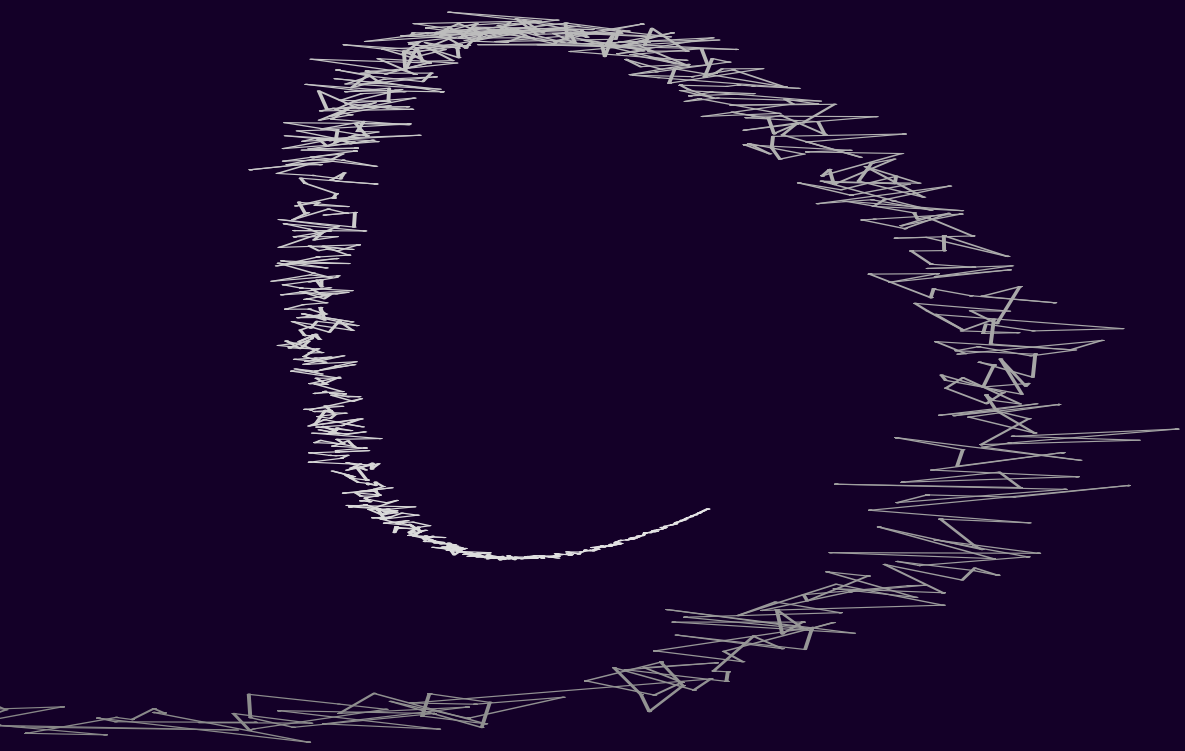
Linköping studies in science and technology. Dissertations.
No. 1754



Accelerating Monte Carlo methods for Bayesian inference in dynamical models

Johan Dahlin

LINKÖPING UNIVERSITY

Linköping studies in science and technology. Dissertations.
No. 1754

Accelerating Monte Carlo methods for Bayesian inference in dynamical models

Johan Dahlin

LINKÖPING UNIVERSITY

Cover illustration: A Markov chain generated by the Metropolis-Hastings algorithm with an autoregressive proposal on a manifold given by a parametric function.

This thesis was typeset using the LaTeX typesetting system originally developed by Leslie Lamport, based on TeX created by Donald Knuth. The text is set in Garamond and Cabin. The source code is set in Inconsolata. All plots are made using R (R Core Team, 2015) together with colors from the RColorBrewer package (Neuwirth, 2014). Most simulations are carried out in R and Python with the exception of Paper F and H.

Linköping studies in science and technology. Dissertations.
No. 1754

Accelerating Monte Carlo methods for Bayesian inference in dynamical models

Johan Dahlin

johan.dahlin@liu.se
liu@johandahlin.com
http://liu.johandahlin.com
Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping
Sweden

ISBN 978-91-7685-797-7 ISSN 0345-7524

Printed by LiU-Tryck, Linköping, Sweden 2016

Denna avhandling tillägnas min familj!

