedure, by taking the local geometric information into account.

In the left column of Figure 3, we present the first 50 iterations of the Markov chain from the three different algorithms. We also show the *true* parameters as dotted lines and the log-likelihood function (calculated on a grid) as gray contours. We note that the added information in the proposals of PMH1 and PMH2 aids the Markov chain in the burn-in phase. This results in that the Markov chains for the proposed algorithms reach the mode of the posterior much quicker than the random walk used in PMH0.

To show the scale invariance of the PMH2-algorithm, we reparametrise the LGSS model as $\{\theta_3, \theta_4\} = \{\phi, \sigma_v/10\}$. We keep the same settings as for the previous parametrisation and rerun the algorithms. From this run we obtain the middle column in Figure 3. We see clearly that the PHM1-algorithm does not perform well and gets stuck at the initial parameter value. The reason is that the second component of the gradient is increased by a factor 10 for the rescaled model. Since we still use the same step length, this will cause the PMH1 algorithm to overshoot the region of high posterior probability when proposing new values, and these will therefore never be accepted.

Finally, to improve the performance we recalibrate the three algorithms on the

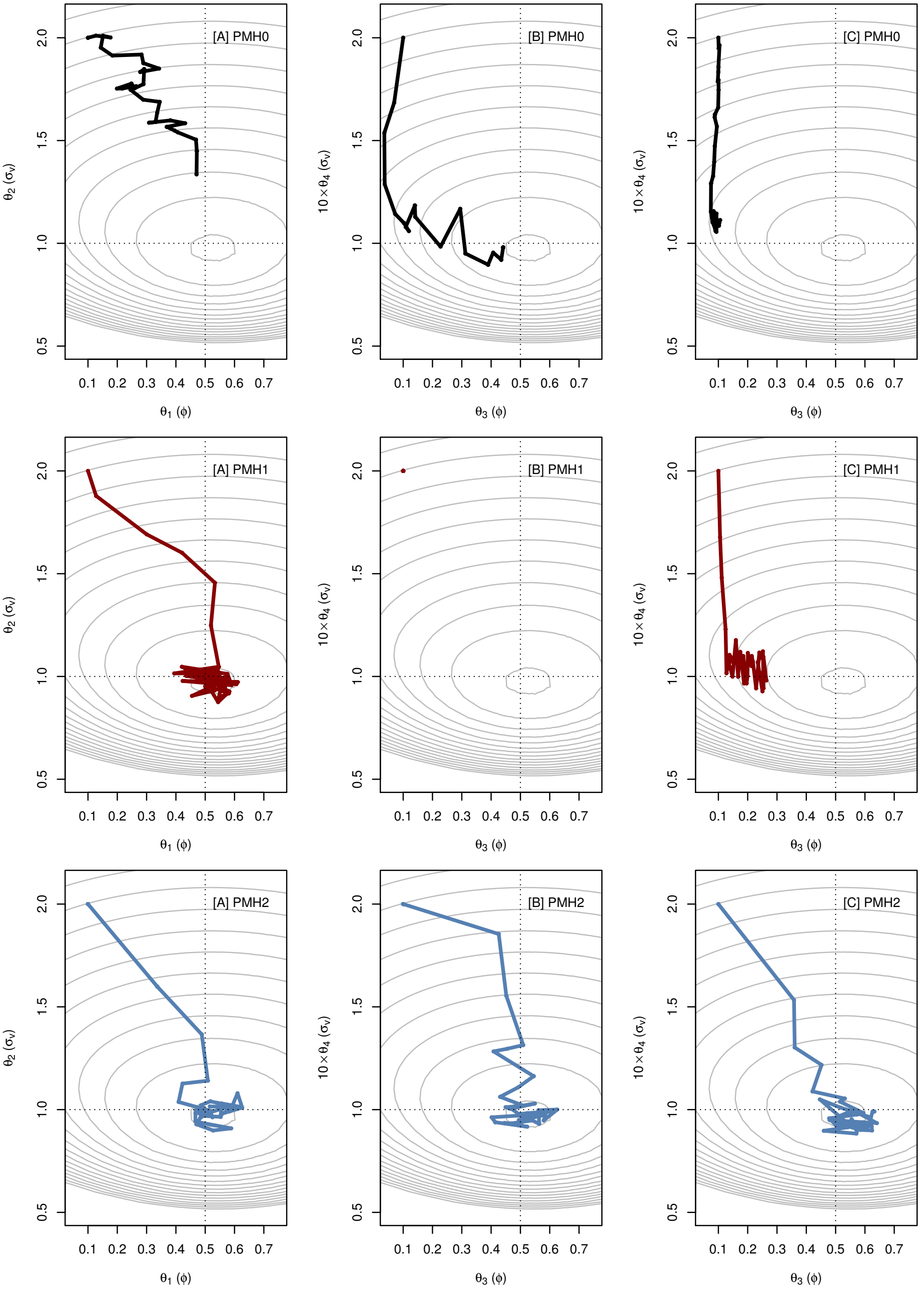

Figure 3: *The trace plots of the first 50 steps using three different proposals and LGSS models. The dotted lines show the* true *parameters of the model from which the data is generated. The gray contours show the log-likelihood function.*

new parametrisation using the same procedure as before. We then obtain the new step lengths $\{0.005, 0.0075, 1.0\}$. The resulting Markov chains are presented in the right column of Figure 3. Despite the new step lengths, PMH0 and PMH1 continue to struggle. The reason is that the step lengths are limited by the small posterior variance in the θ_4-parameter, resulting in a very slow progression in the θ_3-direction. Again, for PMH2, the added second order information is used to rescale the proposal in each dimension resulting in a much more efficient exploration of the posterior than for PMH0 and PMH1.

4.3 The mixing of the Markov chains at stationarity

We return to the LGSS model in (24) and generate 25 data sets using the parameters $\theta^{(1)}$, with the same settings as before. We are interested in investigating the mixing of the Markov chains at stationarity. For this, we use the effective sample size (ESS)

$$\mathsf{ESS}(\theta_{1:M_2}) = S\left[1 + 2\sum_{k=1}^{K} \rho_k(\theta_{1:M_2})\right]^{-1}, \tag{25}$$

where $\rho_k(\theta_{1:M_2})$ denotes the autocorrelation at lag k of the trace $\theta_{1:M_2}$ of the parameters (after the burn-in has been discarded) and S denotes the number of posterior samples. A high value of the ESS indicates that we obtain many uncorrelated samples from the target distribution, indicating that the chain is mixing well.

We use the original parameterisation, $\{\theta_1, \theta_2\} = \{\phi, \sigma_v\}$, and we initialise the algorithms at the true parameter values to avoid a long burn-in phase. To tune the step lengths for this experiment, we let the step lengths vary with a multiplier $\rho \in [0.5, 1.5]$ of a base length $\{0.10, 0.065, 2.00\}$, for each algorithm, respectively. We run the algorithms for these intervals for the first 5 simulated data sets.

In Figure 4, we present the acceptance rates and the total ESS for each algorithm using the different step lengths. We see that PMH1 and PMH2 have distinct peaks in the total ESS which both corresponds to an acceptance rate of about 60 %. From the peaks of the ESS values, we obtain the best step lengths for the three methods, given by $\{0.08, 0.075, 1.50\}$ (after correction with the step length multiplier). We stress that these values are not universal and they will likely depend on the model, the amount of data, etc.

Finally, we compute the average acceptance rate and ESS for the 25 simulated data sets during $M = 10\,000$ Monte Carlo iterations (after discarding the burn-in $M_2 = 5\,000$ iterations are left) using the same settings as before and the calibrated step lengths. The results are presented in Table 1, where the median and interquartile range (the distance between the 25% and 75% quartiles) are presented for each method using different SMC algorithms.

The added information increases the ESS about three times for PMH1 and PMH2 compared with the random walk proposal in PMH0. The extra information brought by the gradient and the Hessian improves the mixing of the Markov chains

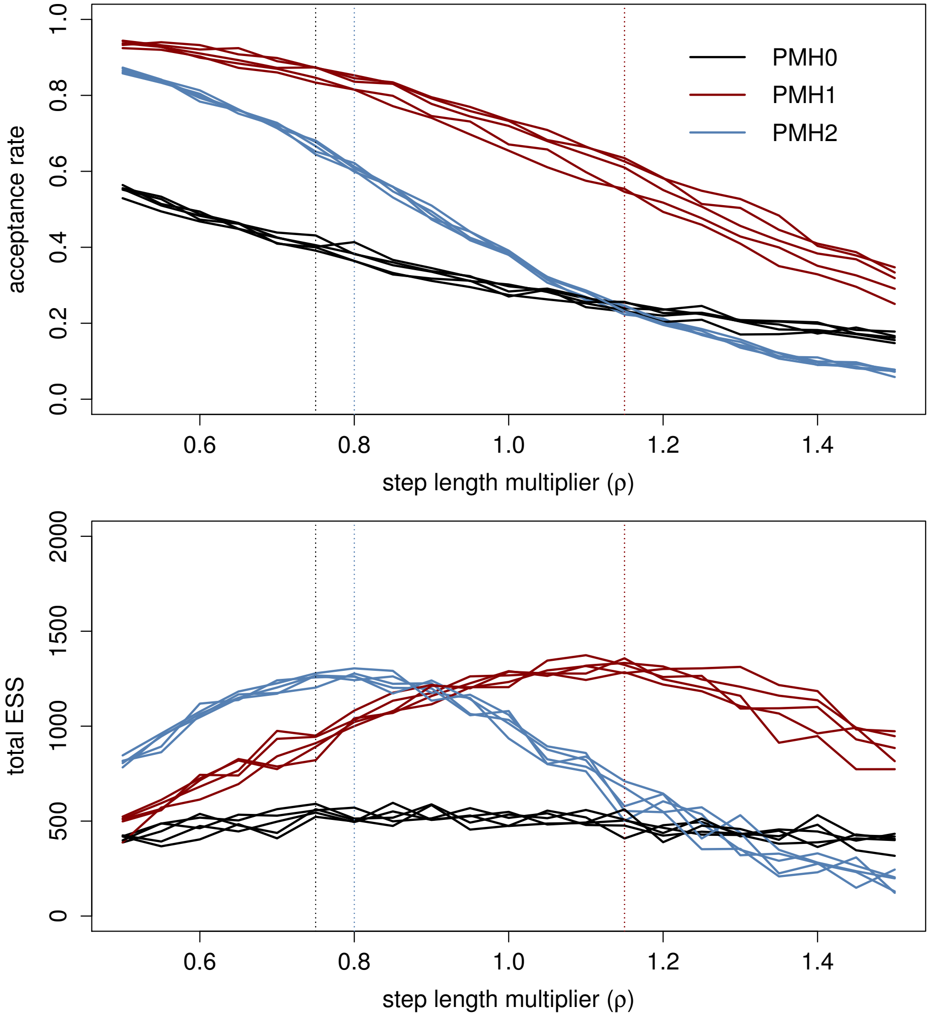

Figure 4: *Acceptance rates and total ESS for the three different algorithms in 5 different data sets using different step length multipliers. Dotted lines indicates the multipliers that gives the largest ESS for each algorithm.*

	Acc. rate	$\mathsf{ESS}(\phi)$		$\mathsf{ESS}(\sigma_v)$	
Alg.	Median	Median	IQR	Median	IQR
Zeroth order PMH (PMH0)					
bPF(500)	0.02	19	10	19	27
bPF(1000)	0.06	60	73	63	81
bPF(2000)	0.15	170	146	343	398
faPF(50)	0.37	543	446	657	458
faPF(100)	0.38	558	378	760	429
faPF(200)	0.38	674	508	678	391
First order PMH (PMH1)					
bPF(500)	0.02	27	38	21	34
bPF(1000)	0.10	79	91	104	164
bPF(2000)	0.22	222	317	303	538
faPF(50)	0.58	1440	579	**1741**	634
faPF(100)	0.59	1334	853	**1659**	361
faPF(200)	0.58	1435	651	**1475**	300
Second order PMH (PMH2)					
bPF(500)	0.03	29	29	31	48
bPF(1000)	0.10	85	213	77	97
bPF(2000)	0.24	400	430	268	219
faPF(50)	0.66	**1747**	488	1363	670
faPF(100)	0.66	**1538**	398	1100	466
faPF(200)	0.66	**1772**	755	1413	699

Table 1: *Median and interquartile ranges (IQR) for the acceptance rate and effective sample size (ESS). The values are averaged over* 25 *different data sets from the LGSS model with* $T = 250$ *using* $M = 10\,000$ *(discarding the first* 5 000 *iterations as burn-in).*

in this model, which results in a more efficient exploration of the posterior. Note that, for this parametrisation of the LGSS model the posterior is quite isotropic (which can also be seen in the left column of Figure 3). Hence, the conditions are in fact rather favourable for PMH0 and PMH1. For a more non-isotropic posterior we expect PMH2 to have a more clear advantage over the other methods, for the reasons discussed above.

5 Discussion and future work

Adding the gradient and Hessian information to the PMH proposal can have beneficial results including: (i) a shorter burn-in phase, (ii) a better mixing of the Markov chain, and (iii) scale-invariance of the proposal which simplifies tuning. The latter point is true in particular for PMH2, since this method takes the local curvature of the posterior into account, effectively making the method invariant to affine transformations.

It is common to distinguish between two phases of MCMC algorithms: the burn-in (transient) phase and the stationary phase. The proposed methods can improve upon the original PMH0 during both of these phases. However, we have seen empirically that the *best* choices for the step lengths of the algorithms can differ between the phases. Typically, a smaller step length is preferred during burn-in and a larger during stationarity (the opposite holds for PMH0). The reason for this is that during burn-in, the (natural) gradient information will heavily skew the proposal in a direction of increasing posterior probability. That is, the methods tend to be *aggressive* and propose large steps to make rapid progression toward regions of high posterior probability. While this is intuitively appealing, the problem is that we require the Markov chains to be reversible at all times. The reverse of these large steps can have very low probability which prevents them from being accepted.

One interesting direction for future work is therefore to pursue adaptive algorithms (see e.g. Andrieu and Thoms (2008), Peters et al. (2010) and Pitt et al. (2012)), to automatically tune the step lengths during the different phases of the algorithms. It can also be useful to (either adaptively on non-adaptively) use different step lengths for the drift term and for the covariance matrix in PMH1 and PMH2. Another interesting possibility is to relax the reversibility requirement during burn-in; see Diaconis et al. (2000) for a related reference. This would cause the methods to behave like optimisation procedures during the initial phase, but transition into samplers during the second phase.

The performance of the PMH2 algorithm depends on the accuracy of the estimate of the Hessian matrix. Errors in the estimate can cause the matrix to be indefinite (even when the actual Hessian is not). We have considered a simple heuristic to cope with this issue, namely to add a diagonal regulariser to the matrix whenever needed. Alternative ways of dealing with this problem include projections of the eigenvalues of the Hessian matrix, adaptive refinement of the estimate, and hybrid methods that make use of, e.g. the PMH1 proposal when the estimate of the

Hessian is deemed to be unreliable. Further investigation of these alternatives is an interesting topic for future work.

Future work also includes a thorough theoretical analysis of the proposed algorithm to determine e.g. the optimal acceptance probabilities and step lengths. Perhaps a good starting point for this could be the scenario when the data record is long, thus making the posterior approximately Gaussian. Finally, another very interesting direction for future work is to extend the proposed methods to develop particle versions of the HMC and mHMC algorithms. The reason for this is motivated by the large improvement in mixing seen in e.g. Neal (2010) and Girolami and Calderhead (2011) for high dimensional problems in *vanilla* MH sampling.

Bibliography

C. Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373, 2008.

C. Andrieu and M. Vihola. Convergence properties of pseudo-marginal Markov chain Monte Carlo algorithms. arXiv.org, arXiv:1210.1484, October 2012.

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

M. A. Beaumont. Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–1160, 2003.

O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models*. Springer, 2005.

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis Hastings using Langevin dynamics. In *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

J. Dahlin, F. Lindsten, and T. B. Schön. Second-order particle MCMC for Bayesian parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014a. (accepted for publication).

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Pre-print*, 2014b. arXiv:1311.0686v2.

P. Del Moral. *Feynman-Kac Formulae - Genealogical and Interacting Particle Systems with Applications*. Probability and its Applications. Springer, 2004.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

P. Del Moral, A. Doucet, and S. Singh. Forward smoothing using sequential Monte Carlo. *Pre-print*, 2010. arXiv:1012.5390v1.

P. Diaconis, S. Holmes, and R. Neal. Analysis of a nonreversible Markov chain sampler. *Annals of Applied Probability*, 10(3):685–1064, 2000.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

A. Doucet, P. Jacob, and A. M. Johansen. Discussion on Riemann manifold Langevin and Hamiltonian Monte Carlo methods. Journal of the Royal Statistical Society: Series B Statistical Methodology, 73(2), p 162, 2011.

A. Doucet, M. K. Pitt, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. arXiv.org, arXiv:1210.1871, October 2012.

A. Doucet, P. E. Jacob, and S. Rubenthaler. Derivative-Free Estimation of the Score Vector and Observed Information Matrix with Application to State-Space Models. *Pre-print*, 2013. arXiv:1304.5768v2.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

R. G. Everitt. Bayesian parameter estimation for latent Markov random fields and social networks. *Journal of Computational and Graphical Statistics*, 21(4): 940–960, 2012.

R. A. Fisher. Theory of statistical estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 22(05):700–725, 1925.

T. Flury and N. Shephard. Bayesian inference based only on simulated likelihood: particle filter analysis of dynamic economic models. *Econometric Theory*, 27(5): 933–956, 2011.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):1–37, 2011.

A. Golightly and D. J. Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface Focus*, 1(6):807–820, 2011.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proceedings of Radar and Signal Processing*, 140(2):107–113, 1993.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

G. Kitagawa and S. Sato. Monte Carlo smoothing and self-organising state-space model. In A. Doucet, N. de Fretias, and N. Gordon, editors, *Sequential Monte Carlo methods in practice*, pages 177–195. Springer, 2001.

T. A. Louis. Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 44(02):226–233, 1982.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

R. M. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. Jones, and X-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/ CRC Press, June 2010.

C Nemeth and P. Fearnhead. Particle Metropolis adjusted Langevin algorithms for state-space models. *Pre-print*, 2014. arXiv:1402.0694v1.

C. Nemeth, P. Fearnhead, and L. Mihaylova. Particle approximations of the score and observed information matrix for parameter estimation in state space models with linear computational cost. *Pre-print*, Jun 2013. arXiv:1306.0735v1.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2 edition, 2006.

J. Olsson, O. Cappé, R. Douc, and E. Moulines. Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state space models. *Bernoulli*, 14(1):155–179, 2008.

G. W. Peters, G. R. Hosack, and K. R. Hayes. Ecological non-linear state space model selection via adaptive particle Markov chain Monte Carlo. *Pre-print*, 2010. arXiv:1005.2238v1.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

M. K. Pitt, R. S. Silva, P. Giordani, and R. Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012.

G. Poyiadjis, A. Doucet, and S. S. Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 2011.

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, August 1965.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2 edition, 2004.

G. O. Roberts and J. S. Rosenthal. Optimal Scaling of Discrete Approximations to Langevin Diffusions. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 60(1):255–268, 1998.

G. O. Roberts and O. Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and Computing in Applied Probability*, 4(4):337–357, 2003.

G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7 (1):110–120, 1997.

C. Sherlock, A. H. Thiery, G. O. Roberts, and J. S. Rosenthal. On the efficency of pseudo-marginal random walk Metropolis algorithms. *Pre-print*, 2013. arXiv:1309.7209v1.

Paper B

Particle filter-based Gaussian process optimisation for parameter inference

Authors: J. Dahlin and F. Lindsten

Supported by the project *Probabilistic modelling of dynamical systems* (Contract number: 621-2013-5524) funded by the Swedish Research Council.

Edited version of the paper:

> J. Dahlin and F. Lindsten. Particle filter-based Gaussian process optimisation for parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication).

Particle filter-based Gaussian process optimisation for parameter inference

J. Dahlin[⋆] and F. Lindsten[†]

[⋆]Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden.
johan.dahlin@isy.liu.se

[†]Dept. of Engineering,
University of Cambridge,
CB2 1PZ Cambridge, United Kingdom.
fredrik.lindsten@eng.cam.ac.uk

Abstract

We propose a novel method for maximum-likelihood-based parameter inference in nonlinear and/or non-Gaussian state space models. The method is an iterative procedure with three steps. At each iteration a particle filter is used to estimate the value of the log-likelihood function at the current parameter iterate. Using these log-likelihood estimates, a surrogate objective function is created by utilizing a Gaussian process model. Finally, we use a heuristic procedure to obtain a revised parameter iterate, providing an automatic trade-off between exploration and exploitation of the surrogate model. The method is profiled on two state space models with good performance both considering accuracy and computational cost.

1 Introduction

We are interested in maximum likelihood-based (ML) parameter inference in nonlinear and/or non-Gaussian state space models (SSM). An SSM with latent states $x_{1:T} \triangleq \{x_t\}_{t=1}^{T}$ and measurements $y_{1:T} \triangleq \{y_t\}_{t=1}^{T}$is defined as

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

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

where $f_\theta(\cdot)$ and $g_\theta(\cdot)$ denote known distributions parame- trised by the unknown static parameter vector $\theta \in \Theta \subseteq \mathbb{R}^d$. For simplicity, we assume that the initial state x_0 is known. Let $\mathcal{L}(\theta) \triangleq p_\theta(y_{1:T})$ denote the likelihood of $y_{1:T}$ for a given value of θ. In ML estimation, we wish to estimate θ by solving,

$$\widehat{\theta}_{\text{ML}} = \underset{\theta \in \Theta}{\operatorname{argmax}}\, \mathcal{L}(\theta) = \underset{\theta \in \Theta}{\operatorname{argmax}}\, \ell(\theta), \tag{2}$$

where $\ell(\theta) \triangleq \log \mathcal{L}(\theta)$ denotes the log-likelihood function. Extensive treatments on ML inference are found in e.g. Ljung (1999) and Lehmann and Casella (1998).

The likelihood for a general SSM can be expressed as

$$\mathcal{L}(\theta) = p(y_1) \prod_{t=2}^{T} p_\theta(y_t | y_{1:t-1}), \tag{3}$$

where $p_\theta(y_t | y_{1:t-1})$ denotes the one-step predictive density. For a linear Gaussian models, these densities can be computed exactly by using the Kalman filter. However, for a nonlinear model the one-step predictive densities are in general intractable. It is therefore also intractable to evaluate the objective function in (2), which poses an obvious difficulty in addressing the ML problem.

Recently, ML estimation has been carried out in nonlinear SSMs by the aid of Sequential Monte Carlo (Doucet and Johansen, 2011). This includes e.g. using gradient-based search (Poyiadjis et al., 2011) and the Expectation Maximisation (EM) algorithm (Schön et al., 2011; Lindsten, 2013). However, some of these methods require computationally costly particle smoothing to estimate the necessary quantities, which can be a problem in some situations.

An alternative is to make use of the simultaneous perturbation stochastic approximation (SPSA) algorithm (Spall, 1987), which uses a steepest ascent algorithm with a stochastic approximation scheme to estimate the solution to (2). The gradients are estimated using finite differences with random perturbations. This results in that the algorithm only needs to sample the likelihood function twice at each iteration, independent of the dimension of the problem. SPSA is used in combination with SMC in e.g. Singh et al. (2011) and Ehrlich et al. (2012).

Another approach for maximum likelihood estimation is based on approximate inference based on Laplace approximations and moment matching. We do not consider these methods any further in this paper and refer interested readers to e.g. Bishop (2006), Khan et al. (2012) and Bell (2000) for more information.

In this paper, we propose a novel algorithm for ML estimation of static parameters in a nonlinear SSM. The method combines particle filtering (PF) with Gaussian process optimisation (GPO) (Jones, 2001; Boyle, 2007; Lizotte, 2008). The latter is a method well-suited for optimisation when it is costly to evaluate the objective function. The resulting algorithm is efficient in the sense that it provides accurate parameter estimates while making use of only a small number of (costly) log-likelihood evaluations.

2 Maximum likelihood estimation with a surrogate cost function

We now turn to our new procedure for ML estimation of general nonlinear SSMs (1). We start by outlining the main ideas of the procedure on a high level. The individual steps of the algorithm are discussed in detail in the consecutive sections. The algorithm is an iterative procedure, which thus generates a sequence of iterates $\{\theta_k\}_{k\geq 0}$ for the model parameters. Each iteration consists of three main steps:

(i) Given the current iterate θ_k, compute an estimate of the objective function (i.e. the log-likelihood) for this parameter value, denoted as $\widehat{\ell}_k \approx \ell(\theta_k)$.

(ii) Given the collection of tuples $\{\theta_j, \widehat{\ell}_j\}_{j=0}^{k}$ generated up to the current iterate, create a model of the (intractable) objective function $\ell(\theta)$.

(iii) Use the model as a surrogate for the objective function to generate a new iterate θ_{k+1}.

Note that the method requires only one estimation of the log-likelihood function at each iteration. This is promising, since it is typically computationally costly to estimate the log-likelihood value and we therefore wish to keep the number of such evaluations as low as possible.

For step (i), i.e. evaluating the log-likelihood function for a given value of θ, we use a PF, resulting in a (noisy) estimate of the objective function. This step is discussed in Section 3. For steps (ii) and (iii), we apply the GPO framework. First, we construct a surrogate for the objective function by modelling it as a Gaussian process, taking the information available in the previous iterates $\{\theta_j, \widehat{\ell}_j\}_{j=0}^{k}$ into account. This is discussed in Section 4.

Then, we make use of a heuristic, referred to as an *acquisition rule*, to find the next iterate θ_{k+1} based on the GP model. The acquisition rule is such that it favours values of θ for which the model predicts a large value of the objective function and/or where there is a high uncertainty in the model. This is useful since it automatically results in a trade-off between exploration and exploitation of the model.

In this paper, we consider a simple numerical example to illustrate the different steps of the algorithm during the derivation. For this, the linear Gaussian state space (LGSS) model,

$$x_{t+1}|x_t \sim \mathcal{N}\left(x_{t+1}; \theta x_t, 1\right), \tag{4a}$$

$$y_t|x_t \sim \mathcal{N}\left(y_t; x_t, 0.1^2\right), \tag{4b}$$

with $\Theta = [-1, 1]$ and parameter $\theta^\star = 0.5$ is simulated for $T = 250$ time steps. The complete algorithm is evaluated in Section 6 on this model, as well as on a nonlinear SSM.

3 Estimating the log-likelihood

We begin this section with a brief description of a PF. For more general introductions, see e.g. Doucet and Johansen (2011). We then continue with discussing the specific problem of likelihood estimation using the PF.

3.1 The particle filter

The PF is a sequential Monte Carlo method used to approximate e.g. the intractable filtering distribution $p_\theta(x_t|y_{1:t})$ for a general SSM (1). This is done by representing it by a set of N weighted particles $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ according to

$$\widehat{p}_\theta(\mathrm{d}x_t|y_{1:t}) \triangleq \sum_{i=1}^{N} \frac{w_t^{(i)}}{\sum_{k=1}^N w_t^{(k)}} \delta_{x_t^{(i)}}(\mathrm{d}x_t),$$

where $w_t^{(i)}$ and $x_t^{(i)}$ denote the weight and state of particle i at time t, respectively. Here, $\delta_z(\mathrm{d}x_t)$ denotes the Dirac measure located at the point z. These approximations are generated sequentially in time t. Given the particles at time $t-1$, the PF proceeds to time t by: (a) resampling, (b) propagation and (c) weighting.