Abstract

Making decisions and predictions from noisy observations are two important and challenging problems in many areas of society. Some examples of applications are recommendation systems for online shopping and streaming services, connecting genes with certain diseases and modelling climate change. In this thesis, we make use of Bayesian statistics to construct probabilistic models given prior information and historical data, which can be used for decision support and predictions. The main obstacle with this approach is that it often results in mathematical problems lacking analytical solutions. To cope with this, we make use of statistical simulation algorithms known as Monte Carlo methods to approximate the intractable solution. These methods enjoy well-understood statistical properties but are often computational prohibitive to employ.

The main contribution of this thesis is the exploration of different strategies for accelerating inference methods based on sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC). That is, strategies for reducing the computational effort while keeping or improving the accuracy. A major part of the thesis is devoted to proposing such strategies for the MCMC method known as the particle Metropolis-Hastings (PMH) algorithm. We investigate two strategies: (i) introducing estimates of the gradient and Hessian of the target to better tailor the algorithm to the problem and (ii) introducing a positive correlation between the point-wise estimates of the target.

Furthermore, we propose an algorithm based on the combination of SMC and Gaussian process optimisation, which can provide reasonable estimates of the posterior but with a significant decrease in computational effort compared with PMH. Moreover, we explore the use of sparseness priors for approximate inference in over-parametrised mixed effects models and autoregressive processes. This can potentially be a practical strategy for inference in the big data era. Finally, we propose a general method for increasing the accuracy of the parameter estimates in non-linear state space models by applying a designed input signal.

Populärvetenskaplig sammanfattning

Borde Riksbanken höja eller sänka reporäntan vid sitt nästa möte för att nå inflationsmålet? Vilka gener är förknippade med en viss sjukdom? Hur kan Netflix och Spotify veta vilka filmer och vilken musik som jag vill lyssna på härnäst?

Dessa tre problem är exempel på frågor där statistiska modeller kan vara användbara för att ge hjälp och underlag för beslut. Statistiska modeller kombinerar teoretisk kunskap om exempelvis det svenska ekonomiska systemet med historisk data för att ge prognoser av framtida skeenden. Dessa prognoser kan sedan användas för att utvärdera exempelvis vad som skulle hända med inflationen i Sverige om arbetslösheten sjunker eller hur värdet på mitt pensionssparande förändras när Stockholmsbörsen rasar. Tillämpningar som dessa och många andra gör statistiska modeller viktiga för många delar av samhället.

Ett sätt att ta fram statistiska modeller bygger på att kontinuerligt uppdatera en modell allteftersom mer information samlas in. Detta angreppssätt kallas för Bayesiansk statistik och är särskilt användbart när man sedan tidigare har bra insikter i modellen eller tillgång till endast lite historisk data för att bygga modellen. En nackdel med Bayesiansk statistik är att de beräkningar som krävs för att uppdatera modellen med den nya informationen ofta är mycket komplicerade. I sådana situationer kan man istället simulera utfallet från miljontals varianter av modellen och sedan jämföra dessa mot de historiska observationerna som finns till hands. Man kan sedan medelvärdesbilda över de varianter som gav bäst resultat för att på så sätt ta fram en slutlig modell. Det kan därför ibland ta dagar eller veckor för att ta fram en modell. Problemet blir särskilt stort när man använder mer avancerade modeller som skulle kunna ge bättre prognoser men som tar för lång tid för att bygga.

I denna avhandling använder vi ett antal olika strategier för att underlätta eller förbättra dessa simuleringar. Vi föreslår exempelvis att ta hänsyn till fler insikter om systemet och därmed minska antalet varianter av modellen som behöver undersökas. Vi kan således redan utesluta vissa modeller eftersom vi har en bra uppfattning om ungefär hur en bra modell ska se ut. Vi kan också förändra simuleringen så att den enklare rör sig mellan olika typer av modeller. På detta sätt utforskas rymden av alla möjliga modeller på ett mer effektivt sätt. Vi föreslår ett antal olika kombinationer och förändringar av befintliga metoder för att snabba upp anpassningen av modellen till observationerna. Vi visar att beräkningstiden i vissa fall kan minska ifrån några dagar till någon timme. Förhoppningsvis kommer detta i framtiden leda till att man i praktiken kan använda mer avancerade modeller som i sin tur resulterar i bättre prognoser och beslut.