In step (a), the particles are resampled with replacement, using the probabilities given by their (normalized) importance weights. This is done to rejuvenate the particle system and to put emphasis on the most probable particles. The result is an unweighted particle system $\{\widetilde{x}_{t-1}^{(i)}, 1/N\}_{i=1}^N$, targeting the same distribution $p_\theta(x_{t-1}|y_{1:t-1})$.

In step (b), the particles are propagated to time t by sampling from a proposal kernel $x_t^{(i)} \sim R_\theta\big(x_t|\widetilde{x}_{t-1}^{(i)}, y_t\big)$ from $i = 1$ to N. Finally in Step (c), the particles are assigned importance weights. This is done to account for the discrepancy between the proposal and the target densities. The importance weights are given by

$$w_t^{(i)} = W_\theta(x_t^{(i)}, \widetilde{x}_{t-1}^{(i)}) = \frac{g_\theta(y_t|x_t^{(i)}) f_\theta(x_t^{(i)}|\widetilde{x}_{t-1}^{(i)})}{R_\theta\Big(x_t^{(i)}|\widetilde{x}_{t-1}^{(i)}, y_t\Big)}. \tag{5}$$

In the sequel, we use the *bootstrap* PF which means that new particles are proposed according to the state dynamics, i.e. $R_\theta(\,\cdot\,) = f_\theta(\,\cdot\,)$ and $w_t^{(i)} = g_\theta(y_t|x_t^{(i)})$. Although more sophisticated alternatives exist, see e.g. the fully-adapted PF introduced in Pitt and Shephard (1999).

Algorithm 1 PF for log-likelihood estimation

Inputs: An SSM (1), $y_{1:T}$ (obs.) and N (no. particles).
Output: $\widehat{\ell}(\theta)$ (est. of the log-likelihood).

```
Initialise particles $x_0^{(i)}$ for $i = 1$ to $N$.
for $t = 1$ to $T$ do
    Resample the particles with weights $\{w_{t-1}^{(i)}\}_{i=1}^N$.
    Propagate the particles using $R_\theta(\,\cdot\,)$.
    Compute (5) to obtain $\{w_t^{(i)}\}_{i=1}^N$.
end for
Compute (6) to obtain $\widehat{\ell}(\theta)$.
```

3.2 Estimation of the likelihood

In order to use the PF for estimating the likelihood, we start by writing the one-step predictive density as

$$
\begin{aligned}
p_\theta(y_t|y_{1:t-1}) &= \int p_\theta(y_t, x_t|x_{t-1}) p_\theta(x_{t-1}|y_{1:t-1}) \, \mathrm{d}x_{t-1:t} \\
&= \int W_\theta(x_t, x_{t-1}) R_\theta(x_t|x_{t-1}, y_t) p_\theta(x_{t-1}|y_{1:t-1}) \, \mathrm{d}x_{t-1:t},
\end{aligned}
$$

where we have multiplied and divided with the proposal kernel $R_\theta(\,\cdot\,)$. To approximate the integral, we note that the (unweighted) particle pairs $\{\widetilde{x}_{t-1}^{(i)}, x_t^{(i)}\}_{i=1}^N$ are approximately drawn from $R_\theta(x_t|x_{t-1}, y_t)p_\theta(x_{t-1}|y_{1:t-1})$. Consequently, we obtain the Monte Carlo approximation

$$
p_\theta(y_t|y_{1:t-1}) \approx \frac{1}{N} \sum_{i=1}^N w_t^{(i)}.
$$

By inserting this approximation into (3) we obtain the particle estimate of the likelihood,

$$
\widehat{\mathcal{L}}(\theta) = \prod_{t=1}^T \left(\frac{1}{N} \sum_{i=1}^N w_t^{(i)} \right).
$$

This likelihood estimator has been studied extensively in the SMC literature. The estimator is consistent and, in fact, also unbiased for any $N \geq 1$; see e.g. Pitt et al. (2012) and Proposition 7.4.1 in Del Moral (2004). Furthermore, a central limit theorem holds,

$$
\sqrt{N} \Big[\widehat{\mathcal{L}}(\theta) - \mathcal{L}(\theta) \Big] \xrightarrow{d} \mathcal{N}\Big(0, \psi^2(\theta)\Big),
$$

for some asymptotic variance $\psi^2(\theta)$; see Proposition 9.4.1 in Del Moral (2004).

3.3 Estimation of the log-likelihood

However, working directly with the likelihood typically results in numerical difficulties. To avoid problems with numerical precision, we instead use an estimate of the log-likelihood

$$\widehat{\ell}(\theta) = \log \widehat{\mathcal{L}}(\theta) = \sum_{t=1}^{T} \log \left[\sum_{i=1}^{N} w_t^{(i)} \right] - T \log N. \tag{6}$$

The resulting complete algorithm for estimating the log-likelihood using a PF is presented in Algorithm 1.

Note that, by taking the logarithm of $\widehat{\mathcal{L}}(\theta)$, we introduce a bias into the estimator. However, by the second-order delta method (Casella and Berger, 2001), the asymptotic normality carries over to the log-likelihood estimate,

$$\sqrt{N} \left[\widehat{\ell}(\theta) - \ell(\theta) \right] \xrightarrow{d} \mathcal{N}\left(0, \gamma^2(\theta)\right), \tag{7}$$

where $\gamma(\theta) = \psi(\theta)/\mathcal{L}(\theta)$. Motivated by this, we make the assumption that the log-likelihood estimates are Gaussian distributed and centered around the true log-likelihood value. That is, we can write

$$\widehat{\ell}(\theta) = \ell(\theta) + z, \qquad z \sim \mathcal{N}(0, \sigma_z^2). \tag{8}$$

Similar normality assumptions have previously been used by Pitt et al. (2012) and Doucet et al. (2012). The unknown variance σ_z^2 is treated as a free parameter that is estimated on-the-fly as we run the proposed estimation algorithm. That is, we do *not* have to estimate σ_z^2 by making any initial test runs. We return to this in the sequel.

We validate the Gaussian assumption (8) using a small numerical experiment to illustrate the bias and variance, at a finite number of particles. We calculate 1 000 estimates of the log-likelihood $\ell(0.5)$ for the model in (4). This is done by running Algorithm 1 independently 1 000 times with $N = 1\,000$ particles.

In Figure 1, we present the distribution of the error in the estimates together with a QQ-plot. Both plots validate that the estimates are approximatively distributed according to a Gaussian distribution. Also a Lilliefors hypothesis test (Lilliefors, 1967) does not reject the null hypothesis, that the measurements are drawn from a Gaussian distribution at significance level $\alpha = 0.05$.

4 Modelling the surrogate function

From the previous, we consider a naive approach to solve (2) by creating a grid of the parameter space and estimating the log-likelihood in each grid point. The parameter estimate is then obtained as the grid point that maximises the objective function. The problem here is that as the dimension of the parameter space increases, an exponentially increasing number of grid points is required to retain the accuracy of the estimate.

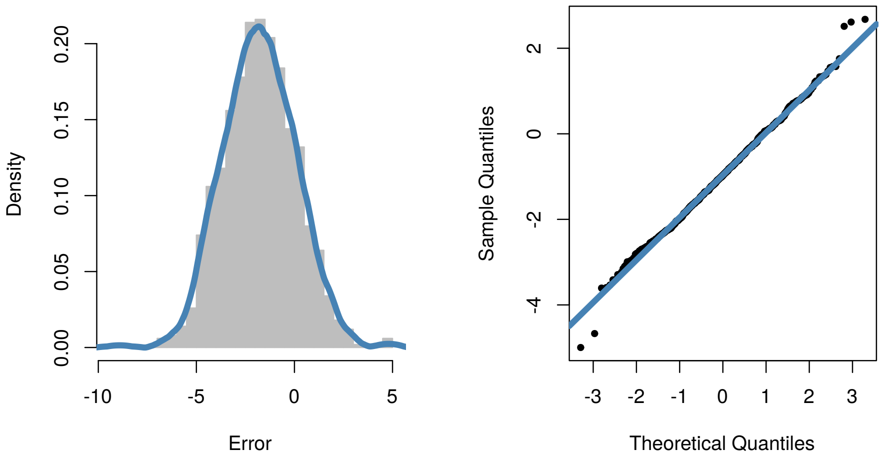

Figure 1: *Left: the histogram and kernel density estimate (blue line) of the estimation error of the log-likelihood in the LGSS model* (4) *at* $\theta = \theta^{\star}$. *Right: the QQ-plot of the data with the theoretical quantiles marked with the solid blue line.*

Furthermore, using finite differences to compute the gradient of the log-likelihood is problematic due to the noise in (8). This problem can be mitigated by using a particle smoother, as previously discussed in e.g. Poyiadjis et al. (2011), but this is even more computationally expensive than running the particle filter. Instead, we construct a model of the noisy log-likelihood evaluations in Step (ii). This model then serves as a surrogate for the actual objective function.

4.1 Gaussian process model

In this paper, we use a GP for this purpose, as these processes are possibly flexible enough to capture the overall structure of the log-likelihood for many SSMs. GPs can be seen as a generalisation of the multivariate Gaussian distribution and are commonly used as *priors over functions*. In this view, the resulting posterior obtained by conditioning upon some observations, describes the functions that could have generated the observations. This makes GPs a popular class of nonparameteric models used for e.g. regression, classification and optimisation, see e.g. Rasmussen and Williams (2006) and Murphy (2012).

In the following, we model the log-likelihood $\ell(\theta)$ as being *a priori* distributed according to a GP. That is,

$$\ell(\,\cdot\,) \sim \mathcal{GP}\Big(m(\,\cdot\,), \kappa(\,\cdot\,,\,\cdot\,)\Big), \tag{9}$$

where the process is fully described by the mean function $m(\,\cdot\,)$ and the covariance function $\kappa(\,\cdot\,,\,\cdot\,)$.

4.2 Updating the model and the hyperparameters

To ease the presentation, we here consider a particular iteration k of the GP and the PF. Let $\mathcal{D}_k = \{\boldsymbol{\theta}_k, \widehat{\boldsymbol{\ell}}_k\} = \{\theta_j, \widehat{\ell}(\theta_j)\}_{j=1}^{k}$ denote a set of iterates, where $\boldsymbol{\theta}_k$ and $\widehat{\boldsymbol{\ell}}_k$ denote vectors obtained by stacking the k parameters and noisy log-likelihood estimates, respectively.

It follows that the *posterior distribution* is given by

$$\ell(\theta)|\mathcal{D}_k \sim \mathcal{N}\Big(\mu(\theta|\mathcal{D}_k), \sigma^2(\theta|\mathcal{D}_k) + \sigma_z^2\Big), \tag{10}$$

where $\mu(\theta|\mathcal{D}_k)$ and $\sigma^2(\theta|\mathcal{D}_k)$ denote the posterior mean and variance given the iterates $\mathcal{D}_k$, respectively. By standard results for the Gaussian distribution, we have

$$\mu(\theta|\mathcal{D}_k) = m(\theta) + \kappa(\theta, \boldsymbol{\theta}_k)\Gamma^{-1}\left[\widehat{\boldsymbol{\ell}}_k - m(\theta)\right], \tag{11a}$$

$$\sigma^2(\theta|\mathcal{D}_k) = \kappa(\theta,\theta) - \kappa(\theta, \boldsymbol{\theta}_k)\Gamma^{-1}\kappa(\boldsymbol{\theta}_k, \theta), \tag{11b}$$

with $\Gamma = \kappa(\boldsymbol{\theta}_k, \boldsymbol{\theta}_k) + \sigma_z^2 \mathbf{I}_{k\times k}$, and where $\mathbf{I}_{k\times k}$ denotes a $k \times k$-identity matrix. Here we note that the posterior distribution can be sequentially updated to save computations, see the aforementioned references for details.

In the GP model presented, we use some mean function and covariance function that possibly depend on some unknown hyperparameters. Also, we need to estimate the unknown noise variance σ_z^2 in (8). For this, we adopt the emperical Bayes (EB) procedure to estimate these quantities. This is done by numerically optimising the marginal likelihood of the data with respect to the hyperparameters.

4.3 Example of log-likelihood modelling

We end this section by an example to illustrate the usefulness of GPs in modelling the log-likelihood. In the upper part of Figure 2, we show the posterior distribution of the log-likelihood of the model in (4). The posterior is estimated using three (left) and six (right) samples of the log-likelihood drawn at some randomly selected parameters. With information from only six samples, the mean of the surrogate function passes close to the observed iterates with a reasonable confidence interval.

5 Acquisition rules

The remaining problem in the proposed algorithm is how to select the parameters at which the log-likelihood should be evaluated in step (iii). A simple choice would be to consider a random sampling approach, which works well when the dimension of the parameters is small. However, when the dimension increases, we are faced with the curse-of-dimensionality and independent sampling is inefficient.

As previously discussed, we instead use acquisition rules that balances exploration and exploitation of the parameter space and makes use of the posterior distribution

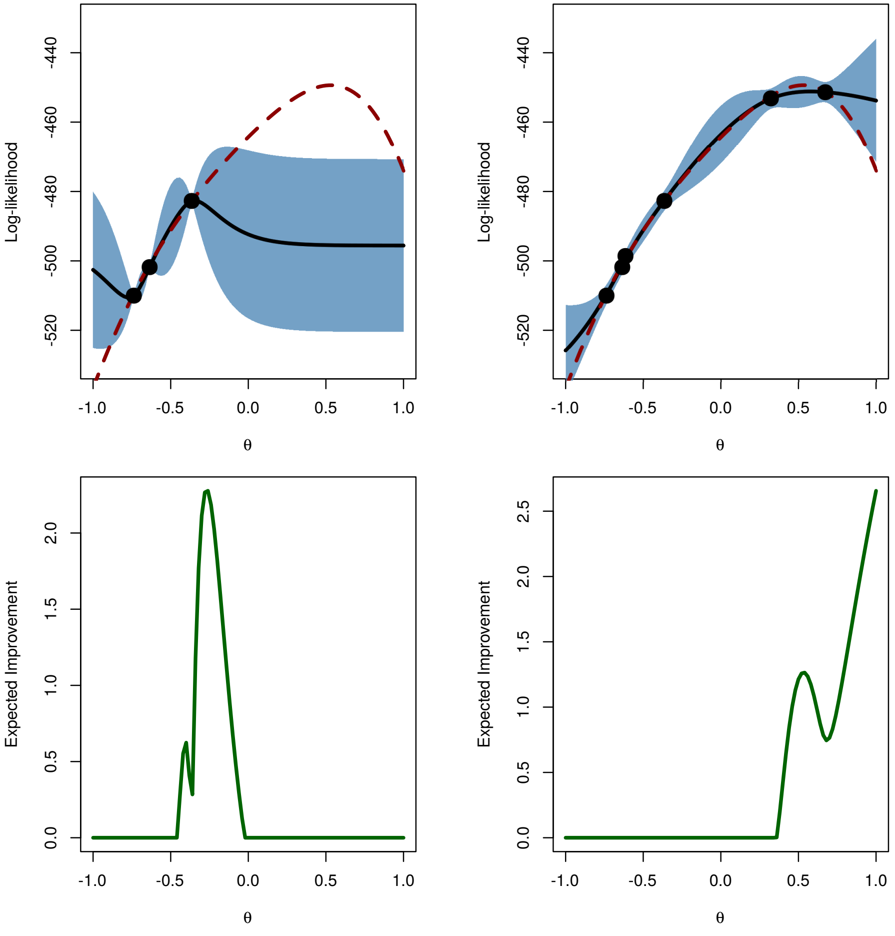

Figure 2: *Upper: The surrogate function of the LGSS model (4) using three (left) and six (right) uniform samples "●", respectively. The solid line presents the value of the predictive mean function with its 95% CI in blue and the dashed red line presents the true likelihood. Lower: The corresponding EIs using $\zeta = 0.01$.*

obtained from the GP. These heuristics are well-studied in GPO and simulation-based comparisons are presented in e.g. Lizotte (2008). In this paper, we follow their general recommendations and use expected improvement (EI) (Jones, 2001).

5.1 Expected improvement

Consider the *predicted improvement* defined as

$$I(\theta) = \max\Big\{0, \ell(\theta) - \mu_{\max} - \zeta\Big\}, \tag{12}$$

where ζ is a user-defined coefficient that balances exploration and exploitation. Also, introduce the expected peak of the log-likelihood function,

$$\mu_{\max} = \max_{\theta \in \boldsymbol{\theta}_k} \mu(\theta|\mathcal{D}_k), \tag{13a}$$

over the previous iterates. Here, we again consider a particular iteration k in the notation for brevity.

Finally, by using the posterior obtained from the GP, we can write the EI as

$$\begin{aligned} \mathbb{E}[I(\theta)|\mathcal{D}_k] &= \sigma(\theta)\Big[Z(\theta)\Phi\big(Z(\theta)\big) + \phi\big(Z(\theta)\big)\Big], \text{ with} \\ Z(\theta) &= \sigma^{-1}(\theta)\Big[\mu(\theta) - \mu_{\max} - \zeta\Big], \end{aligned} \tag{14}$$

where we drop the dependence on $\mathcal{D}_k$ for brevity. Here, Φ and ϕ denote the CDF and PDF of the standard Gaussian distribution, respectively. An acquisition rule follows by the maximising argument

$$\theta_{k+1} = \operatorname*{argmax}_{\theta \in \Theta} \mathbb{E}\Big[I(\theta)|\mathcal{D}_k\Big], \tag{15}$$

i.e. we sample the likelihood in θ_{k+1} during the next iteration of the algorithm.

In the lower part of Figure 2, the expected improvements are shown for the situation discussed in the previous example. The two situations correspond to an exploitation step (left) and an exploration step (right), respectively. In the former, we sample in the neighbourhood of the current predicted peak. In the latter, we sample in an area where the uncertainty is large to determine if there is a peak in that area.

From the expression in (14), we expect a high value of EI for parameters where the variance $\sigma(\theta)$ is large. If also the predictive mean $\mu(\theta)$ is larger than $\mu_{\max}$, then the EI assumes even larger values for these parameters. This gives the desired behaviour of the acquisition function discussed previously.

6 Numerical illustrations

Finally, we are ready to combine the methods discussed in the previous three sections into the final algorithm and it is presented in Algorithm 2. In the following, we use an LGSS model and a nonlinear model to illustrate the behaviour and the performance of the proposed algorithm. We compare the proposed method in the

Algorithm 2 Particle-based parameter inference in nonlinear SSMs using Gaussian process optimisation

```
INPUTS: Algorithm 1, $K$ (no. iterations) and $\theta_1$ (initial parameter).
OUTPUT: $\widehat{\theta}$ (est. of the parameter).

Initialise the parameter estimate in $\theta_1$.
for $k = 1$ to $K$ do
    Sample $\widehat{\ell}(\theta_k)$ using Algorithm 1.
    Compute (10) and (11) to obtain $\ell(\theta)|\mathcal{D}_k$.
    Compute (13) to obtain $\mu_{\max}$.
    Compute (15) to obtain $\theta_{k+1}$.
end for
Compute the maximiser $\mu(\theta|\mathcal{D}_K)$ to obtain $\widehat{\theta}$.
```

latter model with the SPSA algorithm (Spall, 1987). This algorithm is selected as it also only makes use of zero-order information (the log-likelihood estimates) and is known to perform well in many problems, see e.g. Spall (1998).

6.1 Implementation details

For the GP, we use a constant mean function and the Matérn kernel with $\nu = 3/2$. Note that, other choices of mean functions and kernels (especially the combination of kernels) can possibly improve the performance of the algorithm. This is especially important in models where the log-likelihood in non-isotropic.

The *GPML toolbox* (Rasmussen and Williams, 2006) is used for estimation of the hyperparameters by EB and for the computation of the predictive distribution in (10). For the acquisition function, we use the EI with $\zeta = 0.01$ following the recommendations in Lizotte (2008).

The optimisation in (15) is non-convex and therefore difficult to carry out in a global setting. Two common approaches in GPO are to use multiple local search algorithms in a Monte Carlo setting (Lizotte, 2008) or using a global optimisation algorithm (Brochu et al., 2010). In this paper, we use the latter method with the gradient-free DIRECT global optimisation algorithm (Jones et al., 1993) and the implementation written by Daniel E. Finkel, available from `http://www4.ncsu.edu/~ctk/Finkel_Direct/`. A maximum of 500 iterations and (cheap) evaluations of the surrogate function are used in the DIRECT algorithm for each optimisation.

6.2 Linear Gaussian state space model

We begin with the LGSS model using one parameter in (4), as this enables us to investigate the behaviour of the proposed algorithm in detail. We use $N = 1\,000$ particles, $K = 50$ iterations and the initial parameter $\theta_1 = -0.98$. In Figure 3, we present the surrogate function and the expected improvement at different iterations. The algorithm converges rather quickly for this simple toy example with the parameter estimate $\widehat{\theta} = 0.48$. As a comparison, the MLE obtained by the Kalman filter by maximisation on a grid of parameter values is $\theta_{\text{MLE}} = 0.44$.

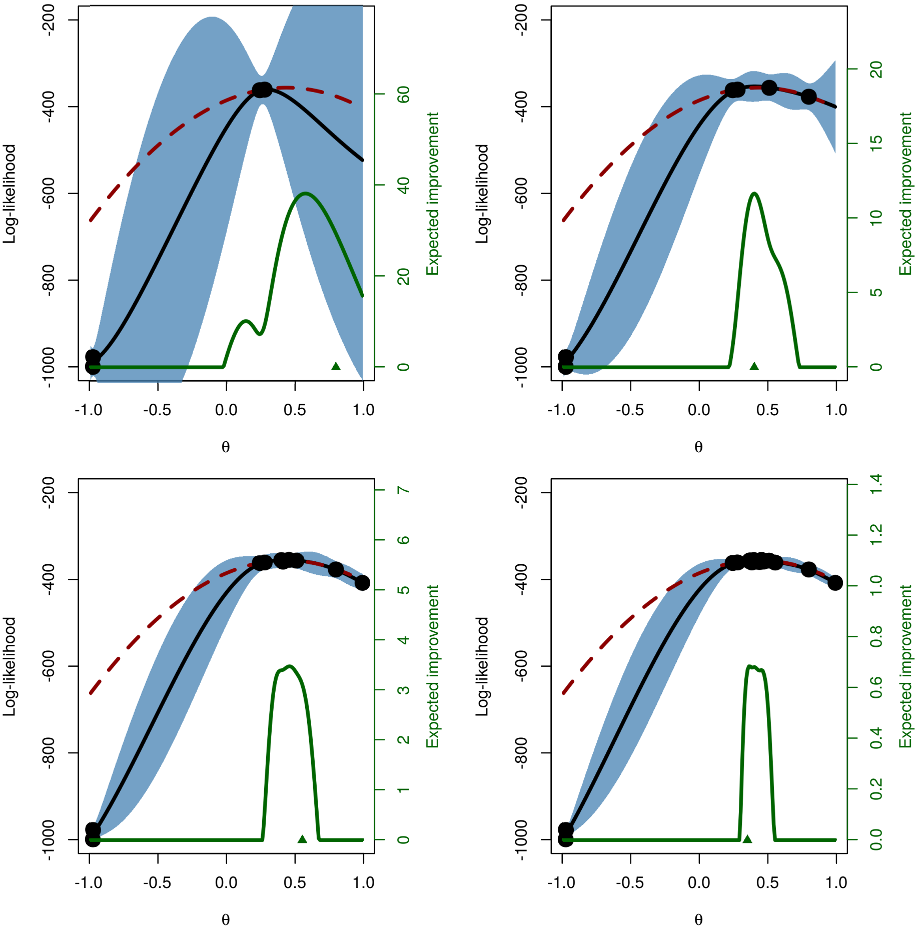

Figure 3: *The surrogate model (solid line) and EI (green line) at iterations* $\{5, 10, 15, 50\}$ *for the LGSS model. The true log-likelihood is presented as the dashed red line. The* 95% *confidence of the surrogate function is marked by blue. "●" and "△" indicate samples from the log-likelihood and the maximum of the EI obtained by the DIRECT alg.*

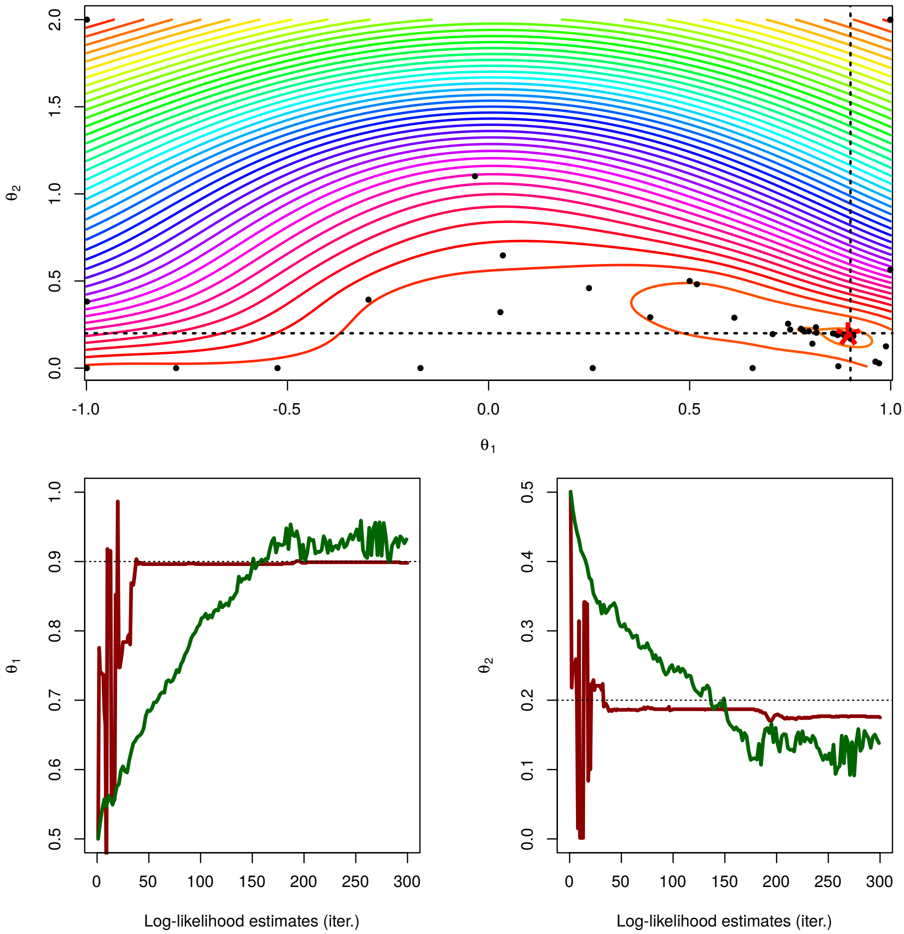

Figure 4: *Upper: the log-likelihood model generated using the K iterates with the est. parameters (red star). Lower: the estimates of θ_1 (left) and θ_2 (right) using GPO (red) and SPSA (green). The true parameters are presented by dotted lines.*

6.3 Nonlinear stochastic volatility model

Consider the Hull-White stochastic volatility model (Hull and White, 1987),

$$x_{t+1}|x_t \sim \mathcal{N}\left(x_{t+1}; \theta_1 x_t, \theta_2^2\right), \tag{16a}$$

$$y_t|x_t \sim \mathcal{N}\left(y_t; 0, 0.7^2 \exp(x_t)\right), \tag{16b}$$

where the parameters are $\theta^\star = \{\theta_1^\star, \theta_2^\star\} = \{0.90, 0.20\}$. We use $\Theta = \Theta_1 \times \Theta_2 = [-1, 1] \times [0, 2]$, $T = 250$ time steps, $N = 1\,000$ particles, $K = 300$ iterations and the initial parameter $\theta_1 = \{0.5, 0.5\}$. We implement the SPSA algorithm as suggested by Spall (1998) using the recommended settings for the parameters α, γ and C. We manually tune $a = 0.03$ and $c = 0.04$ to achieve good performance for our problem.

The GPO algorithm again converges rather quickly after about 50 evaluations of the log-likelihood and returns the parameter estimate $\widehat{\theta} = \{0.896, 0.187\}$. The SPSA algorithm converges slower and requires more than 200 evaluations of the log-likelihood to reach the neighbourhood of the true parameters. Even more iterations are required for the estimates to stabilise. This shows, for this particular example, that the GPO algorithm could be a competitive choice for ML estimation.

7 Conclusions

The results in the previous section indicate that the proposed method does not require many estimates of the intractable log-likelihood. This is due to the GP model that captures the overall structure well and enables an efficient sampling mechanism in the form of the acquisition rule. With this and the comparison with SPSA in mind, we hope that this algorithm shall turn out to be a competitive alternative to more advanced algorithms.

Important future work includes benchmarking of the proposed method, alternative acquisition rules and investigating possibilities for bias-compensation of the log-likelihood estimate. Also, the Gaussian process models can be useful as an alternative to compute the gradient (score function) and negative Hessian (the observed information matrix) of the log-likelihood. Estimating the latter is an important problem in e.g. nonlinear input design, and this approach could decrease the variance in such estimates.

At `http://users.isy.liu.se/en/rt/johda87/`, we provide source code to reproduce some of the numerical illustrations in this paper.

Acknowledgement

The authors would like to thank Prof. Thomas B. Schön, Dr. Carl E. Rasmussen, Roger Frigola and Andrew McHutchon for interesting discussions and suggestions that greatly improved this paper.

Bibliography

B. M. Bell. The marginal likelihood for parameters in a discrete Gauss-Markov process. *IEEE Transactions on Signal Processing*, 48(3):870–873, 2000.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, USA, 2006.

P. Boyle. *Gaussian processes for regression and optimisation*. PhD thesis, Victoria University of Wellington, 2007.

E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *Pre-print*, 2010. arXiv:1012.2599v1.

G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Press, 2 edition, 2001.

J. Dahlin and F. Lindsten. Particle filter-based Gaussian process optimisation for parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication).

P. Del Moral. *Feynman-Kac Formulae - Genealogical and Interacting Particle Systems with Applications*. Probability and its Applications. Springer, 2004.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

A. Doucet, M. K. Pitt, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. arXiv.org, arXiv:1210.1871, October 2012.

E. Ehrlich, A. Jasra, and N. Kantas. Static Parameter Estimation for ABC Approximations of Hidden Markov Models. *Pre-print*, 2012. arXiv:1210.4683v1.

J. Hull and A. White. The pricing of options on assets with stochastic volatilities. *The Journal of Finance*, 42(2):281–300, 1987.

D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

M. E. Khan, S. Mohamed, and K. P. Murphy. Fast bayesian inference for non-conjugate gaussian process regression. In *Proceedings of the 2012 Conference on Neural Information Processing Systems (NIPS)*, pages 3149–3157, Lake Tahoe, Nevada, USA, December 2012.

E. L. Lehmann and G. Casella. *Theory of point estimation*. Springer, 1998.

H. W. Lilliefors. On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown. *Journal of the American Statistical Association*, 62(318): 399–402, 1967.

F. Lindsten. 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, Canada, May 2013.

D. J. Lizotte. *Practical Bayesian optimization*. PhD thesis, University of Alberta, 2008.

L. Ljung. *System identification: theory for the user*. Prentice Hall, 1999.

K. P. Murphy. *Machine learning: a probabilistic perspective*. The MIT Press, 2012.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

M. K. Pitt, R. S. Silva, P. Giordani, and R. Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012.

G. Poyiadjis, A. Doucet, and S. S. Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 2011.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

S. S. Singh, N. Whiteley, and S. J. Godsill. Approximate likelihood estimation of static parameters in multi-target models. In D. Barber, A. T. Cemgil, and S. Chiappa, editors, *Inference and Learning in Dynamic Models*, pages 225–244. Cambridge University Press, 2011.

J. C. Spall. A stochastic approximation technique for generating maximum likelihood parameter estimates. In *American Control Conference*, pages 1161–1167, Minneapolis, MN, USA, June 1987.

J. C. Spall. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 34 (3):817–823, 1998.

Paper C

Approximate inference in state space models with intractable likelihoods using Gaussian process optimisation

Authors: J. Dahlin, T. B. Schön and M. Villani

Supported by the project *Probabilistic modelling of dynamical systems* (Contract number: 621-2013-5524) funded by the Swedish Research Council.

Preliminary version:

Technical Report LiTH-ISY-R-3075, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Approximate inference in state space models with intractable likelihoods using Gaussian process optimisation

J. Dahlin[⋆], T. B. Schön[†] and M. Villani[‡]

[⋆]Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden.
johan.dahlin@isy.liu.se

[†]Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
thomas.schon@it.uu.se

[‡]Dept. of Computer
and Information Science,
Linköping University,
SE-581 83 Linköping, Sweden.
mattias.villani@liu.se

Abstract

We propose a novel method for MAP parameter inference in nonlinear state space models with intractable likelihoods. The method is based on a combination of Gaussian process optimisation (GPO), sequential Monte Carlo (SMC) and approximate Bayesian computations (ABC). SMC and ABC are used to approximate the intractable likelihood by using the similarity between simulated realisations from the model and the data obtained from the system. The GPO algorithm is used for the MAP parameter estimation given noisy estimates of the log-likelihood. The proposed parameter inference method is evaluated in three problems using both synthetic and real-world data. The results are promising, indicating that the proposed algorithm converges fast and with reasonable accuracy compared with existing methods.

1 Introduction

We are interested in computing the maximum a posteriori (MAP) parameter estimate in nonlinear state space models (SSMs) with intractable likelihood functions. An SSM with latent states $x_{1:T} \triangleq \{x_t\}_{t=1}^{T}$ and measurements $y_{1:T} \triangleq \{y_t\}_{t=1}^{T}$ is defined as

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

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

where $f_\theta(\,\cdot\,)$ and $g_\theta(\,\cdot\,)$ denote known distributions parametrised by the unknown static parameter vector $\theta \in \Theta \subseteq \mathbb{R}^d$. The initial state x_0 is distributed according to $x_0 \sim \mu(x_0)$, which for simplicity is assumed to be independent of θ.

The MAP parameter estimate is given by the maximisation problem

$$\widehat{\theta}_{\text{MAP}} = \operatorname*{argmax}_{\theta \in \Theta} \log p(\theta | y_{1:T}) = \operatorname*{argmax}_{\theta \in \Theta} \Big[\ell(\theta) + \log p(\theta) \Big], \tag{2}$$

where $\log p(\theta | y_{1:T})$, $\ell(\theta)$ and $\log p(\theta)$ denote the parameter log-posterior, the log-likelihood and the log-prior, respectively. The log-likelihood is analytically intractable for SSMs but can be estimated using SMC algorithms (Doucet and Johansen, 2011).

However, SMC methods require that we can evaluate $g_\theta(y_t | x_t)$ point-wise, which is not possible for SSMs with intractable likelihoods. An example is an SSM with observation noise following the α-stable distribution to model the observation noise in the SSM. This is popular in the financial literature to model heavy-tailed behaviour observed in log-returns from the stock market. For more information about this type of models, see Yildirim et al. (2013), Chib et al. (2002) and Jacquier et al. (2004). Another example is stochastic kinetic models used in computational systems biology, see Owen et al. (2014) for more information. Another reason for intractable likelihoods can be computational infeasibility. An example of this is when the dimension of the state vector is too large for SMC algorithms to handle with reasonable computational cost.

In this paper, we propose a novel algorithm that can approximate the solution to (2) in SSMs with intractable likelihoods. The method combines approximate Bayesian computations (ABCs) (Marin et al., 2011), Gaussian process optimisation (GPO) (Lizotte, 2008; Snoek et al., 2012) and SMC to compute the parameter estimate. The likelihood is approximated using the SMC-ABC algorithm (Jasra et al., 2012) by comparing simulated realisations from the likelihood with the observed data record. The GPO algorithm is used to carry out the optimisation of the posterior to obtain the MAP estimate. The proposed method is demonstrated in three numerical illustrations using synthetic and real-world data. The results indicate that the method converges fast and quite accurately to the true parameters of the model.

Many alternative methods based on ABC have been proposed for parameter inference in models with intractable likelihoods. Examples of these methods are accept/reject sampling (Pritchard et al., 1999), Gibbs sampling (Peters et al., 2012), SMC sampling (Jasra et al., 2012) and population Monte Carlo (Beaumont et al., 2009). More specific methods for ML-base parameter inference in nonlinear SSMs are found in Ehrlich et al. (2012) and Yildirim et al. (2013). The novelty in our proposed method is the use of GPO for efficient optimisation of the posterior distribution estimated by the SMC-ABC algorithm.

The main advantage with the proposed method is the use of GPO to compute the MAP estimate. This makes the method efficient compared with alternative methods as it requires fewer computationally costly evaluations of the log-posterior. This property is the result of that the GPO operates by constructing a surrogate function to emulate the parameter posterior of the SSM. The information in the surrogate function can be used to decide where to focus the sampling of the posterior. This is the main reason for the computational gains compared with some other optimisation methods, e.g. gradient-based search.

2 An intuitive overview

In this section, we given an overview of the proposed method for MAP parameter estimate in nonlinear SSMs with intractable likelihoods. The individual steps are discussed in detail in the consecutive sections of this paper. The proposed method is an iterative method, where each iteration consists of three different steps:

(i) compute an estimate of the log-posterior distribution $\xi_k = \log \widehat{p}(\theta_k | y_{1:T})$.

(ii) build a surrogate function using $\{\boldsymbol{\theta}_k, \boldsymbol{\xi}_k\} = \{\theta_j, \xi_j\}_{j=1}^{k}$.

(iii) use an *acquisition rule* to determine θ_{k+1}.

In the first step, we make use of the SMC-ABC algorithm (Jasra et al., 2012) to sample the log-posterior. This method replaces the log-likelihood estimate by a kernel function, which compares the recorded data with simulated realisations from the likelihood. The required number of realisations is often quite large and this results in a large computational cost. We discuss this step in more detail in Section 3.

The second and third steps constitute the GPO algorithm, which is an iterative derivative-free global optimisation algorithm (Lizotte, 2008; Snoek et al., 2012). An advantage with the GPO algorithm is that it typically requires a relative small amount of samples from the objective function. Therefore this algorithm is suitable for our problem, as the log-posterior estimates are computationally costly to obtain. In Step (ii), we construct a surrogate function of the log-posterior given the collection of samples obtain from Step (i). Here, we make use of the predictive distribution of a GP as the surrogate function, which we discuss in detail in Section 4.1.

In Step (iii), we make use of the surrogate function together with a heuristic referred to as an *acquisition function* to select the next point to sample the log-posterior in. This rule selects a point where either the predictive mean and its covariance is large. In the first case, the rule is said to exploit the current information as we focus the sampling around the predicted mode. In the second case, we instead explore the parameter space to search for another higher peak. We discuss the details of this step in Section 4.2.

3 Estimating the posterior distribution

In this section, we discuss how to use a combination of SMC and ABC to estimate the intractable log-likelihood for a nonlinear SSM (1). As previously discussed, the main problem is that the log-likelihood cannot be evaluated analytically and hence the log-posterior cannot be estimated using SMC. Here, we give a short introduction to SMC-ABC and refer interested readers to e.g. Doucet and Johansen (2011) and Jasra et al. (2012) for more information.

3.1 State inference

The filtering distribution in general analytically intractable for a nonlinear SSM but can be approximated using a particle filter (PF), which is an instance of SMC algorithms. The PF is an iterative method that computes a particle system $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ for each time step t. This system consists of N particles index by $i \in \{1, \ldots, N\}$ where $x_t^{(i)}$ and $w_t^{(i)}$ denote the particle i and its importance weight. The filtering distribution can then be approximated by the *empirical filtering distribution* induced by the particle system as

$$\widehat{p}_\theta(\mathrm{d}x_t | y_{1:t}) \triangleq \sum_{i=1}^N \frac{w_t^{(i)}}{\sum_{k=1}^N w_t^{(k)}} \delta_{x_t^{(i)}}(\mathrm{d}x_t),$$

where $w_t^{(i)}$ and $x_t^{(i)}$ denote the (unnormalised) weight and state of particle i at time t, respectively. Here, $\delta_z(\mathrm{d}x_t)$ denotes the Dirac measure located at $x = z$.

The particle system consists of N particles that are computed by an iterative procedure consisting of three steps: (a) resample , (b) propagation and (c) weighting. For nonlinear SSMs with intractable likelihoods we cannot apply the standard bootstrap PF (bPF) discussed in Gordon et al. (1993) and Doucet and Johansen (2011), since the particle weight depends on the intractable $g_\theta(y_t|x_t)$.

Instead, it is suggested in Jasra et al. (2012) to augment the nonlinear SSM (1) to obtain an extended model,

$$x_t | x_{t-1} \sim f_\theta(x_t | x_{t-1}), \tag{3a}$$

$$u_t | x_t \sim g_\theta(u_t | x_t), \tag{3b}$$

$$y_t | u_t \sim \mathcal{K}_{\theta,\epsilon}(y_t | u_t), \tag{3c}$$

where $u_{1:T}$ and $\mathcal{K}_{\theta,\epsilon}(y_t|u_t)$ denote *pseudo observations* and a kernel function. Here, $\epsilon > 0$ denotes the bandwidth of the kernel and as a result also the *precision* of the approximation. To see why this construction is useful, consider the joint distribution of the states and the measurements for a nonlinear SSM (1) and its augmented version,

$$p_\theta(x_{0:T}, y_{1:T}) = \mu(x_0) \prod_{t=1}^{T} f_\theta(x_t|x_{t-1}) g_\theta(y_t|x_t), \tag{4a}$$

$$p_\theta(x_{0:T}, y_{1:T}, u_{1:T}) = \mu(x_0) \prod_{t=1}^{T} \mathcal{K}_{\theta,\epsilon}(y_t|u_t) f_\theta(x_t|x_{t-1}) g_\theta(u_t|x_t). \tag{4b}$$

If $\epsilon \to 0$, it follows from the properties of the kernel function that $u_t \to y_t$ and we recover (4a) from (4b).

By the use of the augmented SSM (3), the authors of Jasra et al. (2012) constructs a new PF algorithm in analogue with the bPF. We now proceed with discussing each of the three steps in the algorithm and how they relate to the original bPF formulation.

In Step (a), the particle system $\{x_t^{(i)}\}_{i=1}^N$ is *resampled* by sampling an ancestor index $a_t^{(i)}$ from a multinomial distribution with probabilities

$$\mathbb{P}(a_t^{(i)} = j) = w_{t-1}^{(j)} \left[\sum_{k=1}^{N} w_{t-1}^{(k)} \right]^{-1}, \qquad i, j = 1, \ldots, N. \tag{5}$$

This is done to rejuvenate the particle system and to put emphasis on the most probable particles. In Step (b), each particle is *propagated* to time t by sampling from a proposal kernel,

$$x_t^{(i)} \sim R_\theta\Big(x_t | x_{1:t-1}^{a_t^{(i)}}, y_t\Big), \qquad i = 1, \ldots, N. \tag{6}$$

For each particle, we generate a psuedo measurement $u_t^{(i)}$ by sampling from the intractable density $g_\theta(u_t|x_t)$, i.e.

$$u_t^{(i)} \sim g_\theta(u_t|x_t^{(i)}), \qquad i = 1, \ldots, N. \tag{7}$$

Finally in Step (c), each particle is assigned importance *weights*. This is done to account for the discrepancy between the proposal and the target densities. In the standard bPF algorithm, the weights are proportional to the density $g_\theta(y_t|x_t)$, which we have assumed is intractable and cannot be point-wise evaluated. Instead, we make use of the kernel to compute the importance weight for each particle by

$$w_t^{(i)} = \mathcal{K}_{\theta,\epsilon}\Big(y_t | u_t^{(i)}\Big). \tag{8}$$

Hence, we have reviewed the algorithm proposed in Jasra et al. (2012) that enables state inference in nonlinear SSMs with intractable likelihoods. The remaining question is what kernel functions are useful for this application. In this work, we mainly discuss two different kernels given by

$$\mathcal{K}_{\theta,\epsilon}(y_t|u_t) = \begin{cases} \mathbb{I}\Big[|y_t - u_t| \leq \epsilon\Big], & \text{(standard SMC-ABC)} \\ \phi_m\Big(y_t; u_t, \epsilon\, I_m\Big), & \text{(smooth SMC-ABC)} \end{cases}$$

where $|\cdot|$ denotes the $\mathcal{L}_1$-norm and $\phi_m(\,\cdot\,)$ denotes the probability density function of the m-variate normal distribution. In the following, we refer to the PF algorithms resulting from these two kernel functions as the *standard* and the *smooth* SMC-ABC algorithms, respectively,

3.2 Estimation of the log-likelihood

The estimate of the log-likelihood for nonlinear SSM with an intractable likelihood follows from calculations analogue to the tractable case, see Dahlin and Lindsten (2014). The log-likelihood for a nonlinear SSM can be written as

$$\ell(\theta) = \sum_{t=1}^{T} \log p_\theta(y_t | y_{1:t-1}),$$

where $p_\theta(y_t|y_{1:t-1})$ denotes the intractable one-step-ahead predictor. This predictor can be estimated using the Monte Carlo approximation

$$p_\theta(y_t|y_{1:t-1}) \approx \frac{1}{N} \sum_{i=1}^{N} w_t^{(i)},$$

which results in the log-likelihood estimate

$$\widehat{\ell}(\theta) = \sum_{t=1}^{T} \log \left[\sum_{i=1}^{N} w_t^{(i)} \right] - T \log N, \tag{9}$$

by the ABC approximation. Note that, the log-likelihood estimate (9) is biased for a finite number of particles N using standard and smooth SMC-ABC. However, we present some numerical illustrations in Section 6 that indicates that the bias does not largely influence the parameter estimates.

We end this section, by presenting the procedure for estimating the log-likelihood in a nonlinear SSM with an intractable likelihood in Algorithm 1. This algorithm is similar to a bPF, adding the step in which we simulate u_t and replacing the weighting function.

4 Gaussian process optimisation

In this section, we discuss the details of Steps (ii) and (iii) in the proposed algorithm. As previously mentioned, these steps correspond to the GPO algorithm and more information regarding the details of this algorithm is available in Lizotte (2008), Snoek et al. (2012) and Boyle (2007).

4.1 Constructing the surrogate function

In this work, we make use of a GP prior to model the log-posterior distribution and assume that the errors of the log-posterior estimates are Gaussian distributed. This results in that the surrogate function in the GPO algorithm is given by the predictive distribution obtained from the GP. From Bayes' theorem, it also follows

Algorithm 1 SMC-ABC for likelihood estimation

INPUTS: An SSM (1), $y_{1:T}$ (observations), N (no. particles), $\mathcal{K}_{\theta,\epsilon}(\,\cdot\,)$ (ABC kernel function) and ϵ (precision).
OUTPUT: $\widehat{\ell}(\theta)$ (est. of the log-likelihood).

```
Sample $x_0^{(i)} \sim \mu_\theta(x_0)$ for $i = 1, \ldots, N$.
for $t = 1$ to $T$ do
    Sample the ancestor indices to obtain $\{a_t^{(i)}\}_{i=1}^N$ using (5).
    Propagate the particles to obtain to obtain $\{x_t^{(i)}\}_{i=1}^N$ using (6).
    Simulate the pseudo observation to obtain $\{u_t^{(i)}\}_{i=1}^N$ using (7).
    Compute the particle weights to obtain $\{w_t^{(i)}\}_{i=1}^N$ using (8).
end for
Compute (9) to obtain $\widehat{\ell}(\theta)$.
```

that the predictive distribution is given by a Gaussian distribution as both the prior and the data are Gaussian.

To formalise this, we assume that the log-posterior is observed in Gaussian noise,

$$\xi_k = \log \widehat{p}(\theta_k | y_{1:T}) = \log p(\theta_k | y_{1:T}) + z_k, \qquad z_t \sim \mathcal{N}(0, \sigma_z^2),$$

where σ_z^2 denotes some unknown variance, which we estimate in a later stage of the algorithm. To compute the surrogate function, we assume that the log-posterior can be modelled by a Gaussian process prior (Rasmussen and Williams, 2006),

$$\log p(\theta | y_{1:T}) \sim \mathcal{GP}\Big(m(\theta), \kappa(\theta, \theta')\Big), \tag{10}$$

where $m(\theta)$ and $\kappa(\theta, \theta')$ denote the mean and the covariance function, respectively. The resulting *predictive distribution* is given by standard properties of the Gaussian distribution as

$$p(\theta | y_{1:T}) | \mathcal{D}_k \sim \mathcal{N}\Big(\mu(\theta | \mathcal{D}_k), \sigma^2(\theta | \mathcal{D}_k)\Big), \tag{11}$$

where we have introduced that notation $\mathcal{D}_k = \{\boldsymbol{\theta}_k, \boldsymbol{\xi}_i\}$ for the information available about the parameter log-posterior at iteration k. Here, the mean and covariance of the *posterior distribution* of the GP are given by

$$\mu(\theta | \mathcal{D}_k) = m(\theta) + \kappa(\theta, \boldsymbol{\theta}_k)\Big[\kappa(\boldsymbol{\theta}_k, \boldsymbol{\theta}_k) + \sigma_z^2 \mathbf{I}_{k \times k}\Big]^{-1} \Big\{\boldsymbol{\xi}_k - m(\theta)\Big\}, \tag{12a}$$

$$\sigma^2(\theta | \mathcal{D}_k) = \kappa(\theta, \theta) - \kappa(\theta, \boldsymbol{\theta}_k)\Big[\kappa(\boldsymbol{\theta}_k, \boldsymbol{\theta}_k) + \sigma_z^2 \mathbf{I}_{k \times k}\Big]^{-1} \kappa(\boldsymbol{\theta}_k, \theta) + \sigma_z^2. \tag{12b}$$

Hence, we can construct the surrogate function of the log-posterior by using (11) obtained from the GP. The hyperparameters in this model are hidden within the mean and covariance functions. These are estimated by maximising the marginal likelihood of the data with respect to these parameters. This is a standard methodology in Gaussian process modelling and is often referred to as *emperical Bayes* (EB), see Rasmussen and Williams (2006).

4.2 The acquisition rule

In this section, we discuss how to select the next parameter to sample the parameter posterior in, given the Gaussian process model from Step (ii). The aim is to construct an *acquisition rule* that selects the next sampling point. In this work, we make use of the expected improvement (EI) (Jones, 2001) as it is generally recommended by Lizotte (2008) for GPO applications.

The EI is calculated using the predictive mean and variance from the Gaussian process model. The main objective of the EI rule is to balance the exploration of the parameter space and the exploitation of the current information. By the use of the predictive distribution, we can compute confidence bounds on the log-posterior. These bounds can be used to make decisions on where the peak of the function is most likely to be located. This enables the GPO algorithm to focus its attention on areas where the uncertainty is large or where the mode is most likely to be found. Therefore, exploring the interesting parts of the parameter space and neglecting the remaining parts. The EI rule (Jones, 2001) incorporates these properties and is calculated as

$$\mathsf{EI}(\theta|\mathcal{D}_k) = \sigma(\theta|\mathcal{D}_k)\Big[Z(\theta)\Phi\big(Z(\theta)\big) + \phi\big(Z(\theta)\big)\Big], \tag{13a}$$

$$Z(\theta) = \sigma^{-1}(\theta|\mathcal{D}_k)\Big[\mu(\theta|\mathcal{D}_k) - \mu_{\max} - \zeta\Big], \tag{13b}$$

where $\mu_{\max}$ and ζ denote the maximum value of $\mu(\theta|\mathcal{D}_k)$ and a user defined parameter that controls the exploitation/exploration behaviour, respectively. In this work, use use $\zeta = 0.01$ as is recommended in Lizotte (2008). Finally, the next parameter in which to sample the parameter posterior is obtained by

$$\theta_{k+1} = \underset{\theta\in\Theta}{\operatorname{argmax}}\,\mathsf{EI}(\theta|\mathcal{D}_k).$$

From practical experience of the authors, it is often useful to add some noise to θ_{k+1} when making inference in SSMs. This is done to improve the exploration of the area around the peak, thus increasing the accuracy of the obtained parameter estimates. This *jittering* can be expressed as

$$\theta_{k+1} = \xi_k + \underset{\theta\in\Theta}{\operatorname{argmax}}\,\mathsf{EI}(\theta|\mathcal{D}_k), \qquad \xi_k \sim \mathcal{U}[-\sigma_\xi, \sigma_\xi], \tag{14}$$

where σ_ξ is some small value determined by the user.

5 Putting the algorithm together

In the previous, we have discussed the three steps of the proposed algorithm in detail. Thereby, we are ready to present the complete procedure for GPO in nonlinear SSMs with an intractable likelihood in Algorithm 2.

In the current implementation, we use an affine (constant and linear) mean function and the Matérn kernel with $\nu = 3/2$ for the Gaussian process prior. Note that, the choice of mean and covariance functions has a large impact on the result

Algorithm 2 Parameter inference in intractable nonlinear SSMs using GPO-ABC

INPUTS: Algorithm 1, K (no. iterations), $p(\theta)$ (parameter prior), $m(\theta)$ (mean function), $\kappa(\theta, \theta')$ (kernel function), θ_1 (initial parameter) and σ_η (jittering factor).
OUTPUT: $\widehat{\theta}$ (est. of the parameter).

```
Initialise the parameter estimate in $\theta_1$.
for $k = 1$ to $K$ do
    Sample $\widehat{\ell}(\theta_k)$ using Algorithm 1.
    Compute the posterior estimate, $\xi_k = \log \widehat{p}(\theta_k | y_{1:T}) = \log p(\theta_k) + \widehat{\ell}(\theta_k)$.
    Compute (11) using (12) to obtain $p(\theta|y_{1:T})|\mathcal{D}_k$.
    Compute $\mu_{\max} = \operatorname{argmax}_{\theta \in \boldsymbol{\theta}_k} \mu(\theta|\mathcal{D}_k)$.
    Compute (14) using (13) to obtain $\theta_{k+1}$.
end for
Compute the maximiser $\mu(\theta|\mathcal{D}_K)$ to obtain $\widehat{\theta}$.
```

of the proposed method and possibly has to be tailored for each SSM individually.

The optimisation of the EI and $\mu(\theta|\mathcal{D}_K)$ are possibly non-convex and therefore difficult to carry out in a global setting. Two common approaches in GPO are to use a few local gradient-based search algorithms starting at randomly selected points (Lizotte, 2008), or using some global optimisation algorithm (Brochu et al., 2010). In this work, we use the latter method with the gradient-free DIRECT global optimisation algorithm (Jones et al., 1993). A maximum of 1 000 iterations and function evaluations (of the posterior given by the GP) are used in the DIRECT algorithm for each optimisation.