Acknowledgments

> Science is a co-operative enterprise, spanning the generations. It's the passing of a torch from teacher to student to teacher. A community of minds reaching back from antiquity and forward to the stars. – *Neil deGrasse Tyson*

This is my humble contribution to the collaboration that is science. This is my dent in the universe! However, I could not have reached this point and written this thesis without the support, encouragement and love from so many people over the years. We all have so much to be grateful for. We often do not have the opportunity to express this and often take things for granted. Therefore, please bare with me on the following pages in my attempt to express my gratitude for all the people that made this journey possible.

To do a PHD means that you spend five years on the boundary of your comfort zone. Sometimes, you are on the inside of the boundary but often you are just (or even further) outside the boundary. The latter is an awesome place to be. There is nothing that develops you more than when you stretch the limits of what you think that you can achieve. However, staying at this place for a long time takes its toll and this is one of the reasons (except of course learning how to do research) for having a guide and mentor along for the journey.

In my case, I got the opportunity to travel along my two amazing supervisors Thomas Schön and Fredrik Lindsten. These guys are really great supervisors and they have skilfully guided me along the way to obtain my PHD. I am truly grateful for all the time, effort and energy that they have put into helping me develop as a researcher and as a person. Thomas has helped me a lot with the long-term perspective with strategy, planning, collaborations and research ideas. Fredrik has helped me with many good ideas, answering hundreds of questions regarding the intricate working of algorithms and helped me iron out subtle mistakes in papers and reports. Thank you also for all the nice times together outside of work. Especially all the running, restaurant visits and team day dinners as Thomas' place!

Along my journey, I crossed paths with Mattias Villani and Robert Kohn, who supported and guided me almost as if I was one of their own PHD students. I am very grateful for our collaborations and the time, inspiration and knowledge you both have given me. A special thanks goes to Robert for the invitation to visit him at UNSW Business School in Sydney, Australia. The autumn that I spent there was truly a wonderful experience in terms of research as well as from a personal perspective. Thank you Robert for you amazing hospitality, your patience and encouragement.

I would like to thank all my co-authors during my time at Linköping University for some wonderful and fruitful collaborations: Christian Andersson Naesseth, Liang Dai, Daniel Hultqvist, Daniel Jönsson, Manon Kok, Robert Kohn, Joel Kronander, Fredrik Lindsten, Cristian Rojas, Jakob Roll, Thomas Schön, Andreas Svensson, Fredrik Svensson, Jonas Unger, Patricio Valenzuela, Mattias Villani, Johan Wågberg and Adrian Wills. Furthermore, many of these co-authors and Olof Sundin helped with proof-reading the thesis and contributed with suggestions to improve it. All remaining errors are entirely my own.

To be able to write a good thesis you require a good working environment. Svante Gunnarsson and Ninna Stensgård are two very important persons in this effort. Thank you for all your support and helpfulness in all matters to help create the best possible situation for

myself and for all the other PHD students. Furthermore, I gratefully acknowledge the financial support from the projects *Learning of complex dynamical systems* (Contract number: 637-2014-466) and *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) and CADICS, a Linnaeus Center, all funded by the Swedish Research Council. 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 (partially) written.

Another aspect of the atmosphere at work is all my wonderful colleagues. My room mate from the early years Michael Roth was always there to discuss work and to keep my streak of perfectionism in check. My friendships with Jonas Linder and Manon Kok have also meant a lot to me. We joined the group at the same time and have spent many hours together both at work and during our spare time. Thank you for your positive attitudes and for all the fun times exploring Beijing, Cape Town, Vancouver, Varberg and France together.