6 Numerical illustrations

In this section, we provide the reader with some numerical illustrations of the performance of the proposed method. First, we estimate the parameters of an α-stable distribution to model real-world financial data with outliers. Second, we use the proposed method for parameter inference in a linear Gaussian system. Finally, we infer the parameters in a nonlinear stochastic volatility model with α-stable returns using real-world financial data. In the first and third illustrations, the likelihood function is intractable and inference must be carried out using ABC or other approximate methods. The second illustration serves only as a comparison to the case where the likelihood can be computed exactly.

6.1 Inference in α-stable data

Consider the problem of estimating the parameters of the model

$$y_t|\theta \sim \mathcal{A}(\alpha, \beta, 0.01, 0),$$

where parameters are $\theta = \{\alpha, \beta\}$ and $\mathcal{A}(\alpha, \beta, \gamma, \eta)$ denotes an α-stable distribution[1] with stability parameter α, skewness parameter β, scale parameter γ and

[1] See Appendix A for an introduction to α-stable distributions.

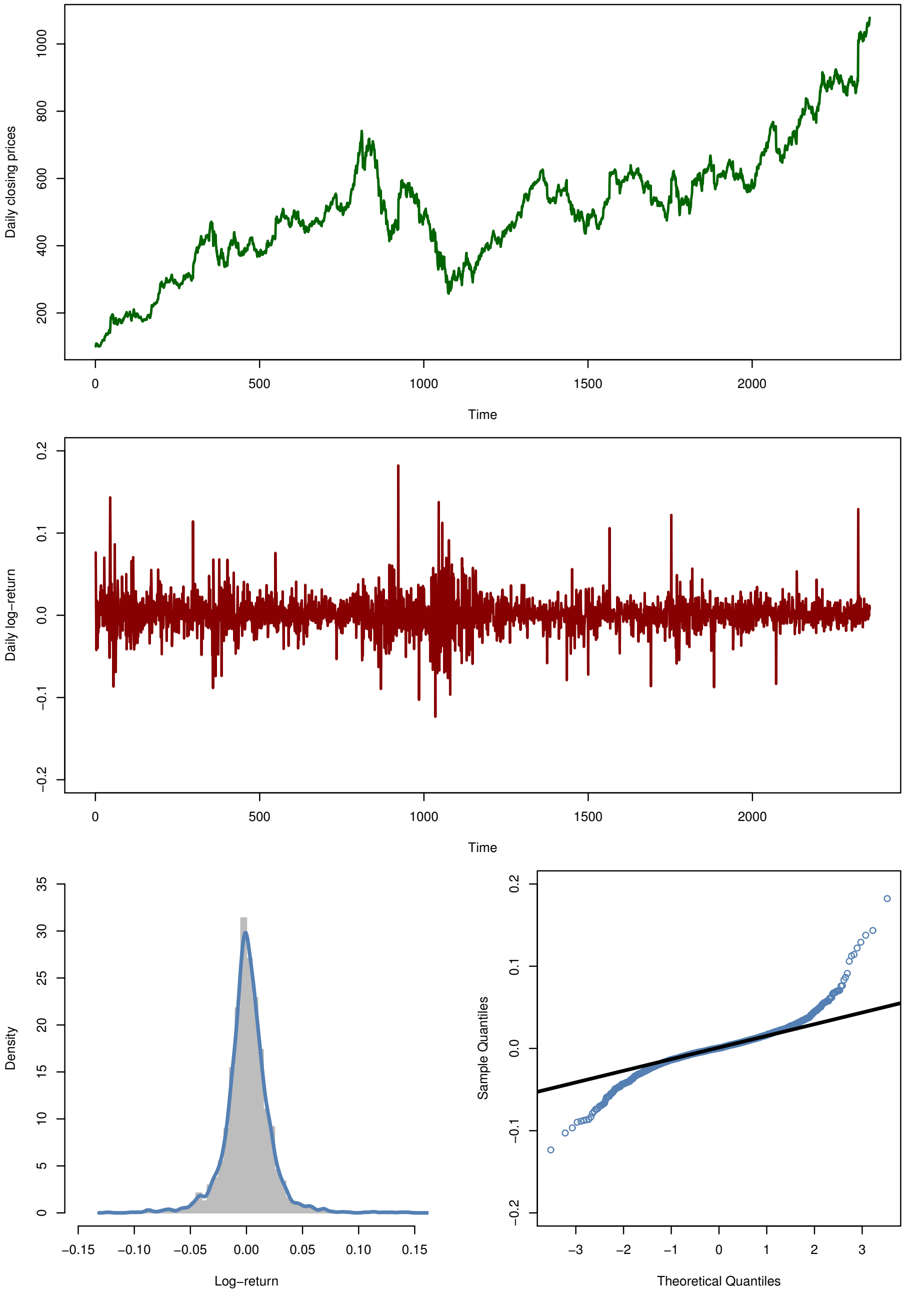

Figure 1: *The closing prices (upper), the log-returns (middle), the histogram with a kernel density estimate (lower left) and QQ-plot (lower right) of the log returns. The data is the Google stock at NASDAQ during the period 2004-08-19 – 2013-12-09.*

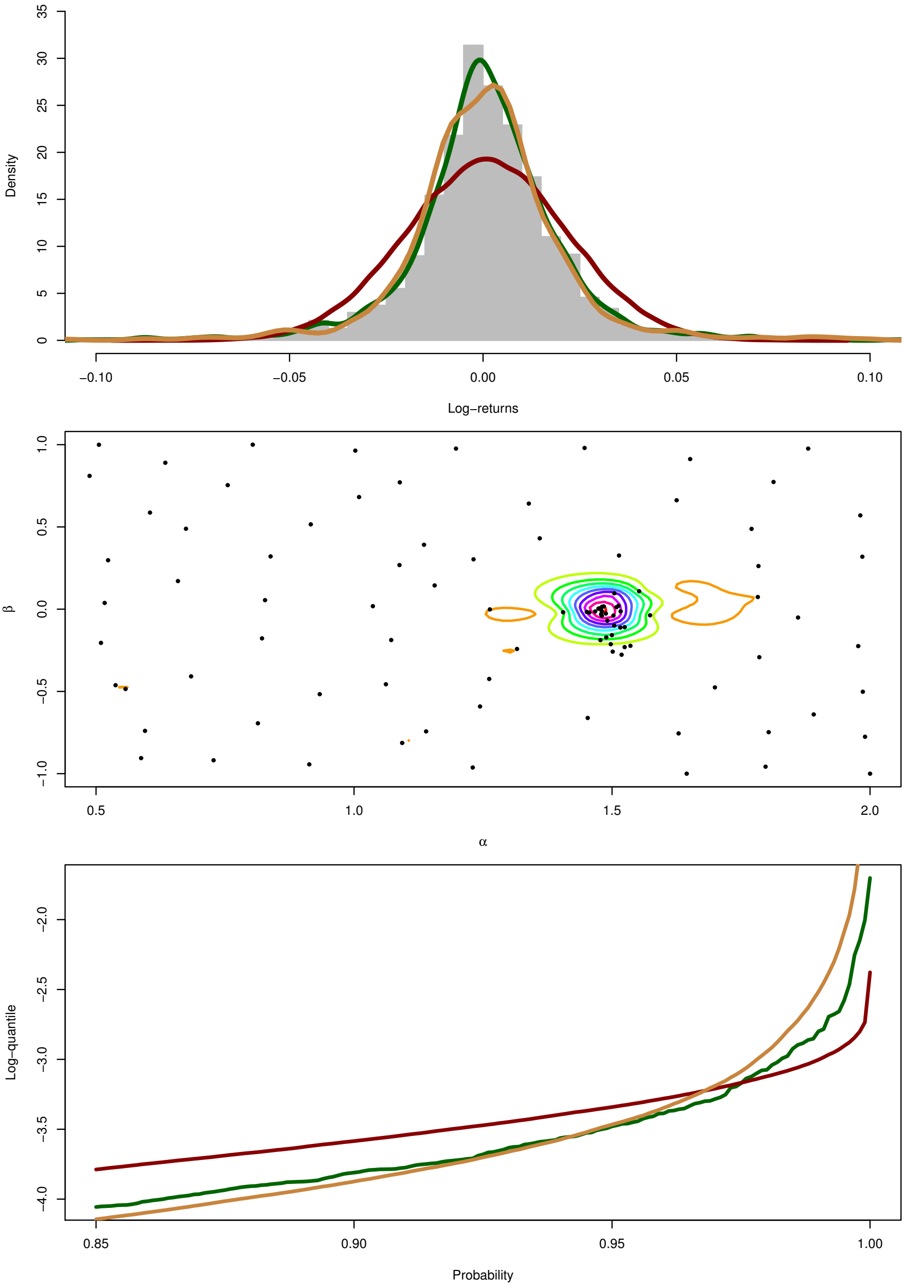

Figure 2: *The histogram and kernel density estimate (upper) in green together with the Gaussian approximation (red) and the fitted α-stable distribution (orange). The estimated parameter log-posterior distribution (middle) of α-stable model of the Google log-return data and the log-quantiles of the three density estimates (lower).*

location parameter η. for this, we can apply Algorithm 2 and replace the SMC-ABC method with a *standard ABC* solution to infer the parameters. That is, we simulate $N = 1\,000$ realisations of the α-stable distribution given the current parameters θ_k and compute

$$\rho(\theta_k) = \sum_{i=1}^{N} \mathbb{I}\Big[\big|\mathcal{S}(y_{1:T}, u_{1:T,i})\big| \leq \epsilon\Big],$$

where we simulate $u_{1:T,i} \sim \mathcal{A}(\theta_k)$ and $y_{1:T}$ denotes the recorded observations. Here, $\mathcal{S}(\,\cdot\,)$ denotes the McCulloch quantile statistics (see Appendix A), which are a near-sufficient statistic for the α-stable distribution. We replace the estimated log-posterior in Algorithm 2 by $\rho(\theta_k)$ and execute the remainder of the algorithm. In the following analysis, we fix the parameter $\gamma = 0.01$ to simplify the problem and use an uniform parameter prior over $\alpha \in [0.5, 2]$ and $\beta \in [-1, 1]$. Here, we use jittering $\sigma_\xi = 0.025$ and precision $\epsilon = 0.10$.

In this problem, the observations are $T = 2\,358$ log-returns of the Google stock at NASDAQ during the period 2004-08-19 – 2013-12-09. We present this data in Figure 1, as a time series (upper) and as the log-returns (middle). The occasional spikes in the latter indicates that the data is distributed according to some other distribution that have somewhat heavier tails than the Gaussian distribution. The heavy tails are also captured in the QQ-plot (lower right), where we see large deviations from that of a Gaussian distribution in the tail behaviour. This is a well-known fact for financial data and by estimating the parameters in a α-stable distribution, we can quantify the behaviour of the tails.

The results from the analysis are presented in Figure 2, where the estimate of the log-posterior distribution is shown (middle). The black dots indicate where the log-posterior is sampled. The algorithm places most samples around the mode which stands out from the surrounding log-posterior distribution. This means that the MAP estimate is a suitable choice in this setting (as the log-posterior is unimodular) and it is estimated as $\widehat{\theta}_{\text{GPO}} = \{1.49, 0.01\}$. As the Gaussian distribution has the parameters $\{2, 0\}$, we conclude that this data has heavier tails than the Gaussian distribution allows for.

In the upper part of Figure 2, we present the histogram of the data with kernel density estimate (blue), the best fit of a Gaussian distribution (red) and the estimated distribution of the α-stable distribution (green). The latter is computed by simulating from the α-stable distribution with the estimated parameters and then computing a kernel density estimate. We see that the estimated distribution parameters fits the data pretty well capturing the overall structure, especially the tails. The tails are compared in log scale in the lower plot. We see that the tails of the α-stable distribution (green) fits the sample quantiles of the data (blue) quite well compared with the Gaussian approximation.

6.2 Linear Gaussian model

Consider the scalar linear Gaussian state space (LGSS) model,

$$x_{t+1}|x_t \sim \mathcal{N}\Big(x_{t+1}; \phi x_t, \sigma_v^2\Big),$$
$$y_t|x_t \sim \mathcal{N}\Big(y_t; x_t, 1\Big),$$

where the parameters are $\theta = \{\phi, \sigma_v\}$. We simulate $T = 1\,000$ data points using the *true* parameters $\theta^\star = \{0.5, 1\}$. For the inference, we use $N = 4\,000$ particles and *smooth ABC* with $\epsilon = 0.1$. We use an uniform prior distribution over $|\phi| < 1$ and $\sigma_v \in [0, 2]$ to insure a stable system and positive standard deviation. We run $K = 150$ iterations and no jittering of parameters, i.e. $\sigma_\xi = 0$. The resulting parameter estimate is obtained as $\widehat{\theta}_{\text{GPO}} = \{0.52, 0.95\}$.

In the upper part of Figure 3, we present a contour plot of the estimated parameter log-posterior as modelled by the Gaussian process. The black dots indicate where the algorithm samples the parameter log-posterior. The dotted lines and red star indicate the parameters of the model from which we simulated the data and the estimated parameters, respectively. The proposed method mainly focus the samples around the peak with only a few samples spread out over the remaining part of the log-posterior. In the lower part of Figure 3, we see that the sampling stabilises quickly and is concentrated around the true log-posterior mode. From these results, we conclude the the proposed method gives accurate parameter estimates using only a few samples of the log-posterior. We also conclude that the acquisition rule balances the trade-off between exploration and exploitation well in this problem.

6.3 Stochastic volatility model with α-stable returns

Consider a stochastic volatility model with symmetric α-stable returns (SVα) (Hull and White, 1987; Casarin, 2004),

$$x_{t+1}|x_t \sim \mathcal{N}\Big(x_{t+1}; \mu + \phi x_t, \sigma_v^2\Big),$$
$$y_t|x_t \sim \mathcal{A}\Big(\alpha, 0, \exp(x_t/2), 0\Big),$$

where the parameters are $\theta = \{\mu, \phi, \sigma_v, \alpha\}$. We estimate the parameters in this model using daily GBP-DEM exchange rates between January 1, 1987 and December 31, 1995 with $T = 3\,827$ samples. We calculate the log-returns by $r_t = 100 \ln(o_{t+1}/o_t)$, where $o_{1:T}$ denotes the original data sequence. The observations $y_{1:T}$ are the residuals from an AR(1) process fitted to the data set $r_{1:T}$. This is done in accordance to Lombardi and Calzolari (2009) and Yildirim et al. (2013) to be able to compare the estimated parameters. The resulting time series $r_{1:T}$ is presented in the upper part of Figure 4. An alternative would be to include a constant term and y_{t-1} into the measurement equation, i.e. rewrite it as $y_t|x_t \sim \mathcal{A}(\alpha, 0, \exp(x_t/2), c - y_{t-1})$ where c denotes a constant mean and y_t the log-return at time t. The resulting parameter inference problem would then include c as a parameter, i.e. $\theta = \{\mu, \phi, \sigma_v, c, \alpha\}$.

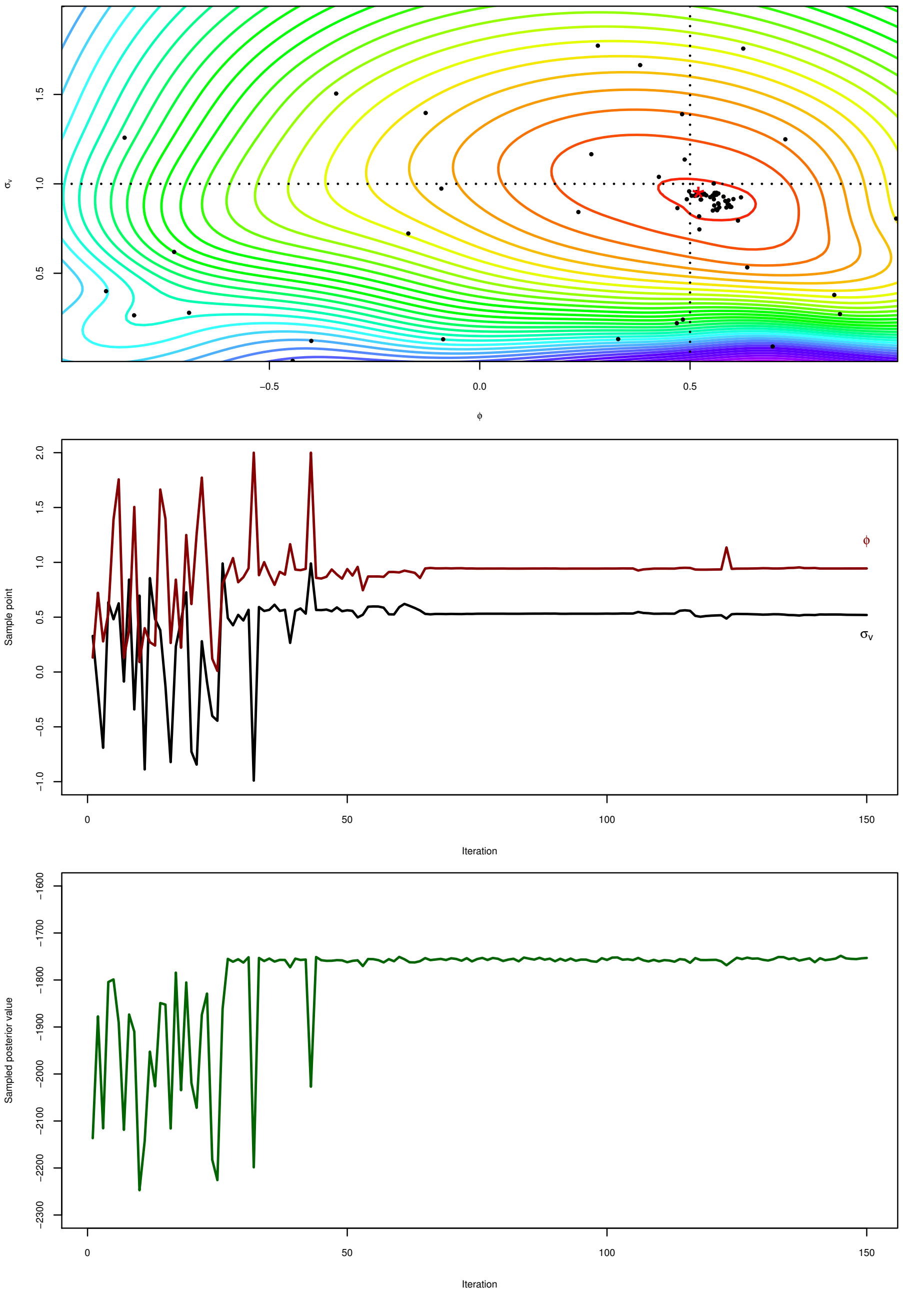

Figure 3: *Upper: the estimated parameter log-posterior of the LGSS model obtain by the proposed method. The MAP estimate (red star), the true parameters (dotted lines) and sample points (black dots) are also indicated. Middle: the parameters in which the proposed method sampled the log-posterior distribution. Lower: the estimated value of the log-posterior distribution.*

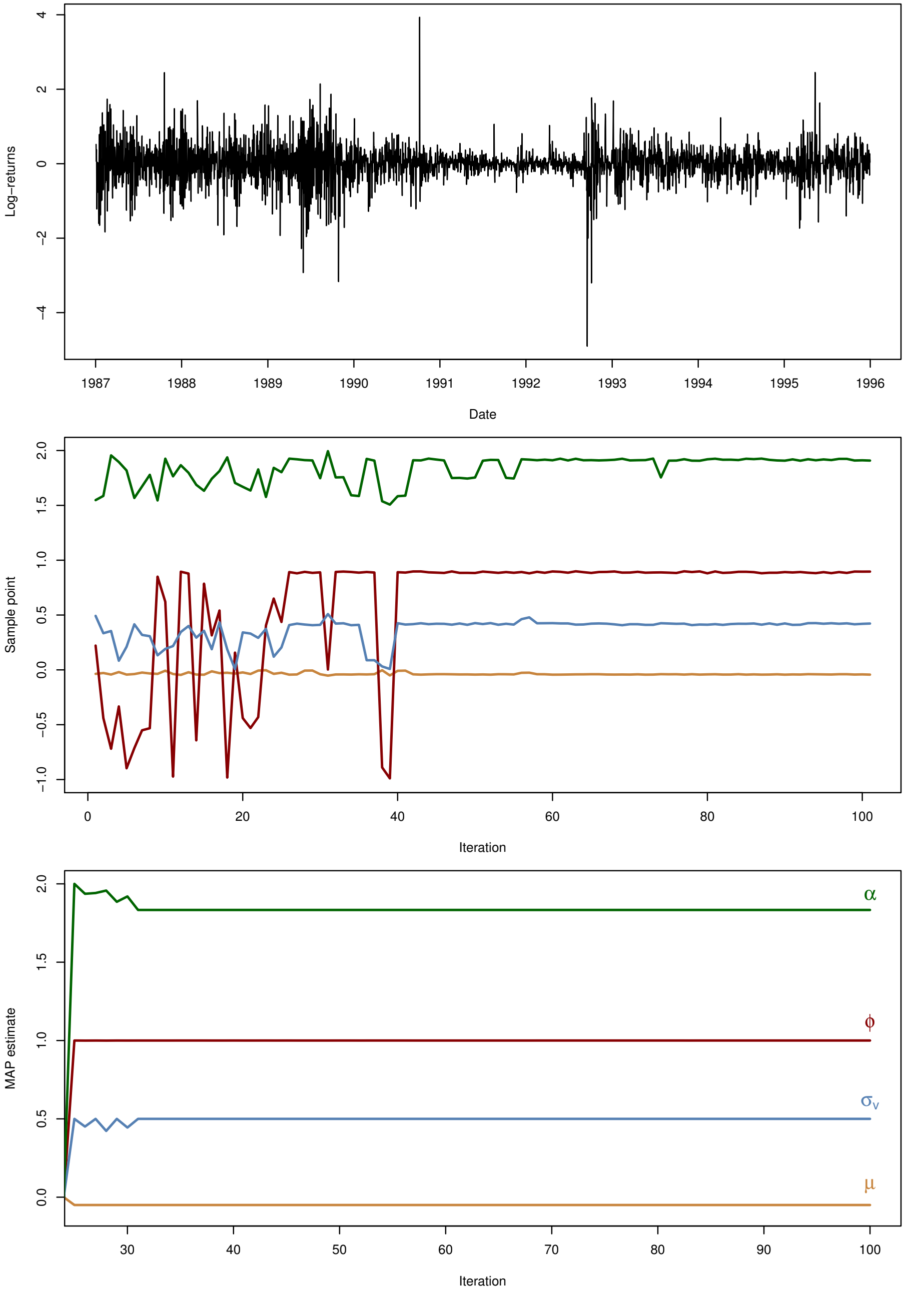

Figure 4: *Upper: the detrended log-returns of the exchange rate between the GBP and DEM during the years 1987 to 1996. Middle: parameters for which the proposed method samples the log-posterior. Lower: The MAP estimate at each the iteration.*

Method	$\widehat{\mu}$	$\widehat{\phi}$	$\widehat{\sigma}_v$	$\widehat{\alpha}$
GPO-SMC	-0.05	0.99	0.50	1.83
Indirect estimation (Lombardi and Calzolari, 2009)	-0.01	0.99	0.01	1.80
Gradient-based search (Yildirim et al., 2013)	-0.01	0.99	0.14	1.76

Table 1: *The parameter estimate in the αSV using three different methods.*

We use the proposed method in Algorithm 2 with *smooth ABC* using $\epsilon = 0.10$, $K = 500$ and jittering $\sigma_\xi = \{0.005, 0.025, 0.025, 0.025\}$ for the four parameters, respectively. We also use an uniform parameter prior over $\mu \in [-0.05, 0]$, $\phi \in [-1, 1]$, $\sigma \in [0, 2]$ and $\alpha \in [1.5, 2]$. We initiate the algorithm by sampling 25 parameters from the parameter prior. This is done to be able to estimate the hyperparameters of the GP and to get an overview of the log-posterior.

In Figure 4, we present the sampling points selected by the algorithm (middle) and the MAP estimate (lower) at each iteration. We note the randomly sampled parameters up to iteration 25, after this the algorithm focus the sampling to a smaller part of the parameter space. The convergence is quick and the parameter estimate has stabilised after about 60 iterations.

The final MAP estimate and the estimates from Lombardi and Calzolari (2009) and Yildirim et al. (2013) are presented in Table 1. Our parameter estimate is close to the other two previously presented estimates in the ϕ and α-parameters. The parameters μ and σ are somewhat larger than in the previously communicated results. This difference could be due to the design of the kernel and mean function used in the GPO part of the proposed method.

7 Conclusions and outlook

In this work, we have examined the potential of ABC to infer parameters using GPO in nonlinear state space models with intractable likelihoods. We have discussed the GPO algorithm and the ABC-SMC method for estimating the intractable likelihood. The proposed algorithm performs well on the three numerical illustrations that are presented in this work. The algorithm converges quickly and also gives reasonable estimates of the parameters, which indicates that it can be an interesting alternative to the existing ML estimation methods. Such solutions, albeit online in nature, requires in the order of thousands of samples from the intractable likelihood function, whereas our algorithm requires an order of magnitude less.

Future work includes more comparisons between the proposed method and the existing methods based on gradient-based optimisation (Ehrlich et al., 2012; Yildirim et al., 2013) and Markov chain Monte Carlo sampling (Peters et al., 2012). GPO can also be used for other optimisation problems, e.g. input design in which e.g. the expected information matrix of an nonlinear SSM is maximised by selecting a suitable input. The GPO algorithm can also be further developed by constructing new acquisition rules and improving the SMC algorithm used to sample the likelihood.

Bibliography

M. A. Beaumont, J-M. Cornuet, J-M. Marin, and C. P. Robert. Adaptive approximate Bayesian computation. *Biometrika*, 96(4):983–990, 2009.

P. Boyle. *Gaussian processes for regression and optimisation*. PhD thesis, Victoria University of Wellington, 2007.

E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *Pre-print*, 2010. arXiv:1012.2599v1.

R. Casarin. Bayesian inference for generalised Markov switching stochastic volatility models, 2004. CEREMADE Journal Working Paper 0414.

J. M. Chambers, C. L. Mallows, and B. Stuck. A method for simulating stable random variables. *Journal of the American Statistical Association*, 71(354): 340–344, 1976.

S. Chib, F. Nardari, and N. Shephard. Markov chain Monte Carlo methods for stochastic volatility models. *Journal of Econometrics*, 108(2):281–316, 2002.

J. Dahlin and F. Lindsten. Particle filter-based Gaussian process optimisation for parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication).

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

E. Ehrlich, A. Jasra, and N. Kantas. Static Parameter Estimation for ABC Approximations of Hidden Markov Models. *Pre-print*, 2012. arXiv:1210.4683v1.

E. F. Fama. The behavior of stock-market prices. *The journal of Business*, 38(1): 34–105, 1965.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proceedings of Radar and Signal Processing*, 140(2):107–113, 1993.

J. Hull and A. White. The pricing of options on assets with stochastic volatilities. *The Journal of Finance*, 42(2):281–300, 1987.

E. Jacquier, N. G. Polson, and P. E. Rossi. Bayesian analysis of stochastic volatility models with fat-tails and correlated errors. *Journal of Econometrics*, 122(1):185–212, 2004.

A. Jasra, S. S. Singh, J. S. Martin, and E. McCoy. Filtering via approximate Bayesian computation. *Statistics and Computing*, 22(6):1223–1237, 2012.

D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

D. J. Lizotte. *Practical Bayesian optimization*. PhD thesis, University of Alberta, 2008.

M. J. Lombardi and G. Calzolari. Indirect estimation of α-stable stochastic volatility models. *Computational Statistics & Data Analysis*, 53(6):2298–2308, 2009.

J-M. Marin, P. Pudlo, C. P. Robert, and R. Ryder. Approximate Bayesian Computational methods. *Pre-print*, 2011. arXiv:1101.0955v2.

J. H. McCulloch. Simple consistent estimators of stable distribution parameters. *Communications in Statistics-Simulation and Computation*, 15(4):1109–1136, 1986.

J. Nolan. *Stable distributions: models for heavy-tailed data*. Birkhauser, 2003.

J. Owen, D. J. Wilkinson, and C. S. Gillespie. Scalable Inference for Markov Processes with Intractable Likelihoods. *Pre-print*, 2014. arXiv:1403.6886v1.

G. W. Peters, S. A. Sisson, and Y. Fan. Likelihood-free Bayesian Inference for α-stable Models. *Comput. Stat. Data Anal.*, 56(11):3743–3756, November 2012.

J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman. Population growth of human Y chromosomes: a study of Y chromosome microsatellites. *Molecular Biology and Evolution*, 16(12):1791–1798, 1999.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pages 2951–2959. Curran Associates, Inc., 2012.

S. Yildirim, S. S. Singh, T. Dean, and A Jasra. Parameter Estimation in Hidden Markov Models with Intractable Likelihoods Using Sequential Monte Carlo. *Preprint*, 2013. arXiv:1311.4117v1.

Appendix

A α-stable distributions

This appendix summarises some important results regarding α-stable distributions. We discuss the properties of a stable distribution, the parametrisation used in this work, how to simulate from the distribution and some near sufficient statistics. For a more detailed presentation, see Nolan (2003), Peters et al. (2012) and references therein.

A.1 Definitions

We start by defining an α-stable distribution in Definition 1. From its definition, we see that the Gaussian distribution is a special case of the α-stable distribution. Also, the Cauchy and Lévy distributions are other special cases of this family.

1 Definition (stable distribution). A non-degenerate random variable X is *stable* if and only if for all $n > 1$, there exist constants $c_n > 0$ and $d_n \in \mathbb{R}$ such that

$$X_1 + X_2 + \ldots + X_n = c_n X + d_n,$$

where $X_1, X_2, \ldots, X_n$ are independent, identical copies of X. Furthermore, X is *strictly stable* if $d_n = 0$ for every n.

The general family of α-stable distributions can be described by the characteristic function $\varphi_X(t) = \mathbb{E}[\exp(itX)]$ for a real valued random variable X. Except for the three members previously mentioned, the probability distribution function (pdf) cannot be expressed in closed-form. This as the Fourier transform of the characteristic function cannot be evaluated in these cases. Instead, we generally work with the characteristic function of the distribution family. Two different common parametrisations of the α-stable distribution are presented in Definitions 2 and 3.

2 Definition (α-stable distribution, parametrisation 0). An univariate α-stable distribution denoted by $\mathcal{A}(\alpha, \beta, \gamma, \eta)$ has the characteristic function

$$\phi(t|\theta) = \begin{cases} \exp\left\{i\eta t - \gamma^{\alpha}|t|^{\alpha}\left[1 + i\beta \tan\left(\frac{\pi\alpha}{2}\right)\mathsf{sgn}(t)\left(|\gamma t|^{1-\alpha} - 1\right)\right]\right\} & \text{if } \alpha \neq 1, \\ \exp\left\{i\eta t - \gamma|t|\left[1 + \frac{2i\beta}{\pi}\mathsf{sgn}(t)\ln\left(\gamma|t|\right)\right]\right\} & \text{if } \alpha = 1, \end{cases}$$

where $\alpha \in [0, 2]$ denotes the stability parameter, $\beta \in [-1, 1]$ denotes the skewness parameters, $\gamma \in \mathbb{R}_+$ denotes the scale parameter and $\eta \in \mathbb{R}$ denotes the location parameter.

3 Definition (α-stable distribution, parametrisation 1). An univariate α-stable distribution denoted by $\mathcal{A}(\alpha, \beta, \gamma, \eta)$ has the characteristic function

$$\phi(t|\theta) = \begin{cases} \exp\left\{i\eta t - \gamma^{\alpha}|t|^{\alpha}\left[1 - i\beta \tan\left(\frac{\pi\alpha}{2}\right)\mathsf{sgn}(t)\right]\right\} & \text{if } \alpha \neq 1, \\ \exp\left\{i\eta t - \gamma|t|\left[1 + \frac{2i\beta}{\pi}\mathsf{sgn}(t)\ln(t)\right]\right\} & \text{if } \alpha = 1, \end{cases}$$

where $\alpha \in [0,2]$ denotes the stability parameter, $\beta \in [-1,1]$ denotes the skewness parameters, $\gamma \in \mathbb{R}_+$ denotes the scale parameter and $\eta \in \mathbb{R}$ denotes the location parameter.

The parametrisation in Definition 2 is referred to as the zero-parametrisation in Nolan (2003). This parametrisation is preferred since to that it results in a simple characteristic function and is continuous in all the parameters. The parametrisation in Definition 3 is referred to as the one-parametrisation and is useful in algebraic evaluations. In this work, we use the one-parametrisation as it is easy to simulate from.

As previously mentioned, there exists three special cases in which the characteristic function can be inverted to obtain the PDF. The Gaussian distribution $\mathcal{N}(\eta, c^2)$ is recovered by using $\{\alpha,\beta,\gamma,\eta\} = \{2,0,\gamma,c\}$, the Cauchy distribution $\mathcal{C}(\eta,c)$ by $\{1,0,c,\eta\}$ and the Lévy distribution $\mathcal{L}(\eta,c)$ by $\{1/2,1,c,\eta\}$. Here, we use the zero-parametrisation in Definition 2 with η and c denoting the location and scale parameter, respectively.

Even if the PDF cannot be written down analytically, it can be estimated by numerical integration. In Figure 5, we present the PDF for different values of the parameters α and β, keeping the other parameters fixed. We see that the α-stable distribution can capture both skewed distributions and heavy tails. This is why they have been used in finance (Lombardi and Calzolari, 2009; Fama, 1965) to model returns and stock prices.

A.2 Simulating random variables

Even if the pdf often is intractable for α-stable distributions, it is quite simple to simulate from them using results from Chambers et al. (1976). In Propositions 4 and 5, we present methods for simulating random variables from the two parametrisations of the distribution.

4 Proposition (Simulating α-stable variable, parametrisation 0). *Assume that we can simulate $w \sim \mathsf{Exp}(1)$ and $u \sim \mathcal{U}(-\pi/2,\pi/2)$. Then, we can obtain a sample from $\mathcal{A}(\alpha,\beta,1,0)$ by*

$$\bar{y} = \begin{cases} S_{\alpha,\beta}\frac{\sin\left[\alpha(u+B_{\alpha,\beta})\right]}{(\cos u)^{\alpha/2}}\left[\frac{\cos\left[u-\alpha(u+B_{\alpha,\beta})\right]}{w}\right]^{\frac{1-\alpha}{\alpha}} & \textit{if } \alpha \neq 1, \\ \frac{2}{\pi}\left[\left(\frac{\pi}{2}+\beta u\right)\tan(u) - \beta\log\frac{\frac{\pi}{2}w\cos u}{\frac{\pi}{2}+\beta u}\right] & \textit{if } \alpha = 1, \end{cases} \tag{15}$$

where we have introduced the following notation

$$S_{\alpha,\beta} = \left[1+\beta^2\tan^2\left(\frac{\pi\alpha}{2}\right)\right]^{-\frac{1}{2\alpha}},$$
$$B_{\alpha,\beta} = \frac{1}{\alpha}\arctan\left(\beta\tan\left(\frac{\pi\alpha}{2}\right)\right).$$

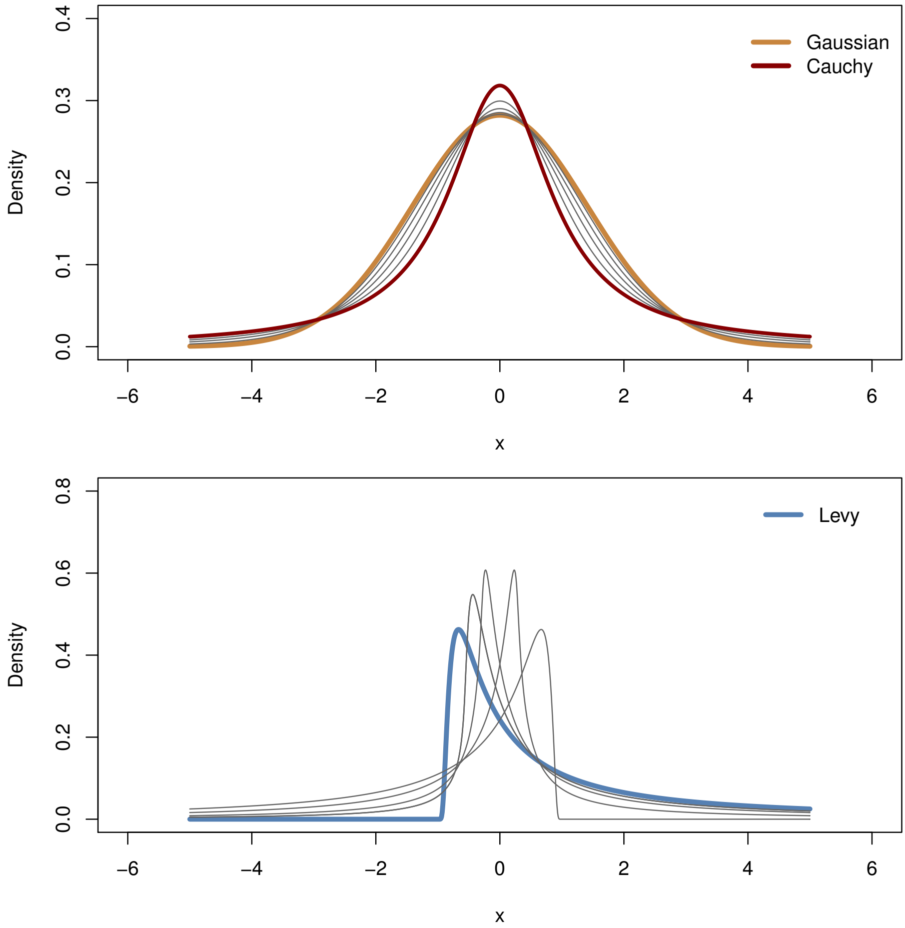

Figure 5: *Estimated probability density functions for α-stable distributions when varying the stability parameter $\alpha \in [0, 2]$ (upper) and the skewness parameter $\beta \in [-1, 1]$ (lower). We use $\beta = 0$ when varying α and $\alpha = 0.5$ when varying β. The location and scale parameters are kept fixed at $\{\gamma, \eta\} = \{1, 0\}$. Here, we use the zero-parametrisation in Definition 2 of the α-stable distribution.*

A sample from $\mathcal{A}(\alpha, \beta, \gamma, \eta)$ is obtained by the transformation

$$y = \begin{cases} \gamma\left(\bar{y} - \beta \tan(\frac{\pi\alpha}{2})\right) + \eta & \text{if } \alpha \neq 1, \\ \gamma \bar{y} + \eta & \text{if } \alpha = 1. \end{cases}$$

5 Proposition (Simulating α-stable variable, parametrisation 1). *Assume that we can simulate $w \sim \mathsf{Exp}(1)$ and $u \sim \mathcal{U}(-\pi/2, \pi/2)$. Then, we can obtain a sample from $\mathcal{A}(\alpha, \beta, 1, 0)$ by*

$$\bar{y} = \begin{cases} \frac{\sin\left[\alpha(u + T_{\alpha,\beta})\right]}{(\cos(\alpha T_{\alpha,\beta})\cos(u))^{1/\alpha}} \left[\frac{\cos\left[\alpha T_{\alpha,\beta} + (\alpha-1)u\right]}{w}\right]^{\frac{1-\alpha}{\alpha}} & \text{if } \alpha \neq 1, \\ \frac{2}{\pi}\left[\left(\frac{\pi}{2} + \beta u\right)\tan(u) - \beta \log \frac{\frac{\pi}{2} w \cos u}{\frac{\pi}{2} + \beta u}\right] & \text{if } \alpha = 1, \end{cases} \tag{16}$$

where we have introduced the following notation

$$T_{\alpha,\beta} = \frac{1}{\alpha} \arctan\left(\beta \tan\left(\frac{\pi\alpha}{2}\right)\right).$$

A sample from $\mathcal{A}(\alpha, \beta, \gamma, \eta)$ is obtained by the transformation

$$y = \begin{cases} \gamma \bar{y} + \eta & \text{if } \alpha \neq 1, \\ \gamma \bar{y} + \left(\eta + \beta \frac{2}{\pi} \gamma \log \gamma\right) & \text{if } \alpha = 1. \end{cases}$$

A.3 Parameter estimation

The moments of α-stable distributions does not always exist for all parameters. For example, the mean exists if $\alpha > 1$ but not otherwise. Also the variance is $2\gamma^2$ if $\alpha = 2$, but does not exist otherwise. This makes parameter estimation difficult in the general case using the usual sample statistics for mean and variance.

One approach to estimate the parameters in the distribution is presented in McCulloch (1986) and is based on sample quantiles of the data. These statistics are used in combination with tabled values to estimate the parameters of the distribution. The various sample statistics are

$$\widehat{\nu}_\alpha = \frac{\widehat{q}_{95}(x) - \widehat{q}_5(x)}{\widehat{q}_{75}(x) - \widehat{q}_{25}(x)}, \quad \widehat{\nu}_\beta = \frac{\widehat{q}_{95}(x) + \widehat{q}_5(x) - 2\widehat{q}_{50}(x)}{\widehat{q}_{95}(x) - \widehat{q}_5(x)}, \quad \widehat{\nu}_\gamma = \frac{\widehat{q}_{75}(x) - \widehat{q}_{25}(x)}{\gamma},$$

where $\widehat{q}_k(x)$ denotes the k sample quantile of the data x. The location parameter η can be estimated using a similar expression with the estimates of the other parameters. As previously mentioned, these statistics can be used with the tables in McCulloch (1986) to estimate the parameters of an α-stable distribution. The statistics are also useful as near-sufficient statistics in ABC-based algorithms.

Paper D

A graph/particle-based method for experiment design in nonlinear systems

Authors: P. E. Valenzuela, J. Dahlin, C. R. Rojas and T. B. Schön

Supported by the project *Probabilistic modelling of dynamical systems* (Contract number: 621-2013-5524) funded by the Swedish Research Council.

Edited version of the paper:

> P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. A graph/particle-based method for experiment design in nonlinear systems. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication).

A graph/particle-based method for experiment design in nonlinear systems

P. E. Valenzuela[⋆], J. Dahlin[†], C. R. Rojas[⋆] and T. B. Schön[‡]

[⋆]Dept. of Automatic Control,
KTH, Royal Institute of Technology,
SE-100 44 Stockholm, Sweden.
{pva,crro}@kth.se

[†]Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden.
johan.dahlin@isy.liu.se

[‡]Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
thomas.schon@it.uu.se

Abstract

We propose an extended method for experiment design in nonlinear state space models. The proposed input design technique optimizes a scalar cost function of the information matrix, by computing the optimal stationary probability mass function (pmf) from which an input sequence is sampled. The feasible set of the stationary pmf is a polytope, allowing it to be expressed as a convex combination of its extreme points. The extreme points in the feasible set of pmf's can be computed using graph theory. Therefore, the final information matrix can be approximated as a convex combination of the information matrices associated with each extreme point. For nonlinear systems, the information matrices for each extreme point can be computed by using particle methods. Numerical examples show that the proposed technique can be successfully employed for experiment design in nonlinear systems.

1 Introduction

Experiment design deals with the generation of an input signal that maximizes the information retrieved from an experiment. Some of the initial contributions are discussed in Cox (1958) and Goodwin and Payne (1977). Since then, many contributions to the subject have been developed; see e.g. Fedorov (1972), Whittle (1973), Hildebrand and Gevers (2003), Hildebrand and Gevers (2003) and the references therein.

In this article, a new method for experiment design in nonlinear systems is presented, which extends the input design methods proposed in Gopaluni et al. (2011) and Valenzuela et al. (2013). The objective is to design an experiment as a realization of a stationary process, such that the system is identified with maximum accuracy as defined by a scalar function of the Fisher information matrix, and under the assumption that the input can adopt a finite set of values. The assumption on the input class modifies the class of input sequences considered in Gopaluni et al. (2011). The optimization of the stationary probability mass function (pmf) is done by maximizing a scalar cost function of the information matrix over the feasible set of pmf's.

Using concepts from graph theory (Zaman, 1983; Johnson, 1975; Tarjan, 1972), we can express the feasible set of pmf's as a convex combination of the measures for the extreme points of the set. Therefore, the information matrix corresponding to a feasible pmf can be expressed as the convex combination of the information matrices associated with the extreme points of the feasible set. Since the exact computation of the information matrices for nonlinear systems is often intractable, we use particle methods to compute sampled information matrices for the extreme points of the feasible set. This allows us to extend the technique of Valenzuela et al. (2013) to more general nonlinear model structures. An attractive property of the method is that the optimization problem is convex even for nonlinear systems. In addition, since the input is restricted to a finite set of possible values, the method can naturally handle amplitude limitations.

Previous results on input design have mostly been concerned with linear systems. A Markov chain approach to input design is presented in Brighenti et al. (2009), where the input is modelled as the output of a Markov chain. Suzuki and Sugie (2007) presents a time domain experiment design method for system identification. Linear matrix inequalities (LMI) are used to solve the input design problem in Jansson and Hjalmarsson (2005) and Lindqvist and Hjalmarsson (2000). A robust approach for input design is presented in Rojas et al. (2007), where the input signal is designed to optimize a cost function over a set where the true parameter is assumed to lie.

In recent years, the interest in input design for nonlinear systems has increased. The main problem here is that the frequency domain approach for experiment design used in linear systems is no longer valid. An analysis of input design for nonlinear systems using the knowledge of linear systems is considered in Hjalmarsson and Mårtensson (2007). In Larsson et al. (2010) an input design method for a particular class of nonlinear systems is presented.

Input design for structured nonlinear systems is discussed in Vincent et al. (2009). Gopaluni et al. (2011) introduces a particle filter method for input design in nonlinear systems. An analysis of input design for a class of Wiener systems is considered in Cock et al. (2013). A graph theory approach for input design for output-error like nonlinear system is presented in Valenzuela et al. (2013). The results presented allow to design input signals when the system contains nonlinear functions, but the restrictions on the system dynamics and/or the input structure are the main limitations of most of the previous contributions. Moreover, with the exception of Brighenti et al. (2009), Larsson et al. (2010) and Valenzuela et al. (2013), the proposed methods cannot handle amplitude limitations on the input signal, which could arise due to physical and/or safety reasons.

2 Problem formulation

In this article, the objective is to design an input signal $u_{1:n_{\text{seq}}} := \{u_t\}_{t=1}^{n_{\text{seq}}}$, as a realization of a stationary process. This is done such that a state space model (SSM) can be identified with maximum accuracy as defined by a scalar function of the Fisher information matrix $\mathcal{I}_F$ (Ljung, 1999). An SSM with states $x_{1:T} := \{x_t\}_{t=1}^{T}$, inputs $u_{1:T}$ and measurements $y_{1:T}$ is given by

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

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

Here, $f_\theta(\,\cdot\,)$ and $g_\theta(\,\cdot\,)$ denote known probability distributions parametrised by $\theta \in \Theta \subset \mathbb{R}^d$. For the remainder of this article, we make the rather restrictive albeit standard assumption that we know the initial state x_0 and the true model structure (1) with true parameters θ_0. Hence, we can write the joint distribution of states and measurements for (1) as

$$p_\theta(x_{1:T}, y_{1:T} | u_{1:T}) = \prod_{t=1}^{T} f_\theta(x_t | x_{t-1}, u_{t-1}) g_\theta(y_t | x_t, u_t). \tag{2}$$

This quantity is used in the sequel for estimating $\mathcal{I}_F$ by

$$\mathcal{I}_F := \mathbf{E}_\theta \left\{ \mathcal{S}(\theta_0) \mathcal{S}^\top(\theta_0) \right\}, \tag{3a}$$

$$\mathcal{S}(\theta_0) := \nabla \log l_\theta(y_{1:n_{\text{seq}}}) \big|_{\theta=\theta_0}, \tag{3b}$$

where $l_\theta(y_{1:n_{\text{seq}}})$ and $\mathcal{S}(\theta)$ denote the likelihood function and the score function, respectively. Note, that the expected value in (3a) is with respect to the stochastic processes in (1) and the realizations of $u_{1:n_{\text{seq}}}$.

We note that (3a) depends on the cumulative density function (cdf) of $u_{1:n_{\text{seq}}}$, say $P_u(u_{1:n_{\text{seq}}})$. Therefore, the input design problem is to find a cdf $P_u^{\text{opt}}(u_{1:n_{\text{seq}}})$ which optimizes a scalar function of (3a). We define this scalar function as $h_m : \mathbb{R}^{d \times d} \to \mathbb{R}$. To obtain the desired results, h_m must be a nondecreasing matrix function (Boyd and Vandenberghe, 2004, pp. 108). Different choices of h_m have been proposed in the literature, see e.g. Rojas et al. (2007); some examples are $h_m = \det$, and $h_m = -\operatorname{tr}\{(\ \cdot\ )^{-1}\}$. In this work, we leave the selection of h_m to the user.

Since $P_u^{\text{opt}}(u_{1:n_{\text{seq}}})$ has to be a stationary cdf, the optimization must be constrained to the set

$$\mathcal{P} := \Bigg\{ P_u : \mathbb{R}^{n_{\text{seq}}} \to \mathbb{R} |\, P_u(\mathbf{x}) \geq 0,\ \forall \mathbf{x} \in \mathbb{R}^{n_{\text{seq}}}; P_u \text{ is monotone non-decreasing;}$$

$$\lim_{\substack{x_i \to \infty \\ i=\{1,\dots,n_{\text{seq}}\} \\ \mathbf{x}=(x_1,\dots,x_{n_{\text{seq}}})}} P_u(\mathbf{x}) = 1; \int_{v\in\mathbb{R}} dP_u(v,\, \mathbf{z}) = \int_{v\in\mathbb{R}} dP_u(\mathbf{z},\, v)\,, \forall \mathbf{z} \in \mathbb{R}^{n_{\text{seq}}-1} \Bigg\}. \quad (4)$$

The last condition in (4) (with slight abuse of notation) guarantees that $P_u \in \mathcal{P}$ is the cdf of a stationary sequence (Zaman, 1983).

To simplify our analysis, we will assume that u_t can only adopt a finite number c_{seq} of values. We define this set of values as $\mathcal{C}$. With the previous assumption, we can define the following subset of $\mathcal{P}$:

$$\mathcal{P}_{\mathcal{C}} := \Bigg\{ p_u : \mathcal{C}^{n_{\text{seq}}} \to \mathbb{R} |\, p_u(\mathbf{x}) \geq 0,\ \forall \mathbf{x} \in \mathcal{C}^{n_{\text{seq}}}; \sum_{\mathbf{x}\in\mathcal{C}^{n_{\text{seq}}}} p_u(\mathbf{x}) = 1;$$

$$\sum_{v\in\mathcal{C}} p_u(v,\, \mathbf{z}) = \sum_{v\in\mathcal{C}} p_u(\mathbf{z},\, v)\,, \forall \mathbf{z} \in \mathcal{C}^{(n_{\text{seq}}-1)} \Bigg\}. \quad (5)$$

The set introduced in (5) will constrain the pmf $p_u(u_{1:n_{\text{seq}}})$.

The problem described can be summarized as

1 Problem. Design an optimal input signal $u_{1:n_{\text{seq}}} \in \mathcal{C}^{n_{\text{seq}}}$ as a realization from $p_u^{\text{opt}}(u_{1:n_{\text{seq}}})$, where

$$p_u^{\text{opt}} := \arg \max_{p_u \in \mathcal{P}_{\mathcal{C}}} h_m(\mathcal{I}_F(p_u))\,, \quad (6)$$

with $h_m : \mathbb{R}^{d\times d} \to \mathbb{R}$ a matrix nondecreasing function, and $\mathcal{I}_F \in \mathbb{R}^{d\times d}$ defined as in (3). ■

3 New input design method

In this section, we discuss the proposed input design method, which is based on three steps. In the first step, we calculate basis input signals, which are used to excite the system. In the second step, we iteratively calculate the information matrix estimate and the optimal weighting of the basis inputs in a Monte Carlo setting. In the third step, we generate an optimal input sequence using the estimated optimal weighting of the basis inputs.

3.1 Graph theoretical input design

Problem 1 is often hard to solve explicitly since

(i) we need to represent the elements in $\mathcal{P}_{\mathcal{C}}$ as a linear combination of its basis functions, and

(ii) the stationary pmf p_u is of dimension n_{seq}, where n_{seq} could potentially be very large.

These issues make Problem 1 computationally intractable.

To solve issue (ii), we assume that p_u is an extension from the subspace of stationary pmf's of memory length n_m, where $n_m << n_{\text{seq}}$. To address issue (i), notice that all the elements in $\mathcal{P}_{\mathcal{C}}$ can be represented as a convex combination of its extreme points (Valenzuela et al., 2013). We will refer to $\mathcal{V}_{\mathcal{P}_{\mathcal{C}}} := \{v_1, \ldots, v_{n_{\mathcal{V}}}\}$ as the set of the extreme points of $\mathcal{P}_{\mathcal{C}}$.

To find all the elements in $\mathcal{V}_{\mathcal{P}_{\mathcal{C}}}$, we will make use of graph theory as follows. $\mathcal{C}^{n_m}$ is composed of $(c_{\text{seq}})^{n_m}$ elements. Each element in $\mathcal{C}^{n_m}$ can be viewed as one node in a graph. In addition, the transitions (edges) between the elements in $\mathcal{C}^{n_m}$ are given by the feasible values of u_{k+1} when we move from $(u_{k-n_m+1}, \ldots, u_k)$ to $(u_{k-n_m+2}, \ldots, u_{k+1})$, for all integers $k \geq 0$. Figure 1 illustrates this idea, when $c_{\text{seq}} = 2$, $n_m = 2$, and $\mathcal{C} = \{0, 1\}$. From this figure we can see that, if we are in node $(0, 1)$ at time t, then we can only transit to node $(1, 0)$ or $(1, 1)$ at time $t+1$.

To find all the elements in $\mathcal{V}_{\mathcal{P}_{\mathcal{C}}}$ we rely on the concept of prime cycles. A *prime cycle* is an elementary cycle whose set of nodes do not have a proper subset which is an elementary cycle (Zaman, 1983, pp. 678). It has been proved that the prime cycles of a graph describe all the elements in the set $\mathcal{V}_{\mathcal{P}_{\mathcal{C}}}$ (Zaman, 1983, Theorem 6). In other words, each prime cycle defines one element $v_j \in \mathcal{V}_{\mathcal{P}_{\mathcal{C}}}$. Furthermore, each v_j corresponds to a uniform distribution whose support is the set of elements of its prime cycle, for all $j \in \{1, \ldots, n_{\mathcal{V}}\}$ (Zaman, 1983, pp. 681). Therefore, the elements in $\mathcal{V}_{\mathcal{P}_{\mathcal{C}}}$ can be described by finding all the prime cycles associated with the stationary graph $\mathcal{G}_{\mathcal{C}^{n_m}}$ drawn from $\mathcal{C}^{n_m}$.