Furthermore, thank you Sina Khoshfetrat Pakazad for arranging all the nice BBQs and for always being up for discussing politics. It is also important to get some fresh air and see the sun during the long days at work. Hanna Nyqvist has been my loyal companion during our many lunch walks together. Oskar Ljungqvist recently joined the group but has quickly become a good friend. Thank you for all our lunch runs (jogs) together, for encouraging and inspiring my running and for all the nice discussions!

Furthermore, I would like to thank my remaining friends and colleagues at the group. Especially, (without any specific ordering) Christian Andersson Naesseth, Christian Lyzell, Ylva Jung, Isak Nielsen, Daniel Petersson, Zoran Sjanic, Niklas Wahlström, Clas Veibäck and Emre Özkan for all the fun things that we have done together. This includes everything from wine trips in South Africa, wonderful food in France and ordering 30 dumplings each in Beijing to hitting some shallows on open sea with a canoe, cross country skiing in Chamonix and riding the local bus to the Great Wall of China. It has been great fun!

Along the years, I have also spent a fair amount of time at other research groups and with the PHD students within them. A big thanks goes out to the Systems and Control at Uppsala University, Automatic Control at KTH, Economics at UNSW and Statistics and Machine learning at Linköping University. Thank you for your hospitality and for all our research discussions. I especially would like to thank Joel Kronander, Christian Larsson, Andreas Svensson, Soma Tayamon, Patricio Valenzuela and Johan Wågberg for all the good times travelling the world together.

Finally, my family and friends outside of work are a great source of support, inspiration and encouragement. My family is always there when needed with love and kindness as well as to help with all possible practical matters. My friends always provide refuge from work when the stress levels are high and the motivation falters. Thank you all for believing in me and in supporting me even when (at times) I did not myself. I hope we all can spend some more time together in the years to come. I love you all and you mean the world to me!

Linköping, March 21, 2016
Johan Dahlin

Contents

I Background

II Papers

Part I

Background

1 Introductory overview

Modelling of dynamical systems is an integrate part of modern science. Two major applications are to describe some observed data and to make forecasts about the future behaviour of a system. The latter is an essential ingredient in making decision from noisy observations in many areas such as business, economics, engineering and medicine. A standard approach for forecasting and decision making is to make use of *probabilistic models* (Ghahramani, 2015), which are created by combining some pre-defined model with observations from the true system. This approach is also known as *data-driven modelling* and is probably the most popular alternative for decision support today.

The probabilistic model is usually constructed by making use of statistical inference. One such framework is Bayesian statistics, which allows for sequentially updating the model as more observations arrive. Another benefit is that the uncertainty regarding the model can be included when making decisions. However, a major problem with Bayesian inference is that model updates, prediction and other computations often are posed as intractable integrals. Hence, these cannot be computed in closed-form and approximations are required.

A typical example is to compute the mean of the so-called posterior distribution $\pi(x \,|\, y)$, which encodes our prior beliefs of some quantity $x \in \mathcal{X}$ and the information in some data denoted by y. The posterior mean can be computed by

$$\mathbb{E}_\pi[x] = \int_{\mathcal{X}} x \, \pi(x \,|\, y) \, \mathrm{d}x,$$

where the dimension of the problem is determined by the dimension of x. Hence, this can correspond to a high-dimensional integration problem, which is difficult to approximate using numerical methods such as curvatures.

Instead, Monte Carlo methods or variational inference are often applied to approximate the update by statistical simulation and analytical approximations, respectively. In this thesis,

we focus on the former family of methods, which is based on generating a large number of random scenarios or outcomes. Hence, Monte Carlo algorithms are often computational intensive and can require days or weeks to run. This is especially a problem for dynamical models and this thesis is devoted to try to decrease the time that is required to implement and execute these algorithms while keeping their accuracy.

Examples of applications

We begin by presenting a few applications where acceleration of Monte Carlo methods could be important. As previously discussed, these methods are essential for Bayesian inference in dynamical models, which themselves have applications in many different fields. One example is platooning, where trucks are driven as a group to reduce the air drag and therefore to increase fuel-efficiency. This is possible by using e.g., model predictive control (MPC; Mayne, 2014) to control the trucks to keep a certain distance, see Turri et al. (2015). Here, an accurate model is important as it is used to forecast future outcomes. Bayesian modelling can be useful in this setting as it also can take into account the uncertainty of model when computing predictions.