It is known that all the prime cycles associated with $\mathcal{G}_{\mathcal{C}^{n_m}}$ can be derived from the elementary cycles associated with $\mathcal{G}_{\mathcal{C}^{(n_m-1)}}$ (Zaman, 1983, Lemma 4), which can be found by using existing algorithms[1]. To illustrate this, we consider the graph depicted in Figure 2. One elementary cycle for this graph is given by $(0, 1, 0)$. Using (Zaman, 1983, Lemma 4), the elements of one prime cycle for the graph $\mathcal{G}_{\mathcal{C}^2}$ are obtained as a concatenation of the elements in the elementary cycle $(0, 1, 0)$. Hence, the prime cycle in $\mathcal{G}_{\mathcal{C}^2}$ associated with this elementary cycle is given by $((0, 1), (1, 0), (0, 1))$.

Since we know the prime cycles, it is possible to generate an input sequence $\{u_t^j\}_{t=0}^{T}$ from v_j, which will be referred to as the *basis inputs*. As an example,

[1] For the examples in Section 4, we have used the algorithm presented in (Johnson, 1975, pp. 79–80) complemented with the one proposed by (Tarjan, 1972, pp. 157).

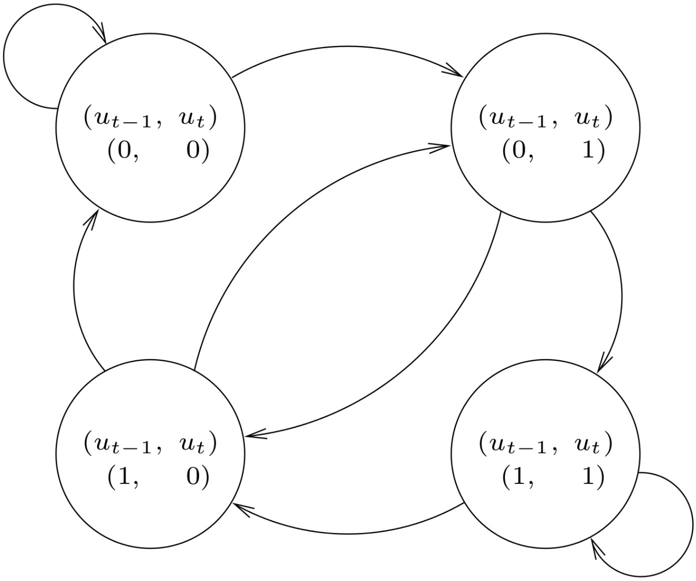

Figure 1: *Example of graph derived from $\mathcal{C}^{n_m}$, with $c_{\text{seq}} = 2$, $n_m = 2$, and $\mathcal{C} := \{0, 1\}$.*

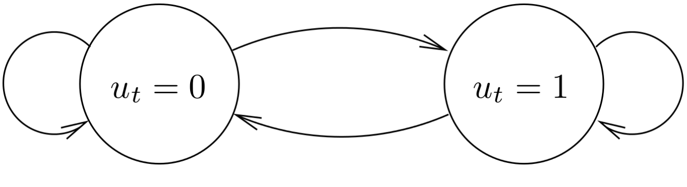

Figure 2: *Example of graph derived from $\mathcal{C}^{n_m}$, with $c_{\text{seq}} = 2$, $n_m = 1$, and $\mathcal{C} := \{0, 1\}$.*

we use the graph depicted in Figure 1. One prime cycle for this graph is given by $((0, 1), (1, 0), (0, 1))$. Therefore, the sequence $\{u_t^j\}_{t=0}^T$ is given by taking the last element of each node, i.e., $\{u_t^j\}_{t=0}^T = \{1, 0, 1, 0, \ldots, ((-1)^T + 1)/2\}$.

Given $\{u_t^j\}_{t=0}^T$, we can use them to obtain the corresponding information matrix for $v_j \in \mathcal{V}_{\mathcal{P}_\mathcal{C}}$, say $\mathcal{I}_F^{(j)}$. However, in general the matrix $\mathcal{I}_F^{(j)}$ cannot be computed explicitly. To overcome this problem, we use Sequential Monte Carlo (SMC) methods to approximate $\mathcal{I}_F^{(j)}$, as discussed in the next subsection.

3.2 Estimation of the score function

SMC methods are a family of methods that can be used e.g. to estimate the filtering and smoothing distributions in SSMs. General introductions to SMC samplers are given in e.g. Doucet and Johansen (2011) and Del Moral et al. (2006). Here, we introduce the *auxiliary particle filter* (APF) (Pitt and Shephard, 1999) and the fixed-lag (FL) particle smoother (Kitagawa and Sato, 2001) to estimate the score function for (1). In the next subsection, the score function estimates are used to estimate the information matrix using (3a).

The APF estimates the smoothing distribution by

$$\widehat{p}_\theta(\mathrm{d}x_{1:t}|y_{1:t}) := \sum_{i=1}^{N} \frac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}} \delta_{x_{1:t}^{(i)}}(\mathrm{d}x_{1:t}), \tag{7}$$

where the particle system is denoted by $\{x_{1:t}^{(i)}, w_t^{(i)}\}_{i=1}^N$. Here, $w_t^{(i)}$ and $x_{1:t}^{(i)}$ denote the weights and the particle trajectories computed by the APF. Here, $\delta_z(\mathrm{d}x_{1:t})$ denotes the Dirac measure at z.

The particle system is sequentially computed using two steps: (i) sampling/propagation and (ii) weighting. The first step can be seen as sampling from a proposal kernel,

$$\{a_t^{(i)}, x_t^{(i)}\} \sim \frac{w_{t-1}^{a_t}}{\sum_{k=1}^{N} w_{t-1}^{(k)}} R_{\theta,t}(x_t|x_{t-1}^{a_t}, u_{t-1}), \tag{8}$$

where we append the sampled particle to the trajectory by $x_{1:t}^{(i)} = \{x_{1:t-1}^{(i)}, x_t^{(i)}\}$. Here, $R_{\theta,t}(\,\cdot\,)$ denotes the propagation kernel and the *ancestor index* $a_t^{(i)}$ denotes the index of the *ancestor* at time $t-1$ of particle $x_t^{(i)}$. In the second step, we calculate the (unnormalised) importance weights,

$$w_t^{(i)} \triangleq \frac{g_\theta(y_t|x_t^{(i)}, u_t) f_\theta(x_t|x_{t-1}^{(i)}, u_{t-1})}{R_{\theta,t}(x_t|x_{t-1}^{(i)}, u_{t-1})}. \tag{9}$$

We make use of *Fisher's identity* (Fisher, 1925; Cappé et al., 2005) to rewrite the score function into a form that can be used in combination with SMC methods.

Algorithm 1 PF for score function estimation

INPUTS: An SSM (1), $y_{1:T}$ (obs.), $u_{1:T}$ (inp.) and N (no. particles).
OUTPUT: $\widehat{\mathcal{S}}(\theta)$ (est. of the score).

```
1: Initialise particles $x_0^{(i)}$ for $i = 1$ to $N$.
2: for $t = 1$ to $T$ do
3:     Resample the particles with weights $\{w_{t-1}^{(i)}\}_{i=1}^N$.
4:     Propagate the particles using $R_{\theta,t}(\,\cdot\,)$.
5:     Compute (9) to obtain $\{w_t^{(i)}\}_{i=1}^N$.
6: end for
7: Compute (10) to obtain $\widehat{\mathcal{S}}(\theta)$.
```

This results in that we can write

$$\nabla \log\, l_\theta(y_{1:T}) = \mathbf{E}_\theta\Big[\nabla \log\, p_\theta(x_{1:T}, y_{1:T}|u_{1:T})\big|y_{1:T}, u_{1:T}\Big],$$

where we insert (2) to obtain the form

$$\nabla \log\, l_\theta(y_{1:T}) = \sum_{t=1}^{T} \int \xi_\theta(x_{t-1:t}) p_\theta(x_{t-1:t}|y_{1:T})\,\mathrm{d}x_{t-1:t},$$

with

$$\xi_\theta(x_{t-1:t}) = \nabla\Big[\log\, f_\theta(x_t|x_{t-1}, u_{t-1}) + \log\, g_\theta(y_t|x_t, u_t)\Big],$$

which depends on the two-step marginal smoothing densities. The APF can be used the estimate these quantities but this leads to poor estimates with high variance, due to problems with *particle degeneracy*.

Instead, we use an FL-smoother to estimate the smoothing densities, which reduces the variance of the score estimates (Olsson et al., 2008). The fixed-lag smoother assumes that

$$p_\theta(x_t|y_{1:T}, u_{1:T}) \approx p_\theta(x_t|y_{1:\kappa_t}, u_{1:\kappa_t}),$$

for $\kappa_t = \min(t + \Delta, T)$ with some fixed-lag Δ. This means that measurements after some time has a negligible effect on the state, see Dahlin et al. (2014) for more details about the FL-smoother and its use for score estimation. The resulting expression is obtained as

$$\widehat{\mathcal{S}}(\theta) := \sum_{t=1}^{T}\sum_{i=1}^{N} w_{\kappa_t}^{(i)} \xi_\theta\Big(x_t^{a_{\kappa_t,t}^{(i)}}, x_{t-1}^{a_{\kappa_t,t-1}^{(i)}}, u_t\Big), \tag{10}$$

where $a_{\kappa_t,t}^{(i)}$ denotes the particle at time t which is the ancestor of particle i at time κ_t. The complete procedure for estimating the score function using the FL smoother is outlined in Algorithm 1.

Algorithm 2 Optimal input estimation using Monte Carlo

INPUTS: Algorithm 1, K (no. MC runs) and M (size of each batch).
OUTPUT: $\gamma^\star$ (est. of the optimal weighting of the basis inputs).

```
for $k = 1$ to $K$ do
    Generate $M$ samples using Algorithm 1 for each basis input.
    Estimate the information matrix by (12) for each basis input.
    Solve the problem in (11).
    Set $\gamma_k$ as the weighting factors obtained from the solver.
end for
Compute the sample mean of $\gamma = \{\gamma_1, \ldots, \gamma_K\}$, denote it as $\gamma^\star$.
```

3.3 Monte Carlo-based optimisation

Given $\{\mathcal{I}_F^{(j)}\}_{j=1}^{n_\mathcal{V}}$ associated with the elements in $\mathcal{V}_{\mathcal{P}_\mathcal{C}}$, we can find the corresponding information matrix associated with any element in $\mathcal{P}_\mathcal{C}$ as a convex combination of the $\mathcal{I}_F^{(j)}$'s. By defining $\gamma := \{\alpha_1, \ldots, \alpha_{n_\mathcal{V}}\} \in \mathbb{R}^{n_\mathcal{V}}$, we introduce $\mathcal{I}_F^{\text{app}}(\gamma)$ as the information matrix associated with one element in $\mathcal{P}_\mathcal{C}$ for a given γ such that $\alpha_j \geq 0$, $j \in \{1, \ldots, n_\mathcal{V}\}$, $\sum_{j=1}^{n_\mathcal{V}} \alpha_j = 1$. Therefore, finding the optimal $\mathcal{I}_F^{\text{app}}(\gamma)$ is equivalent to determining the optimal weighting vector γ.

Hence, we can rewrite Problem 1 as

$$\gamma^{\text{opt}} = \arg \max_{\gamma \in \mathbb{R}^{n_\mathcal{V}}} h_m(\mathcal{I}_F^{\text{app}}(\gamma)), \tag{11a}$$

$$\text{st.} \quad \mathcal{I}_F^{\text{app}}(\gamma) := \sum_{j=1}^{n_\mathcal{V}} \alpha_j \, \mathcal{I}_F^{(j)}, \tag{11b}$$

$$\sum_{j=1}^{n_\mathcal{V}} \alpha_j = 1, \tag{11c}$$

$$\alpha_j \geq 0, \text{ for all } j \in \{1, \ldots, n_\mathcal{V}\}, \tag{11d}$$

To solve the optimisation problem (11), we need to estimate the information matrix for each basis input. In the SMC literature, the observed information matrix is often estimated by the use of *Louis' identity* (Louis, 1982; Cappé et al., 2005). However, this approach does not guarantee that the information matrix estimate is positive semi-definite. In the authors' experience, this standard approach also leads to poor accuracy in the estimates.

Instead, we make use of the fact that the information matrix can be expressed as (3), i.e. the variance of the score function. Hence, a straight-forward method for estimating the information matrix is to use the Monte Carlo covariance estimator over some realisations of the system. If we denote each Monte Carlo estimate of

Algorithm 3 New input design method

Inputs: Algorithm 2, $\mathcal{C}$ (input values), n_m (memory) and T (no. input samples).
Output: $\gamma^\star$ (est. of the optimal weighting of the basis inputs).

```
1: Compute all the elementary cycles of $\mathcal{G}_{\mathcal{C}^{(n_m-1)}}$ by using, e.g., (Johnson, 1975, pp. 79–80), (Tarjan, 1972, pp. 157).
2: Compute all the prime cycles of $\mathcal{G}_{\mathcal{C}^{n_m}}$ from the elementary cycles of $\mathcal{G}_{\mathcal{C}^{(n_m-1)}}$ as explained above (c.f. (Zaman, 1983, Lemma 4)).
3: Generate the input signals $\{u_t^j\}_{t=0}^T$ from the prime cycles of $\mathcal{G}_{\mathcal{C}^{n_m}}$, for each $j \in \{1, \ldots, n_{\mathcal{V}}\}$.
4: Execute Algorithm 2.
```

the score function by $\widehat{\mathcal{S}}_m(\theta)$, the information matrix can be estimated using

$$\widehat{\mathcal{I}}_F = \frac{1}{M-1} \sum_{m=1}^{M} \widehat{\mathcal{S}}_m(\theta)\widehat{\mathcal{S}}_m^\top(\theta), \tag{12}$$

where M denotes the number of score estimates. Note, that this is an estimate of Fisher information matrix as the Monte Carlo estimator averages over the system realisations. The estimate is positive semi-definite by construction but inherits some bias from the FL-smoother, see Olsson et al. (2008) for more information. This problem can be handled by using more computationally costly particle smoother. Later, we present results indicating that this bias does not effect the resulting input signal to any large extent.

The information matrix estimate in (12) can be used to estimate $\mathcal{I}_F^{(j)}$ for each basis input. A simple solution is therefore to plug-in the estimates and solve the convex optimisation problem (11) using some standard solver. However, by doing this we neglect the stochastic nature of the estimates and disregard the uncertainty. In practice, this leads to bad estimates of γ.

Instead, we propose the use of a Monte Carlo method which iterates two different steps over K iterations. In step (a), we compute the information matrix estimates $\mathcal{I}_F^{(j)}$ for each input using (12). In step (b), we solve the optimisation problem in (11) using the estimates to obtain γ_k at iteration k. The estimate of the optimal weighting vector $\gamma^\star$ is found using the sample mean of $\gamma = \{\gamma_1, \ldots, \gamma_K\}$, which can be complemented with a CI. Such CI could be useful in determining which of the basis inputs that are significant and should be included in the optimal input sequence. The outline of the complete procedure is presented in Algorithm 2.

3.4 Summary of the method

The proposed method for designing of input signals in $\mathcal{C}^{n_m}$ is summarized in Algorithm 3. The algorithm computes $\gamma^\star$ which defines the optimal pmf $p_u^{\text{opt}}(u_{1:n_m})$ as a convex combination of the measures associated with the elements in $\mathcal{V}_{\mathcal{P}_\mathcal{C}}$, with $\gamma^\star$ as the weighting vector. Notice that $\mathcal{I}_F^{\text{app}}(\gamma)$ in (11b) is linear in the decision variables. Therefore, the optimization (11) is convex.

Input / $h_m(\widehat{\mathcal{I}}_F)$	$\log \det(\widehat{\mathcal{I}}_F)$	$\mathrm{tr}\left\{(\widehat{\mathcal{I}}_F)^{-1}\right\}$
Optimal (det)	20.67(0.01)	$1.51 \cdot 10^{-4}(5.18 \cdot 10^{-7})$
Optimal (tr)	20.82(0.01)	$1.32 \cdot 10^{-4}(4.45 \cdot 10^{-7})$
Binary	**20.91**(0.01)	$\mathbf{1.21 \cdot 10^{-4}}(4.51 \cdot 10^{-7})$
Uniform	19.38(0.01)	$5.32 \cdot 10^{-4}(2.12 \cdot 10^{-6})$

Table 1: *$h_m(\widehat{\mathcal{I}}_F)$, LGSS model.*

4 Numerical examples

The following examples present some applications of the proposed input design method.

4.1 Linear Gaussian state space model

Consider the linear Gaussian state space (LGSS) system,

$$\begin{aligned} x_{t+1} &= \theta_1 x_t + u_t + v_t, & v_t &\sim \mathcal{N}(0, \theta_2^2), \\ y_t &= x_t + e_t, & e_t &\sim \mathcal{N}(0, 0.1^2), \end{aligned}$$

where $\theta = \{\theta_1, \theta_2\}$ denotes the parameters with true values $\theta_0 = \{0.5, 0.1\}$. We design experiments to identify θ with $n_{\text{seq}} = 5 \cdot 10^3$ time steps, memory length $n_m = 2$, and an input assuming values $\mathcal{C} = \{-1,\, 0,\, 1\}$. The optimal experiments maximize $h_m(\mathcal{I}_F^{\text{app}}(\gamma)) = \det(\mathcal{I}_F^{\text{app}}(\gamma))$, and $h_m(\mathcal{I}_F^{\text{app}}(\gamma)) = -\mathrm{tr}\left\{(\mathcal{I}_F^{\text{app}}(\gamma))^{-1}\right\}$.

We generate $\{u_t^j\}_{t=0}^T$ for each $v_j \in \mathcal{V}_{\mathcal{P}_\mathcal{C}}$ ($T = 10^2$) to compute the approximation (12) for each $\mathcal{I}_F^{(j)}$. Finally, the optimal input $u_{1:n_{\text{seq}}}$ is computed by running a Markov chain with $p_u^{\text{opt}}(u_{1:n_m})$ as stationary pmf, where we discard the first $2 \cdot 10^6$ samples and keep the last $n_{\text{seq}} = 5 \cdot 10^3$ ones. In addition, we consider $K = 100$, $M = 5 \cdot 10^3$ and $N = 10^3$. As a benchmark, we generate n_{seq} input samples from uniformly distributed white noise with support $[-1,\, 1]$, and the same amount of samples from binary white noise with values $\{-1,\, 1\}$. These input samples are employed to compute an approximation of $\mathcal{I}_F$ via (12).

Table 1 condenses the results obtained for each input sequence, where *Optimal (det)* and *Optimal (tr)* represent the results for the input sequences obtained from optimizing $\det(\mathcal{I}_F^{\text{app}}(\gamma))$, and $-\mathrm{tr}\left\{(\mathcal{I}_F^{\text{app}}(\gamma))^{-1}\right\}$, respectively. The 95% confidence intervals are given as ± the value in the parentheses. From the data we conclude that, for this particular example, the binary white noise seems to be the best input sequence. Indeed, the proposed input design method tries to mimic the binary white noise excitation, which is clear from the numbers in Table 1.

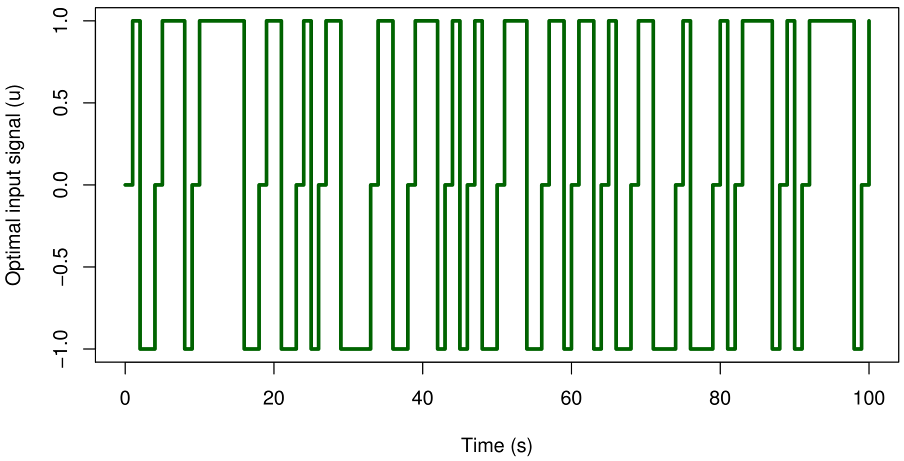

Figure 3: *Input realization, Nonlinear growth model.*

Input / $h_m(\widehat{\mathcal{I}}_F)$	$\log \det(\widehat{\mathcal{I}}_F)$
Optimal	**25.34**(0.01)
Binary	24.75(0.01)
Uniform	24.38(0.01)

Table 2: *$h_m(\widehat{\mathcal{I}}_F)$, Nonlinear growth model.*

4.2 Nonlinear growth model

In this example we consider the system in (Gopaluni et al., 2011, Section 6), given by

$$\begin{aligned} x_{t+1} &= \theta_1 x_t + \frac{x_t}{\theta_2 + x_t^2} + u_t + v_t, && v_t \sim \mathcal{N}(0, 0.1^2), \\ y_t &= \frac{1}{2} x_t + \frac{2}{5} x_t^2 + e_t, && e_t \sim \mathcal{N}(0, 0.1^2), \end{aligned}$$

where $\theta = \{\theta_1, \theta_2\}$ denotes the parameters with true values $\theta_0 = \{0.7, 0.6\}$. We design an experiment with the same settings as in the LGSS model, maximizing $h_m(\mathcal{I}_F^{\text{app}}(\gamma)) = \det(\mathcal{I}_F^{\text{app}}(\gamma))$. A typical input realization obtained from the proposed input design method is presented in Figure 3.

Table 2 presents the results obtained for each input sequence, where *Optimal* represents the result for the input sequence obtained from optimizing $\det(\mathcal{I}_F^{\text{app}}(\gamma))$. The 95% confidence intervals are given as ± the value in the parentheses. From these data we conclude that the extended input design method outperforms the experiment results obtained for binary and uniformly distributed samples. Therefore, our new input design method can be successfully employed to design experiments for this nonlinear system.

5 Conclusion

We have presented a new input design method for state space models, which extends existing input design approaches for nonlinear systems. The extension considers a more general model structure, and a new class for the input sequences. The method maximizes a scalar cost function of the information matrix, by optimizing the stationary pmf from which the input sequence is sampled. The elements in the feasible set of the stationary pmf are computed as a convex combination of its extreme points.

Under the assumption of a finite set of possible values for the input, we use graph theoretical tools to compute the information matrix as a convex combination of the information matrices associated with each extreme point. The information matrix for each extreme point is approximated using particle methods, where the information matrix is computed as the covariance of the score function. The numerical examples show that the extended input design method can be successfully used to design experiments for general nonlinear systems.

In a future work we will combine the proposed input design technique with parameter estimation methods, which will allow to simultaneously estimate the parameters and the optimal input for a nonlinear SSM. We will also consider alternative methods based on Gaussian process models for information matrix estimation. This could improve the accuracy and the efficiency in the information matrix estimation method outlined in this paper.

Finally, as with most optimal input design methods, the one proposed in this contribution relies on knowledge of the true system. This difficulty can be overcome by implementing a robust experiment design scheme on top of it (Rojas et al., 2007) or via an adaptive procedure, where the input signal is re-designed as more information is being collected from the system (Rojas et al., 2011). This approach will be also addressed in a future work.

Acknowledgements

The authors thank to Dr. Fredrik Lindsten for his comments to improve this article.

Bibliography

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

C. Brighenti, B. Wahlberg, and C. R. Rojas. Input design using Markov chains for system identification. In *Proceedings of the joint 48th Conference on Decision and Control and 28th Chinese Conference*, pages 1557–1562, Shangai, P. R. China, dec 2009.

O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models*. Springer, 2005.

A. De Cock, M. Gevers, and J. Schoukens. A preliminary study on optimal input design for nonlinear systems. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, Florence, Italy, December 2013.

D. R. Cox. *Planning of experiments*. New York: Wiley, 1958.

J. Dahlin, F. Lindsten, and T. B. Schön. Second-order particle MCMC for Bayesian parameter inference. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication).

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

V. V. Fedorov. *Theory of optimal experiments*. Academic Press, 1972.

R. A. Fisher. Theory of statistical estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 22(05):700–725, 1925.

G. C. Goodwin and R. L. Payne. *Dynamic System Identification: Experiment Design and Data Analysis*. Academic Press, New York, 1977.

R. B. Gopaluni, T. B. Schön, and A. G. Wills. Input design for nonlinear stochastic dynamical systems - a particle filter approach. In *Proceedings of the 18th IFAC World Congress*, Milano, Italy, August 2011.

R. Hildebrand and M. Gevers. Identification for control: Optimal input design with respect to a worst-case ν-gap cost function. *SIAM Journal of Control Optimization*, 41(5):1586–1608, 2003.

H. Hjalmarsson and J. Mårtensson. Optimal input design for identification of non-linear systems: Learning from the linear case. In *Proceedings of the American Control Conference (ACC)*, pages 1572–1576, New York, United States, July 2007.

H. Jansson and H. Hjalmarsson. Input design via LMIs admitting frequency-wise model specifications in confidence regions. *IEEE Transactions on Automatic Control*, 50(10):1534–1549, oct 2005.

D. B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, mar 1975.

G. Kitagawa and S. Sato. Monte Carlo smoothing and self-organising state-space model. In A. Doucet, N. de Fretias, and N. Gordon, editors, *Sequential Monte Carlo methods in practice*, pages 177–195. Springer, 2001.

C. Larsson, H. Hjalmarsson, and C. R. Rojas. On optimal input design for nonlinear FIR-type systems. In *Proceedings of the 49th IEEE Conference on Decision and Control (CDC)*, pages 7220–7225, Atlanta, USA, December 2010.

K. Lindqvist and H. Hjalmarsson. Optimal input design using linear matrix inequalities. In *Proceedings of the IFAC Symposium on System Identification (SYSID)*, Santa Barbara, California, USA, July 2000.

L. Ljung. *System identification: theory for the user.* Prentice Hall, 1999.

T. A. Louis. Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 44(02):226–233, 1982.

J. Olsson, O. Cappé, R. Douc, and E. Moulines. Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state space models. *Bernoulli*, 14(1):155–179, 2008.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

C. R. Rojas, J. S. Welsh, G. C. Goodwin, and A. Feuer. Robust optimal experiment design for system identification. *Automatica*, 43(6):993–1008, June 2007.

C. R. Rojas, H. Hjalmarsson, L. Gerencsér, and J. Mårtensson. An adaptive method for consistent estimation of real-valued non-minimum phase zeros in stable LTI systems. *Automatica*, 47(7):1388–1398, 2011.

H. Suzuki and T. Sugie. On input design for system identification in time domain. In *Proceedings of the European Control Conference*, Kos, Greece, July 2007.

R. Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146–160, June 1972.

P. E. Valenzuela, C. R. Rojas, and H. Hjalmarsson. Optimal input design for dynamic systems: a graph theory approach. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, Florence, Italy, dec 2013.

P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. A graph/particle-based method for experiment design in nonlinear systems. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, August 2014. (accepted for publication).

T. L. Vincent, C. Novara, K. Hsu, and K. Poola. Input design for structured nonlinear system identification. In *Proceedings of the 15th IFAC Symposium on System Identification (IFAC)*, pages 174–179, Saint-Malo, France, July 2009.

P. Whittle. Some general points in the theory of optimal experiment design. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 1:123–130, 1973.

A. Zaman. Stationarity on finite strings and shift register sequences. *The Annals of Probability*, 11(3):678–684, August 1983.

Paper E

Hierarchical Bayesian approaches for robust inference in ARX models

Authors: J. Dahlin, F. Lindsten, T. B. Schön and A. Wills

Supported by the project *Probabilistic modelling of dynamical systems* (Contract number: 621-2013-5524) funded by the Swedish Research Council.

Edited version of the paper:

J. Dahlin, F. Lindsten, T. B. Schön, and A. Wills. Hierarchical Bayesian ARX models for robust inference. In *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*, Brussels, Belgium, July 2012b.

Preliminary version:

Technical Report LiTH-ISY-R-3041, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Hierarchical Bayesian approaches for robust inference in ARX models

J. Dahlin[⋆], F. Lindsten[†], T. B. Schön[‡] and A. Wills[♣]

[⋆]Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden.
johan.dahlin@isy.liu.se

[†]Dept. of Engineering,
University of Cambridge,
CB2 1PZ Cambridge, United Kingdom.
fredrik.lindsten@eng.cam.ac.uk

[‡]Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
thomas.schon@it.uu.se

[♣]School of EECS,
University of Newcastle,
Callaghan, NSE, Australia.
adrian.wills@newcastle.edu.au

Abstract

Gaussian innovations are the typical choice in most ARX models but using other distributions such as the Student's t could be useful. We demonstrate that this choice of distribution for the innovations provides an increased robustness to data anomalies, such as outliers and missing observations. We consider these models in a Bayesian setting and perform inference using numerical procedures based on Markov Chain Monte Carlo methods. These models include automatic order determination by two alternative methods, based on a parametric model order and a sparseness prior, respectively. The methods and the advantage of our choice of innovations are illustrated in three numerical studies using both simulated data and real EEG data.

1 Introduction

An autoregressive exogenous (ARX) model of orders $n = \{n_a, n_b\}$, is given by

$$y_t + \sum_{i=1}^{n_a} a_i^n y_{t-i} = \sum_{i=1}^{n_b} b_i^n u_{t-i} + e_t, \tag{1}$$

where a_i^n and b_i^n are model coefficients, u_t is a known input signal and e_t is white excitation noise, often assumed to be Gaussian and independent of the input signal. Then, for known model orders n, the maximum likelihood estimate of the unknown ARX coefficients $\theta^n = \{a_1^n, \ldots, a_{n_a}^n, b_1^n, \ldots, b_{n_b}^n\}$ is given by least squares (LS). In practice, we are often faced with the following problems:

1. The appropriate model order is unknown or no "best" model order may exist.

2. The observed data is non-Gaussian in nature, e.g. due to outliers.

In this work, we propose two hierarchical Bayesian ARX models and algorithms to make inference in these models, thereby addressing both of the practical issues mentioned above. The proposed models differs from (1) in two aspects: (i) the excitation noise is modelled as Student's t distributed, and (ii) a built-in form of automatic order selection is used.

The t distribution is more heavy-tailed than the Gaussian distribution, which means that the proposed ARX model can capture "jumps" in the internal state of the system (as an effect of occasional large innovations). Furthermore, we believe that this will result in an inference method that is more robust to model errors and outliers in the observations, a property which we illustrate in this work.

We propose two alternative methods to automatically determine the system order n. Firstly, we let the model order n be a parameter of the Bayesian ARX model. The model order is inferred alongside the other unknown parameters, resulting in a posterior probability distribution over model orders. In the second model, we instead use a sparseness prior over the ARX coefficients, known as automatic relevance determination (ARD) (MacKey, 1994; Neal, 1996).

Based on the models introduced above, the resulting identification problem amounts to finding the posterior distribution of the model parameters θ^n and the order n. This is done using Markov Chain Monte Carlo Methods (see e.g. Robert and Casella (2004)), where we are constructing a Markov Chain with the posterior distribution as its stationary distribution. We can thus compute estimates under the posterior parameter distribution by sampling from the constructed Markov Chain.

For the first model, this is a challenging task as the model order is explicitly included in the parameter vector. This is due to the fact that we are now dealing with a parameter space of varying dimension, which thereby require the Markov Chain to do the same. This will be solved using the reversible jump MCMC (RJ-MCMC) algorithm introduced by Green (1995). The inference problem resulting from the use of an ARD prior is in the other hand solvable using standard MCMC algorithms.

The use of RJ-MCMC to estimate the model order and the parameters of an AR model driven by Gaussian noise, is fairly well studied, see e.g. (Troughton and Godsill, 1998; Godsill, 2001; Brooks et al., 2003). The present work differs from these contributions, mainly in the use of Student's t distributed innovations. Similar models are also considered by Christmas and Everson (2011), who derive a variational Bayes algorithm for the inference problem. This approach is not based on Monte Carlo sampling, but instead makes use of certain deterministic approximations to overcome the intractable integrals that appear in the expression for the posterior distribution.

2 Hierarchical Bayesian ARX Models

In this section, we present the two proposed hierarchical Bayesian ARX models both using Student's t distributed excitation noise, as described in Section 2.1. The models differ in how the model orders are incorporated. The two alternatives are presented in Sections 2.2 and 2.3, respectively.

2.1 Student's t distributed innovations

We model the excitation noise as Student's t-distributed, with scale λ and ν degrees of freedom (DOF)

$$e_t \sim \mathcal{S}t(0, \lambda, \nu). \tag{2}$$

This can equivalently be seen as a latent variable model in which e_t is modelled as zero-mean Gaussian with unknown variance $(\lambda z_t)^{-1}$ and z_t is a gamma distributed latent variable. Hence, an equivalent model to (2) is given by

$$z_t \sim \mathcal{G}(\nu/2, \nu/2), \tag{3a}$$

$$e_t \sim \mathcal{N}(0, (\lambda z_t)^{-1}), \tag{3b}$$

where $\mathcal{G}(\alpha, \beta)$ is the gamma distribution with shape α and inverse scale β and $\mathcal{N}(\mu, \sigma^2)$ is the Gaussian distribution with mean μ and variance σ^2.

Note that λ and ν are unknowns, we wish to infer these in the proposed Bayesian models. As we do not know much about these parameters, vague (non-informative) gamma priors are used as in Christmas and Everson (2011)

$$p(\lambda) = \mathcal{G}(\lambda; \alpha_\lambda, \beta_\lambda), \tag{4a}$$

$$p(\nu) = \mathcal{G}(\nu; \alpha_\nu, \beta_\nu), \tag{4b}$$

where α and β denote hyperparameters that we define below. Note that these are standard choices resulting from the property of *conjugate priors*. This type of priors used in combination with a suitable likelihood gives an analytical expression for the posterior, see e.g. Bishop (2006) for other examples of conjugate priors.

2.2 Parametric model order

The first automatic order determination alternative is to infer the order n along with the model parameters. Assume that there exists some maximum order such that $n_a, n_b \leq n_{\max}$, resulting in $n_{\max}^2$ different model hypotheses

$$\mathcal{M}_n : \quad y_t = (\varphi_t^n)^\top \theta^n + e_t, \tag{5}$$

for $n = \{1,1\}, \{1,2\}, \ldots, \{n_{\max}, n_{\max}\}$, where

$$\varphi_t^n = \{-y_{t-1}, \ldots, -y_{t-n_a}, u_{t-1}, \ldots, u_{t-n_b}\}^\top, \tag{6}$$

denotes the known inputs and outputs, θ^n the model coefficients, and e_t the excitation noise that is assumed to be independent of the input signal. We use a uniform prior distribution over these model hypotheses with order n as

$$p(n) = \begin{cases} 1/n_{\max}^2 & \text{if } n_a, n_b \in \{1, \ldots, n_{\max}\}, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Furthermore, we model the coefficients θ^n as random vectors, with prior distributions

$$p(\theta^n \mid n, \delta) = \mathcal{N}(\theta^n; 0, \delta^{-1} I_{n_a+n_b}), \tag{8}$$

with the same variance δ^{-1} for all orders n and where $\mathbf{I}_n$ denotes the $n \times n$ identity matrix. Finally, we place the standard conjugate gamma prior on δ as

$$p(\delta) = \mathcal{G}(\delta; \alpha_\delta, \beta_\delta). \tag{9}$$

All put together, the collection of unknowns of the model is given by

$$\eta = \{\theta^n, n, \delta, z_{1:T}, \lambda, \nu\}. \tag{10}$$

The latent variables $z_{1:T}$, as well as the coefficients' variance δ^{-1}, can be seen as nuisance parameters which are not really of interest, but they will simplify the inference.

2.3 Automatic relevance determination

An alternative approach for order determination is to use ARD. Consider a high-order ARX model with fixed orders $n = \{n_{\max}, n_{\max}\}$. Hence, we overparameterise the model and the ARX coefficients θ will be a vector of fixed dimension $m = 2n_{\max}$. To avoid overfitting, we place a sparseness prior, known as ARD, on the ARX coefficients

$$p(\theta_i \mid \delta_i) = \mathcal{N}(\theta_i; 0, \delta_i^{-1}), \tag{11}$$

with the conjugate distribution on the variance

$$p(\delta_i) = \mathcal{G}(\delta_i; \alpha_\delta, \beta_\delta), \tag{12}$$

for $i = 1, \ldots, m$. The difference between the ARD prior and (8) is that in (11), each coefficient is governed by a different variance, which is IID according to (12). If there is not enough evidence in the data that the ith parameter should be non-zero, this prior will favor a large value for δ_i which means that the ith parameter in effect will be "switched off". Hence, the ARD prior will encourage a sparse solution; see e.g. MacKey (1994); Neal (1996) for further discussion. When using the ARD prior, the collection of unknowns of the model is given by

$$\eta = \{\theta, \delta_{1:m}, z_{1:T}, \lambda, \nu\}, \tag{13}$$

where θ is the parameter vector of the overparameterised model of order $n_{\max}$.

3 Markov chain Monte Carlo

Assume that we have observed a sequence of input/output pairs $D_T = \{u_{1:T}, y_{1:T}\}$. We then seek the posterior distribution of the model parameters, $p(\eta \mid D_T)$, which is not available in closed form. An MCMC sampler is therefore used to approximately sample from the posterior distribution.

The most fundamental MCMC sampler is known as the Metropolis-Hastings (MH) algorithm. In this method, we propose a new value for the state of the Markov chain from some arbitrary chosen proposal kernel. The proposed value is then accepted with a certain probability, otherwise the previous state of the chain is kept.

A special case of the MH algorithm is the Gibbs sampler. In this method, we loop over the different variables of our model, sampling each variable conditioned on the remaining ones. By using these conditional posterior distributions as proposals, the MH acceptance probability will be exactly one. Hence, the Gibbs sampler will always accept its proposed values. As pointed out by Tierney (1994), it is possible to mix different types of proposals. This will be done in the sampling strategies employed in this work, where we use Gibbs moves for some variables and random walk MH moves for other variables.

A generalisation of the MH sampler is the reversible jump MCMC (RJ-MCMC) sampler (Green, 1995), which allows for moves between parameter spaces of different dimensionality. This approach will be used in this work, for the model presented in Section 2.2. The reason is that when the model order n is seen as a parameter, the dimension of the vector θ^n will change between iterations. An RJ-MCMC sampler can be seen as employing standard MH moves, but all variables that are affected by the changed dimensionality must either be accepted or rejected as a group. That is, in our case, we propose new values for $\{n, \theta^n\}$ as a pair, and either accept or reject both of them (see step (I-1a) below).

For the ARX model with parametric model order, we employ an RJ-MCMC sampler using the following sweep[1],

[1] The reason for why we condition on some variables from time $s+1$ to T, instead of from time 1 to T, is to deal with the unknown initial state of the system. This will be explained in more detail in Section 4.2.

(I-1) Order and ARX coefficients:

(a) Draw $\{\theta^{n^\star}, n^\star\} \mid z_{s+1:T}, \lambda, \delta, D_T$.

(b) Draw $\delta^\star \mid \theta^{n^\star}, n^\star$.

(I-2) Innovation parameters:

(a) Draw $z^\star_{s+1:T} \mid \theta^{n^\star}, n^\star, \lambda, \nu, D_T$.

(b) Draw $\lambda^\star \mid \theta^{n^\star}, n^\star, z^\star_{s+1:T}, D_T$.

(c) Draw $\nu^\star \mid z^\star_{s+1:T}$.

If we instead consider the ARX model with an ARD prior we use the following sweep, denoted ARD-MCMC,

(II-1) ARX coefficients:

(a) Draw $\theta^\star \mid z_{s+1:T}, \lambda, \delta_{1:m}, D_T$.

(b) Draw $\delta^\star_{1:m} \mid \theta^\star$.

(II-2) Innovation parameters:

(a) Draw $z^\star_{s+1:T} \mid \theta^\star, \lambda, \nu, D_T$.

(b) Draw $\lambda^\star \mid \theta^\star, z^\star_{s+1:T}, D_T$.

(c) Draw $\nu^\star \mid z^\star_{s+1:T}$.

The difference between the two methods lies in steps (I-1) and (II-1), where the parameters related to the ARX coefficients are sampled. In steps (I-2) and (II-2), we sample the parameters of the excitation noise distribution, and these steps are essentially the same for both samplers.

4 Posteriors and proposal distributions

In this section, we present the posterior and proposal distributions for the model order and other parameters used by the proposed MCMC methods.

4.1 Model order

Sampling the model order and the ARX coefficients in step (I-1a) is done via a reversible jump MH step. We start by proposing a new model order n', according to some chosen proposal kernel $q(n' \mid n)$. In this work, we follow the suggestion by Troughton and Godsill (1998) and use a constrained random walk with discretised Laplace increments with scale parameter ℓ, i.e.

$$q(n'_a \mid n) \propto \exp(-\ell|n'_a - n_a|), \qquad \text{if } 1 \leq n'_a \leq n_{\max}, \tag{14}$$

and analogously for n_b. This proposal will favour small changes in the model order, but allows for occasional large jumps.

Once we have sampled the proposed model order n', we generate a set of ARX coefficients from the posterior distribution

$$\theta^{n'} \sim p(\theta^{n'} \mid n', z_{s+1:T}, \lambda, \delta, D_T) = \mathcal{N}(\theta^{n'}; \mu_{\theta^{n'}}, \Sigma_{\theta^{n'}}). \tag{15}$$

The expressions for the mean and the covariance of this Gaussian distribution are provided in the subsequent section. Now, since the proposed coefficients $\theta^{n'}$ are directly connected to the model order n', we apply an MH accept/reject decision to the pair $\{\theta^{n'}, n'\}$. The MH acceptance probability is given by

$$\begin{aligned}\rho_{nn'} &\triangleq 1 \wedge \frac{p(n', \theta^{n'} \mid z_{s+1:T}, \lambda, \delta, D_T)}{p(n, \theta^{n} \mid z_{s+1:T}, \lambda, \delta, D_T)} \frac{q(n, \theta^{n} \mid n', \theta^{n'})}{q(n', \theta^{n'} \mid n, \theta^{n})} \\ &= 1 \wedge \frac{p(n' \mid z_{s+1:T}, \lambda, \delta, D_T)}{p(n \mid z_{s+1:T}, \lambda, \delta, D_T)} \frac{q(n \mid n')}{q(n' \mid n)},\end{aligned} \tag{16}$$

where $a \wedge b \triangleq \min(a, b)$. Since

$$p(n \mid z_{s+1:T}, \lambda, \delta, D_T) \propto p(y_{1:T} \mid n, z_{s+1:T}, \lambda, \delta, u_{1:T}) p(n), \tag{17}$$

where the prior over model orders is flat according to (7), the acceptance probability can be simplified to (Troughton and Godsill, 1998)

$$\rho_{nn'} = 1 \wedge \frac{\delta^{\frac{n'}{2}} |\Sigma_{\theta^{n'}}|^{\frac{1}{2}} \exp\left(\frac{1}{2}\mu_{\theta^{n'}}^{\top} \Sigma_{\theta^{n'}}^{-1} \mu_{\theta^{n'}}\right)}{\delta^{\frac{n}{2}} |\Sigma_{\theta^{n}}|^{\frac{1}{2}} \exp\left(\frac{1}{2}\mu_{\theta^{n}}^{\top} \Sigma_{\theta^{n}}^{-1} \mu_{\theta^{n}}\right)} \frac{q(n \mid n')}{q(n' \mid n)}.$$

Note by (21) that the acceptance probability does not depend on the actual value of $\theta^{n'}$. Hence, we do not have to carry out the sampling according to (15) unless the proposed sample is accepted.

4.2 ARX coefficients

The ARX coefficients are sampled in step (I-1a) and step (II-1a) of the two proposed MCMC samplers, respectively. In both cases, we sample from the posterior distribution over the parameters; see (15). In this section, we adopt the notation used in the RJ-MCMC sampler, but the sampling is completely analogous for the ARD-MCMC sampler. A "stacked" version of the linear regression model (5) is

$$y_{s+1:T} = \Phi^{n} \theta^{n} + e_{s+1:T}, \tag{18}$$

where the regression matrix Φ^n is given by

$$\Phi^{n} = \begin{pmatrix} -y_s & \cdots & -y_{s-n_a} & u_s & \cdots & u_{s-n_b+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -y_{T-1} & \cdots & -y_{T-n_a} & u_{T-1} & \cdots & u_{T-n_b} \end{pmatrix}. \tag{19}$$

Here, we have take into account that the initial state of the system is not known, and only use observations from time $s + 1$ to T in the vector of observations on the left hand side of (18). For the RJ-MCMC sampler $s = \max(n_a, n'_a)$ and for the ARD-MCMC sampler $s = n_{\max}$.

Let Δ^{-1} be the covariance matrix for the parameter prior, either according to (8) or according to (11), i.e.

$$\Delta^{-1} = \begin{cases} \delta I_{n_a+n_b} & \text{for RJ-MCMC,} \\ \mathsf{diag}(\delta_1, \ldots, \delta_m) & \text{for ARD-MCMC.} \end{cases} \tag{20}$$

Since we condition on the latent variables $z_{s+1:T}$ (and the varince parameter λ^{-1}), the noise term in (18) can be viewed as Gaussian according to (3b). It follows that the posterior parameter distribution is Gaussian, as already stated in (15), with mean and covariance given by

$$\mu_{\theta^n} = \Sigma_{\theta^n}(\Phi^n)^\top(\lambda z_{s+1:T} \circ y_{s+1:T}), \tag{21a}$$

$$\Sigma_{\theta^n} = \Big((\Phi^n)^\top \mathsf{diag}(\lambda z_{s+1}, \ldots, \lambda z_T)\Phi^n + \Delta\Big)^{-1}, \tag{21b}$$

respectively. Here, $\circ$ denotes elementwise multiplication.

4.3 ARX coefficients variance

We now derive the posterior distributions for the ARX coefficients variance(s), sampled in steps (I-1b) and (II-1b) for the two models, respectively.

Consider first the model described with parametric model order. The ARX coefficients variance δ^{-1} is *a priori* gamma distributed according to (9). The likelihood is given by (8) and an analytical expression for the posterior distribution is easily found as the gamma distributed is a conjugate prior. Thereby motivating the standard choice of a gamma distributed prior for the inverse variance in a Gaussian distribution. It follows from standard results (see e.g. Bishop (2006, p. 100)) that

$$p(\delta \mid \theta^n, n) = \mathcal{G}(\delta; \alpha_\delta^{\text{post}}, \beta_\delta^{\text{post}}), \tag{22}$$

with hyperparameters

$$\alpha_\delta^{\text{post}} = \alpha_\delta + \frac{n_a + n_b}{2}, \quad \text{and} \quad \beta_\delta^{\text{post}} = \beta_\delta + \frac{1}{2}(\theta^n)^\top\theta^n. \tag{23}$$

Similarly, for the ARD model, we get from the prior (12) and the likelihood (11), that the posterior distributions for the ARX coefficients variances are given by

$$p(\delta_i \mid \theta_i) = \mathcal{G}(\delta_i; \alpha_{\delta_i}^{\text{post}}, \beta_{\delta_i}^{\text{post}}), \tag{24}$$

with hyperparameters

$$\alpha_{\delta_i}^{\text{post}} = \alpha_\delta + \frac{1}{2}, \quad \text{and} \quad \beta_{\delta_i}^{\text{post}} = \beta_\delta + \frac{1}{2}\theta_i^2, \tag{25}$$

for $i = 1, \ldots, m$.

4.4 Latent variance variables

Let us now turn to the parameters defining the excitation noise distribution. We start with the latent variance variables $z_{s+1:T}$. These variables are sampled analogously in steps (I-2a) and (II-2a). The latent variables are *a priori* gamma distributed according to (3a) and since they are IID, we focus on one of them, say z_t. Note that we here once again have chosen a prior distribution conjugate to the likelihood.

The likelihood model for z_t is given by (5), where the model order now is fixed since we condition on n (in the ARD model, the order is always fixed)

$$p(y_t \mid z_t, \theta^n, n, \lambda, \nu, \varphi_t^n) = \mathcal{N}(y_t, (\varphi_t^n)^\top \theta^n, (\lambda z_t)^{-1}). \tag{26}$$

It follows that the posterior is given by

$$p(z_t \mid \theta^n, n, \lambda, \nu, D_T) = \mathcal{G}(z_t; \alpha_z^{\text{post}}, \beta_{z_t}^{\text{post}}), \tag{27}$$

with the hyperparameters

$$\alpha_z^{\text{post}} = \frac{1}{\nu} + \frac{1}{2}, \quad \text{and} \quad \beta_{z_t}^{\text{post}} = \frac{\nu}{2} + \frac{\lambda}{2}\epsilon_t^2. \tag{28}$$

Here, the prediction error ϵ_t is given by

$$\epsilon_t = y_t - (\varphi_t^n)^\top \theta^n. \tag{29}$$

We can thus generate $z_{s+1:T}^\star$ by sampling independently from (27) for $t = s + 1, \dots, T$.

4.5 Innovation scale parameter

The innovation scale parameter λ is sampled in steps (I-2b) and (II-2b). This variable follows a model that is very similar to z_t. The difference is that, whereas the individual z_t variables are IID and only enter the likelihood model (5) for a single t each, we have the same λ for all time instances. The posterior distribution of λ is thus given by

$$p(\lambda \mid \theta^n, n, z_{s+1:T}, D_T) = \mathcal{G}(\lambda; \alpha_\lambda^{\text{post}}, \beta_\lambda^{\text{post}}), \tag{30}$$

with

$$\alpha_\lambda^{\text{post}} = \alpha_\lambda + \frac{T-s}{2}, \quad \text{and} \quad \beta_\lambda^{\text{post}} = \beta_\lambda + \frac{1}{2}\epsilon_{s+1:T}^\top(z_{s+1:T} \circ \epsilon_{s+1:T}), \tag{31a}$$

where the prediction errors $\epsilon_{s+1:T}$ are given by (29).

4.6 Innovation DOF

The DOF ν, sampled in steps (I-2c) and (II-2c), is *a priori* gamma distributed according to (4b). The likelihood for this variable is given by (3a). It follows that the posterior of ν is given by

$$p(\nu \mid z_{s+1:T}) \propto p(z_{s+1:T} \mid \nu)p(\nu) = \prod_{t=s+1}^{T} \mathcal{G}(z_t; \nu/2, \nu/2)\mathcal{G}(\nu; \alpha_\nu, \beta_\nu). \tag{32}$$

Unfortunately, this does not correspond to any standard distribution. To circumvent this, we apply an MH accept/reject step to sample the DOF. Hence, we propose a value according to some proposal kernel $\nu' \sim q(\nu' \mid \nu)$. Here, the proposal is taken as a Gaussian random walk, constrained to the positive real line.

The proposed sample is accepted with probability

$$\rho_{\nu\nu'} = 1 \wedge \frac{p(\nu' \mid z_{s+1:T})}{p(\nu \mid z_{s+1:T})} \frac{q(\nu \mid \nu')}{q(\nu' \mid \nu)}, \tag{33}$$

which can be computed using (32).

5 Numerical illustrations

We now give some numerical results to illustrate the performance of the proposed methods. First, we compare the average performance of the MCMC samplers with least squares (LS) in Section 5.1. These experiments are included mostly to build some confidence in the proposed method. We then illustrate how the proposed methods are affected by outliers and missing data in Section 5.2. As a final example, in Section 5.3 we illustrate the performance of the RJ-MCMC on real EEG data.

5.1 Average model performance

We evaluate the proposed methods by analysing the average identification performance for $25,000$ randomly generated ARX systems. These systems are generated by sampling a uniform number of poles and zeros (so that the resulting system is strictly proper) up to some maximum order, here taken as 30. The poles and zeros are generated uniformly over a disc with radius 0.95.

For each system, we generate $T = 450$ observations[2]. The input signal u_t is generated as Gaussian white noise with standard deviation 0.1. The innovations are simulated from a Student's t distribution, $e_t \sim \mathcal{S}t(0, 1, 2)$. The hyperparameters of the model are chosen as $\alpha_\lambda = \beta_\lambda = \alpha_\nu = \beta_\nu = \alpha_\delta = \beta_\delta = 0.1$.

The data is split into three parts with 150 observations each. The first two parts are used for model estimation, and the last part is used for testing the model. For the LS method, we employ cross validation by first estimating models for all possible combinations of model orders n_a and n_b, such that both are less than or equal to $n_{\max} = 30$, on the first batch of data. We then pick the model corresponding to the best model fit (Ljung, 1999, p. 500). The full estimation data set (300 observations) is then used to re-estimate the model parameters. For the MCMC methods, we use all the estimation data at once, since these methods comprise automatic order determination and no explicit order selection is made.

The average model fit for the test data, for the 25,000 ARX systems is given in Table 1. We note a slight statistically significant improvement by using the RJ-MCMC method in comparison with the standard LS technique. Also, the RJ-MCMC appear to perform better than the simpler ARD-MCMC method (for this model class). Therefore, we will focus primarily on the former method in the remainder of the numerical illustrations.

[2] When simulating the systems, we run the simulations for 900 time steps, out of which the first 450 observations are discarded, to remove the effect of transients.

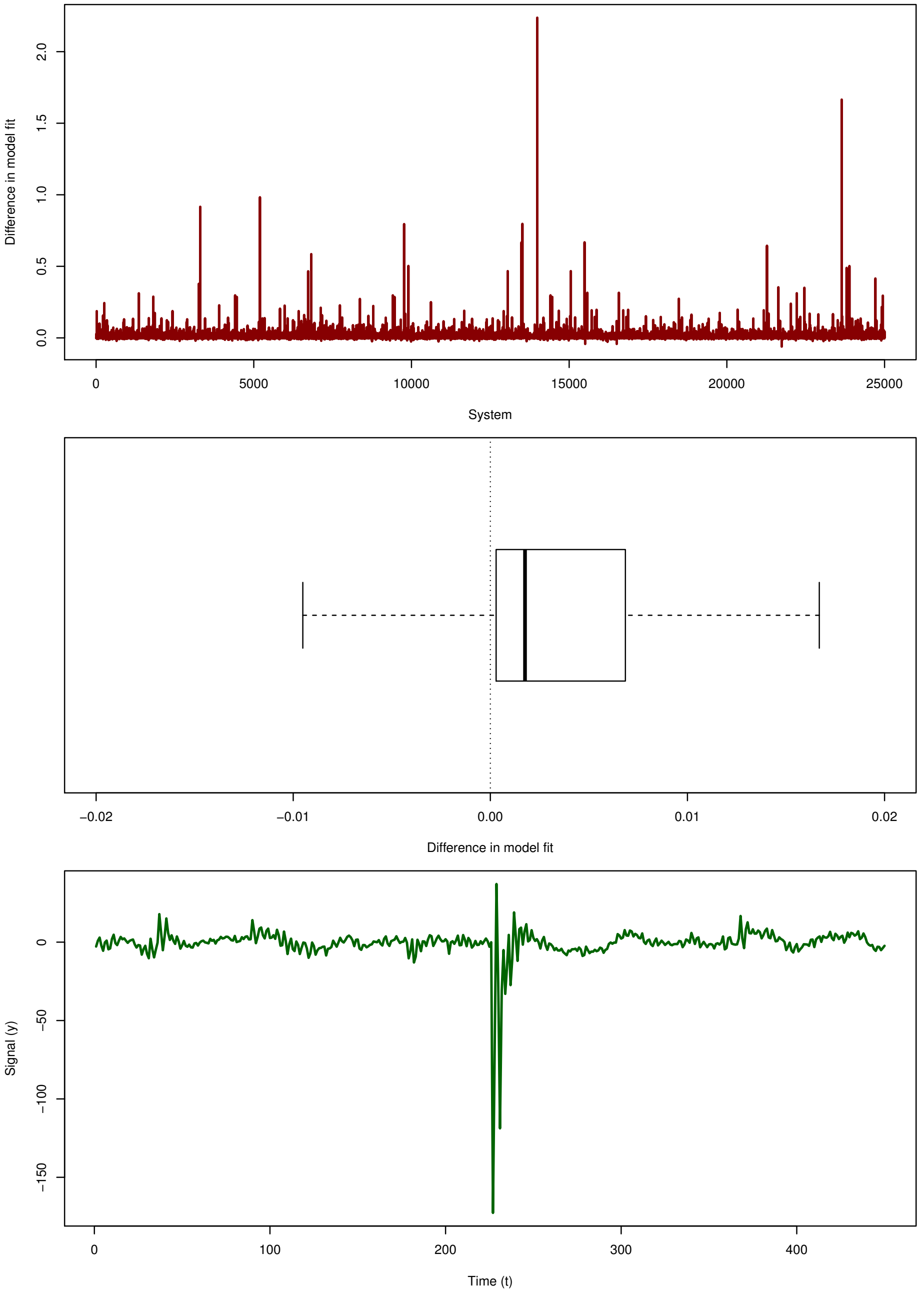

Figure 1: *Upper: The difference in model fit between the RJ-MCMC and LS methods. Middle: A boxplot of the difference in model fit with the outliers removed. Lower: One particular randomly generated ARX model with a large innovation outlier that affects the system output.*

Method	Mean	CI
LS	77.51	[77.21 77.81]
RJ-MCMC	78.24	[77.95 78.83]
ARD-MCMC	77.73	[77.47 78.06]

Table 1: *The average and 95% confidence intervals (CI) for the model fit (in percent) from experiments with* $25,000$ *random ARX models.*

In the upper part of Figure 1, the differences in model fit between RJ-MCMC and LS for all 25,000 systems are shown. We note that there are no cases with large negative values, indicating that the RJ-MCMC method performs at least as good as, or better than, LS for the vast majority of these systems. We also note that there are a few cases in which LS is much worse that RJ-MCMC. Hence, the average model fit for LS is deteriorated by the fact that the method fails completely from "time to time". This is not the case for the proposed RJ-MCMC sampler (nor for the ARD-MCMC sampler), which suggests that the proposed method is more robust to variations in the data.

It is interesting to review a typical case with a large difference in model fit between the two methods. Data from such a case is shown in the lower part of Figure 1. Here, we see a large jump in the system state. The ARX model with Student's t distributed innovations can, due to the heavy tails of the noise distribution, accommodate for the large output values better than the model with Gaussian noise. The model fit for this system was 46.15% for the RJ-MCMC method and 14.98% for the LS methods.

It is important to note that the use of the LS method is due to its simplicity. For the problem under study the LS method is the maximum likelihood (ML) solution to an ARX model with Gaussian noise and a given model order. The ML problem can of course also be posed for the case where t distributed noise is assumed. Another alternative would be to make use of a prediction error method with a robust norm, such as the Huber or Vapnik norm. A cross validation scheme could also be used to handle the automatic order determination in this setting by an exhaustive search of the model set.

5.2 Robustness to outliers and missing data

We continue by evaluating the proposed models and inference algorithms in the presence of missing data or outliers in the observations. The hypothesis is that, due to the use of Student's t innovations in the model, we should be more robust to such data anomalies than an LS estimate (based on a Gaussian assumption).

In these experiments, the innovations used in the data generation are drawn from a Gaussian distribution with unit variance. We then add outliers or missing observations to the outputs of the systems (i.e. this can be interpreted as an effect of sensor imperfections or measurement noise). This is done by randomly selecting between 1–3 % of the observations in the estimation data, which are modified as

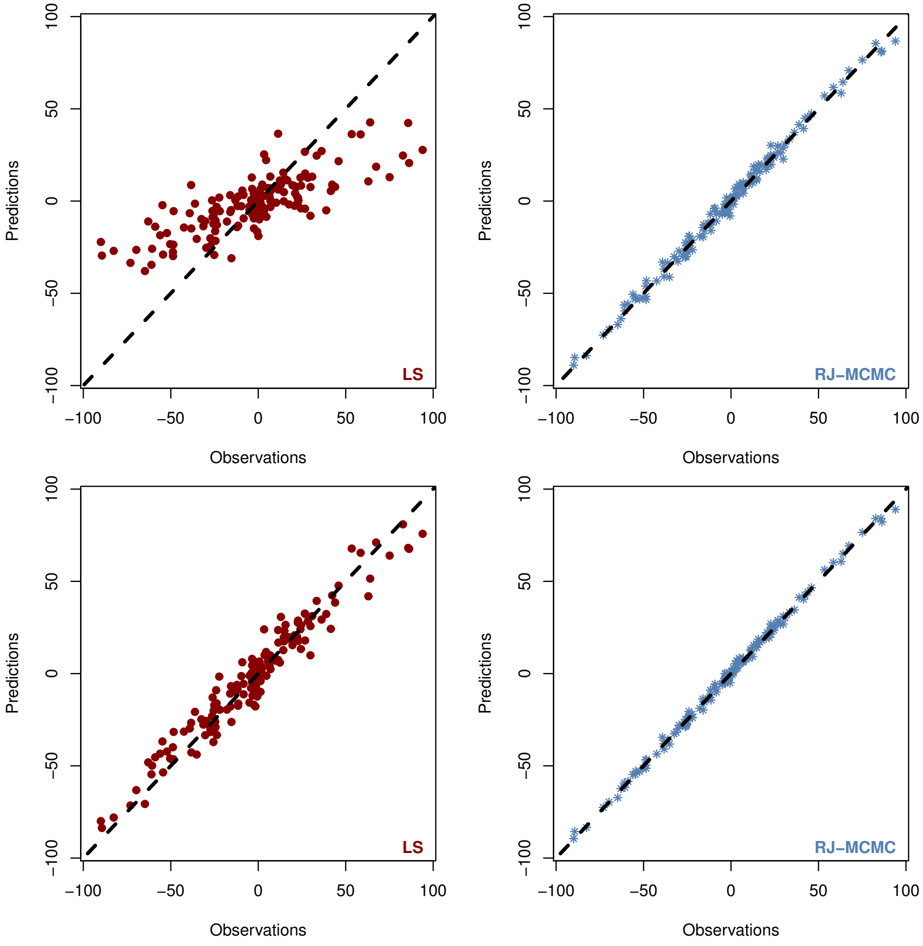

Figure 2: *Predictions versus observations for data with outliers (upper) and data with missing observations (lower). The model fit values for the outlier data example are* 91.6% *for the RJ-MCMC (blue stars) and* 40.2% *for LS (red dots). The corresponding values for the missing data example are* 94.4% *and* 75.7%.

	Outliers		Missing data	
Method	Mean	CI	Mean	CI
LS	39.13	[37.86 40.41]	75.20	[74.00 76.40]
RJ-MCMC	70.54	[69.03 72.04]	80.18	[78.74 81.62]
ARD-MCMC	72.46	[71.02 73.91]	81.57	[80.24 82.90]

Table 2: *The mean and 95% CIs for the model fit (in percent) from* $1,000$ *systems with outliers and missing data, respectively.*

described below. In the first set of experiments we add outliers to the selected observations. The size of the outliers are sampled from a uniform distribution $\mathcal{U}(-5y^+, 5y^+)$, with $y^+ = \max |y_t|$. In the second set of experiment, we instead replace the selected observations by zero-mean Gaussian noise with variance 0.01. This is to represent missing data due to sensor errors, resulting in values close to zero compared with the actual observations.

For each scenario, we generate $1,000$ random ARX systems and simulate $T = 450$ observations from each. We then apply the proposed MCMC samplers and LS with cross validation, similarly to the previous sections, but with the modifications described above. Table 2 gives the average results over the $1,000$ randomly generated models with added outliers and missing values, respectively. Here, we have not corrupted the test data by adding outliers or missing observations, not to overshadow the results[3].

The mean results show statistically certain differences between the LS approach and the two proposed methods. We conclude that, in general the proposed MCMC based methods are more robust to data anomalies such as missing observations or outliers.

In Figure 2, the predicted versus the corresponding observed data points are shown for the RJ-MCMC method (blue stars) and the LS approach (red dots), for two of the data batches. It is clearly visible that the LS method is unable to handle the problem with outliers, and the predictions are systematically too small (in absolute value). LS performs better in the situation with missing data, but the variance of the prediction errors is still clearly larger than for the RJ-MCMC method.

5.3 Real EEG data

We now present some results from real world EEG data, which often include large outliers (and therefore deviates from normality). Therefore this data serves as a good example for when the propose methods are useful in a practical setting. The deviations from normality can be seen in Figure 3, by observing the signal and the QQ-plot, i.e. a comparison between two distributions by plotting their quantiles against each other (Wilk and Gnanadesikan, 1968).

[3] If an outlier is added to the test data, the model fit can be extremely low even if there is a good fit for all time points apart from the one where the outlier occurs.

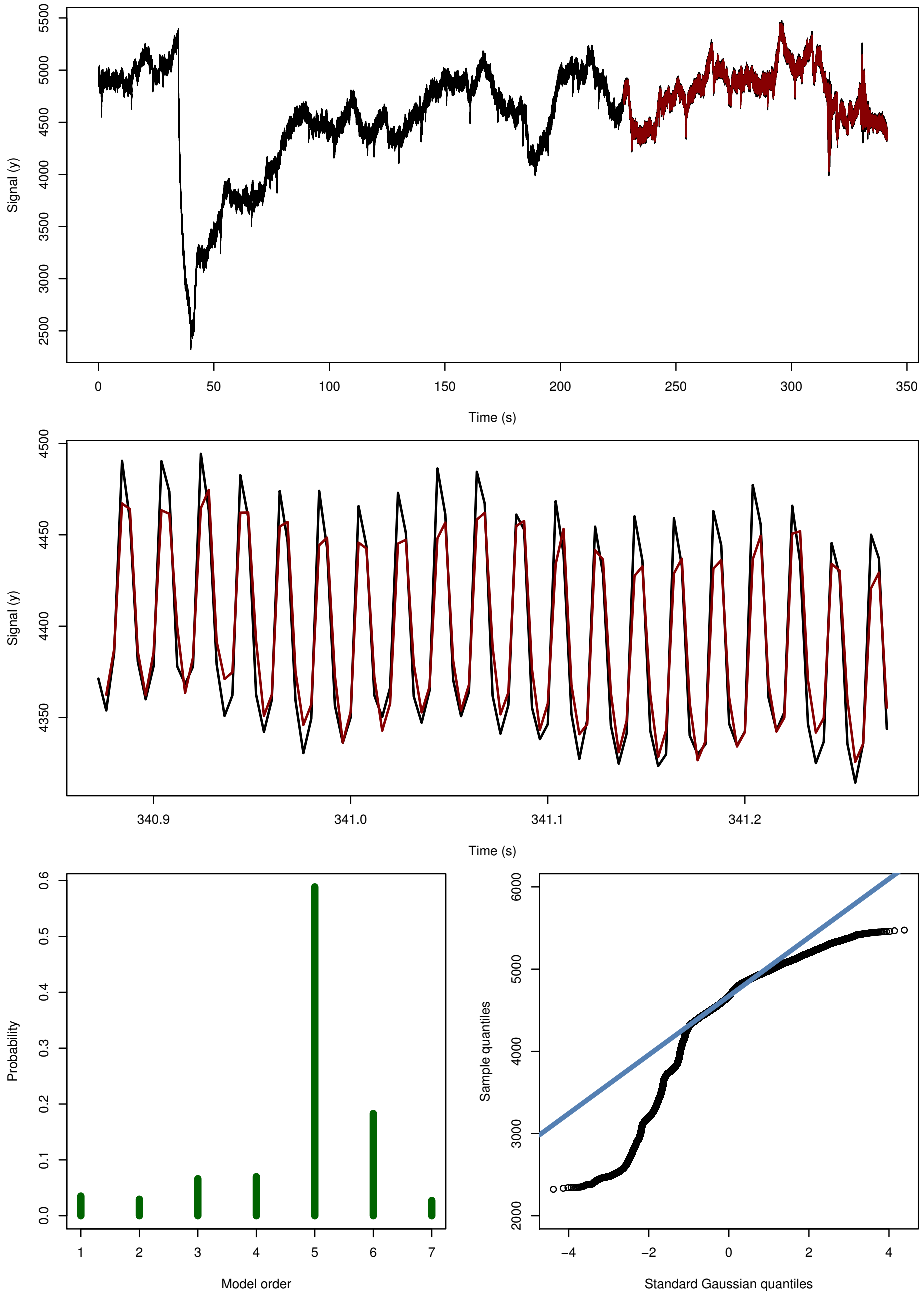

Figure 3: *Upper: the EEG signal (black) collected on one specific channel and patient with the one-step-ahead predictions (red). Middle: the last* 100 *samples from the upper graph. Lower left: The estimated posterior model order density from the RJ-MCMC method. Lower right: The QQ-plot for the data set. The model fit for the results in this figure is* 85.6%.

The RJ-MCMC method with Student's t innovations is used to estimate an AR model for this data set. The resulting estimated posterior density for the model order is shown in the lower left part of Figure 3. Knowing this posterior, allows for e.g. weighting several different models together using the estimated density values.

In addition, we can also estimate the posterior density of the DOF of the innovations. This density is useful for quantifying deviations from normality, as the Gaussian distribution is asymptotically recovered from the Student's t distribution with infinite DOF. As the maximum posterior value is attained at approximately 4.0 DOF, this confirm non-Gaussian innovations.

We have thereby illustrated the usefulness of the proposed methods, both for parameter inference but also for estimating useful posterior densities not easily obtainable in the LS framework.

6 Conclusions and Future work

We have considered hierarchical Bayesian ARX model with Student's t distributed innovations. This was considered to be able to capture non-Gaussian elements in the data and to increase robustness. Furthermore, both models contain a mechanism for automatic order selection. To perform inference in these models, we also derived two MCMC samplers: a reversible jump MCMC (RJ-MCMC) sampler and a standard Gibbs sampler.

Three numerical examples have been presented, providing evidence that the proposed models provide increased robustness to data anomalies, such as outliers and missing data. We have shown that the proposed methods perform on average as good as (ARD-MCMC) or better (RJ-MCMC) than LS with cross validation, when the true system is in the model class. Another benefit with the proposed methods is that they provide a type of information which is not easily attainable using more standard techniques. As an example, this can be the posterior distribution over the model orders, as illustrated in Figure 3.

There are several interesting avenues for future research, and we view the present work as a stepping stone for estimating more complex models. The next step is to generalize the proposed methods to encompass e.g. OE and ARMAX models. A more far reaching step is to generalize the methods to nonlinear systems, possibly by using Particle MCMC methods (Andrieu et al., 2010). It is also interesting to further analyse the use of sparseness priors in this setting.

Acknowledgements

The EEG data was kindly provided by Eline Borch Petersen and Thomas Lunner at Eriksholm Research Centre, Oticon A/S, Denmark.

Bibliography

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, USA, 2006.

S. P. Brooks, P. Giudici, and G. O. Roberts. Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(1):3–55, February 2003.

J. Christmas and R. Everson. Robust autoregression: Student-t innovations using variational Bayes. *IEEE Transactions on Signal Processing*, 59(1):48–57, 2011.

J. Dahlin, F. Lindsten, T. B. Schön, and A. Wills. Hierarchical Bayesian ARX models for robust inference. In *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*, Brussels, Belgium, July 2012.

S. Godsill. On the relationship between Markov chain Monte Carlo methods for model uncertainty. *Journal of Computational and Graphical Statistics*, 10(2): 230–248, 2001.

P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrica*, 82(4):711–732, 1995.

L. Ljung. *System identification: theory for the user*. Prentice Hall, 1999.

D. J. C. MacKey. Bayesian non-linear modelling for the prediction competition. *ASHRAE Transactions*, 100(2):1053–1062, 1994.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2 edition, 2004.

L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

P. T. Troughton and S. J. Godsill. A reversible jump sampler for autoregressive time series. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998.

M. B. Wilk and R. Gnanadesikan. Probability plotting methods for the analysis of data. *Biometrika*, 55(1):1–17, March 1968.

Licentiate Theses
Division of Automatic Control
Linköping University

P. Andersson: Adaptive Forgetting through Multiple Models and Adaptive Control of Car Dynamics. Thesis No. 15, 1983.

B. Wahlberg: On Model Simplification in System Identification. Thesis No. 47, 1985.

A. Isaksson: Identification of Time Varying Systems and Applications of System Identification to Signal Processing. Thesis No. 75, 1986.

G. Malmberg: A Study of Adaptive Control Missiles. Thesis No. 76, 1986.

S. Gunnarsson: On the Mean Square Error of Transfer Function Estimates with Applications to Control. Thesis No. 90, 1986.

M. Viberg: On the Adaptive Array Problem. Thesis No. 117, 1987.

K. Ståhl: On the Frequency Domain Analysis of Nonlinear Systems. Thesis No. 137, 1988.

A. Skeppstedt: Construction of Composite Models from Large Data-Sets. Thesis No. 149, 1988.

P. A. J. Nagy: MaMiS: A Programming Environment for Numeric/Symbolic Data Processing. Thesis No. 153, 1988.

K. Forsman: Applications of Constructive Algebra to Control Problems. Thesis No. 231, 1990.

I. Klein: Planning for a Class of Sequential Control Problems. Thesis No. 234, 1990.

F. Gustafsson: Optimal Segmentation of Linear Regression Parameters. Thesis No. 246, 1990.

H. Hjalmarsson: On Estimation of Model Quality in System Identification. Thesis No. 251, 1990.

S. Andersson: Sensor Array Processing; Application to Mobile Communication Systems and Dimension Reduction. Thesis No. 255, 1990.

K. Wang Chen: Observability and Invertibility of Nonlinear Systems: A Differential Algebraic Approach. Thesis No. 282, 1991.

J. Sjöberg: Regularization Issues in Neural Network Models of Dynamical Systems. Thesis No. 366, 1993.

P. Pucar: Segmentation of Laser Range Radar Images Using Hidden Markov Field Models. Thesis No. 403, 1993.

H. Fortell: Volterra and Algebraic Approaches to the Zero Dynamics. Thesis No. 438, 1994.

T. McKelvey: On State-Space Models in System Identification. Thesis No. 447, 1994.

T. Andersson: Concepts and Algorithms for Non-Linear System Identifiability. Thesis No. 448, 1994.

P. Lindskog: Algorithms and Tools for System Identification Using Prior Knowledge. Thesis No. 456, 1994.

J. Plantin: Algebraic Methods for Verification and Control of Discrete Event Dynamic Systems. Thesis No. 501, 1995.

J. Gunnarsson: On Modeling of Discrete Event Dynamic Systems, Using Symbolic Algebraic Methods. Thesis No. 502, 1995.

A. Ericsson: Fast Power Control to Counteract Rayleigh Fading in Cellular Radio Systems. Thesis No. 527, 1995.

M. Jirstrand: Algebraic Methods for Modeling and Design in Control. Thesis No. 540, 1996.

K. Edström: Simulation of Mode Switching Systems Using Switched Bond Graphs. Thesis No. 586, 1996.

J. Palmqvist: On Integrity Monitoring of Integrated Navigation Systems. Thesis No. 600, 1997.
A. Stenman: Just-in-Time Models with Applications to Dynamical Systems. Thesis No. 601, 1997.
M. Andersson: Experimental Design and Updating of Finite Element Models. Thesis No. 611, 1997.
U. Forssell: Properties and Usage of Closed-Loop Identification Methods. Thesis No. 641, 1997.
M. Larsson: On Modeling and Diagnosis of Discrete Event Dynamic systems. Thesis No. 648, 1997.
N. Bergman: Bayesian Inference in Terrain Navigation. Thesis No. 649, 1997.
V. Einarsson: On Verification of Switched Systems Using Abstractions. Thesis No. 705, 1998.
J. Blom, F. Gunnarsson: Power Control in Cellular Radio Systems. Thesis No. 706, 1998.
P. Spångéus: Hybrid Control using LP and LMI methods – Some Applications. Thesis No. 724, 1998.
M. Norrlöf: On Analysis and Implementation of Iterative Learning Control. Thesis No. 727, 1998.
A. Hagenblad: Aspects of the Identification of Wiener Models. Thesis No. 793, 1999.
F. Tjärnström: Quality Estimation of Approximate Models. Thesis No. 810, 2000.
C. Carlsson: Vehicle Size and Orientation Estimation Using Geometric Fitting. Thesis No. 840, 2000.
J. Löfberg: Linear Model Predictive Control: Stability and Robustness. Thesis No. 866, 2001.
O. Härkegård: Flight Control Design Using Backstepping. Thesis No. 875, 2001.
J. Elbornsson: Equalization of Distortion in A/D Converters. Thesis No. 883, 2001.
J. Roll: Robust Verification and Identification of Piecewise Affine Systems. Thesis No. 899, 2001.
I. Lind: Regressor Selection in System Identification using ANOVA. Thesis No. 921, 2001.
R. Karlsson: Simulation Based Methods for Target Tracking. Thesis No. 930, 2002.
P.-J. Nordlund: Sequential Monte Carlo Filters and Integrated Navigation. Thesis No. 945, 2002.
M. Östring: Identification, Diagnosis, and Control of a Flexible Robot Arm. Thesis No. 948, 2002.
C. Olsson: Active Engine Vibration Isolation using Feedback Control. Thesis No. 968, 2002.
J. Jansson: Tracking and Decision Making for Automotive Collision Avoidance. Thesis No. 965, 2002.
N. Persson: Event Based Sampling with Application to Spectral Estimation. Thesis No. 981, 2002.
D. Lindgren: Subspace Selection Techniques for Classification Problems. Thesis No. 995, 2002.
E. Geijer Lundin: Uplink Load in CDMA Cellular Systems. Thesis No. 1045, 2003.
M. Enqvist: Some Results on Linear Models of Nonlinear Systems. Thesis No. 1046, 2003.
T. Schön: On Computational Methods for Nonlinear Estimation. Thesis No. 1047, 2003.
F. Gunnarsson: On Modeling and Control of Network Queue Dynamics. Thesis No. 1048, 2003.
S. Björklund: A Survey and Comparison of Time-Delay Estimation Methods in Linear Systems. Thesis No. 1061, 2003.

M. Gerdin: Parameter Estimation in Linear Descriptor Systems. Thesis No. 1085, 2004.

A. Eidehall: An Automotive Lane Guidance System. Thesis No. 1122, 2004.

E. Wernholt: On Multivariable and Nonlinear Identification of Industrial Robots. Thesis No. 1131, 2004.

J. Gillberg: Methods for Frequency Domain Estimation of Continuous-Time Models. Thesis No. 1133, 2004.

G. Hendeby: Fundamental Estimation and Detection Limits in Linear Non-Gaussian Systems. Thesis No. 1199, 2005.

D. Axehill: Applications of Integer Quadratic Programming in Control and Communication. Thesis No. 1218, 2005.

J. Sjöberg: Some Results On Optimal Control for Nonlinear Descriptor Systems. Thesis No. 1227, 2006.

D. Törnqvist: Statistical Fault Detection with Applications to IMU Disturbances. Thesis No. 1258, 2006.

H. Tidefelt: Structural algorithms and perturbations in differential-algebraic equations. Thesis No. 1318, 2007.

S. Moberg: On Modeling and Control of Flexible Manipulators. Thesis No. 1336, 2007.

J. Wallén: On Kinematic Modelling and Iterative Learning Control of Industrial Robots. Thesis No. 1343, 2008.

J. Harju Johansson: A Structure Utilizing Inexact Primal-Dual Interior-Point Method for Analysis of Linear Differential Inclusions. Thesis No. 1367, 2008.

J. D. Hol: Pose Estimation and Calibration Algorithms for Vision and Inertial Sensors. Thesis No. 1370, 2008.

H. Ohlsson: Regression on Manifolds with Implications for System Identification. Thesis No. 1382, 2008.

D. Ankelhed: On low order controller synthesis using rational constraints. Thesis No. 1398, 2009.

P. Skoglar: Planning Methods for Aerial Exploration and Ground Target Tracking. Thesis No. 1420, 2009.

C. Lundquist: Automotive Sensor Fusion for Situation Awareness. Thesis No. 1422, 2009.

C. Lyzell: Initialization Methods for System Identification. Thesis No. 1426, 2009.

R. Falkeborn: Structure exploitation in semidefinite programming for control. Thesis No. 1430, 2010.

D. Petersson: Nonlinear Optimization Approaches to $\mathcal{H}_2$-Norm Based LPV Modelling and Control. Thesis No. 1453, 2010.

Z. Sjanic: Navigation and SAR Auto-focusing in a Sensor Fusion Framework. Thesis No. 1464, 2011.

K. Granström: Loop detection and extended target tracking using laser data. Thesis No. 1465, 2011.

J. Callmer: Topics in Localization and Mapping. Thesis No. 1489, 2011.

F. Lindsten: Rao-Blackwellised particle methods for inference and identification. Thesis No. 1480, 2011.

M. Skoglund: Visual Inertial Navigation and Calibration. Thesis No. 1500, 2011.

S. Khoshfetrat Pakazad: Topics in Robustness Analysis. Thesis No. 1512, 2011.

P. Axelsson: On Sensor Fusion Applied to Industrial Manipulators. Thesis No. 1511, 2011.

A. Carvalho Bittencourt: On Modeling and Diagnosis of Friction and Wear in Industrial Robots. Thesis No. 1516, 2012.

P. Rosander: Averaging level control in the presence of frequent inlet flow upsets . Thesis No. 1527, 2012.

N. Wahlström: Localization using Magnetometers and Light Sensors. Thesis No. 1581, 2013.

R. Larsson: System Identification of Flight Mechanical Characteristics. Thesis No. 1599, 2013.

Y. Jung: Estimation of Inverse Models Applied to Power Amplifier Predistortion. Thesis No. 1605, 2013.

M. Syldatk: On Calibration of Ground Sensor Networks. Thesis No. 1611, 2013.

M. Roth: Kalman Filters for Nonlinear Systems and Heavy-Tailed Noise. Thesis No. 1613, 2013.

D. Simon: Model Predictive Control in Flight Control Design — Stability and Reference Tracking. Thesis No. 1642, 2014.

J. Dahlin: Sequential Monte Carlo for inference in nonlinear state space models. Thesis No. 1652, 2014.