In recommendation systems, probabilistic modelling is important to provide suggestions to the user, see Stern et al. (2009). Many online services such as Netflix, Spotify and Amazon are already using such systems to improve customer satisfaction and to increase sales. This problem is interesting because companies such as Amazon and Google have a massive collection of information at their disposal. However, the amount of information regarding a particular user can be quite limited, especially when the user is new to the service. Finding patterns connecting this user to other users on the site is therefore important to be able to pool the data and to provide good suggestions. It is also useful to take the dynamics into account as user preferences can change over time. This was one of the key insights incorporated into the winning algorithm in the Netflix Prize competition. The winning approach proposed by Koren (2009) was awarded one million US dollars by the company.

Climate change and global warming are two big challenges for human kind to solve during this century. Bayesian inference is useful in this setting to e.g., pool the output from different climate models together as discussed by Monteleoni et al. (2011). Again, the ability to take uncertainty into account is important in this setting as well, see Birks (1998). Most natural disasters are quite rare and modelling them is difficult due to the small amounts of data. Bayesian methods can therefore be useful in this setting to estimate probabilities of rare events such as wild fires, see Xue et al. (2012)

Probabilistic modelling is also useful in genomics to fight disease and other health problems, see Bush and Moore (2012) and Rasmussen et al. (2011). A major challenge in this field is to find patterns and structures connecting genes with e.g., cancer, diabetes and heart disease. The massive amount of information makes inference difficult as many sophisticated methods are computationally prohibitive. However, this type of analysis could be useful for *personalised medicine* and *data-driven health care* if the computational challenges can be overcome, see Raghupathi and Raghupathi (2014). Another interesting application in this field is reconstruct the lineage of different species using Phylogenetic trees, see Larget and Simon (1999) and Bouchard-Côté et al. (2012).

We continue with introducing two more applications of probabilistic modelling connected to Monte Carlo methods in the subsequent sections. Two further examples are introduced in Chapter 2 and these follow us throughout the introductory part of this thesis to illustrate important concepts. Finally, more real-world examples are presented in the papers included in Part II of this thesis.

Reconstructing the temperature of pre-historic Earth

In palaeoclimatology, ice varve thickness data is an important source of information to recreate the historical mean temperature on the Earth, which is useful for studying global warming. In the upper part of Figure 1.1, we present the thickness of ice varves (layers of sediments that are deposited from melting glaciers) from Shumway and Stoffer (2011). The silt and sand that are accumulated during each year makes up one varve and changes in the varve thickness indicate temperature changes. That is, thick varves are the result of warm and wet weather, whereas the opposite holds for cold and dry weather.

The data set contains the thickness of 634 ice varves formed at a location in Massachusetts, US between the years 9,883 and 9,250 BC. We note that the mean and the variance of the thickness seem to vary during the period but the data is quite noisy. Therefore, we would like to smooth the data to be able to determine if the variations in the thickness are statistically significant. In the middle part of Figure 1.1, we make use of a non-parametric Bayesian regression model known as the Gaussian process (GP; Rasmussen and Williams, 2006), which is further discussed in Section 2.4.1.

In the lower part of of the same figure, we present the result from using a parametric state space model (SSM) to smooth the data. We introduce the SSM in Section 2.2.3 and show how to make use of Monte Carlo methods to estimate the parameters of the SSM in Sections 3.2.2 and 3.3. In Figure 1.1, the resulting estimates of the mean thickness are presented as a solid line and the 95% confidence intervals are presented as the shaded areas. From this analysis, we conclude that there is no significant change in the mean thickness of the ice varves during this period.

We also note that the uncertainty in the parametric model is smaller and it better follows the data. The reason for this is that the GP model usually assumes homoscedastic variance, whereas the variance is allowed to change with time in the parametric model. However, the non-parametric model is simpler to estimate and it usually takes a couple of seconds on a modern computer. On the other hand, inference for the parameters in the SSM can take about an hour to complete. Therefore, there is a need to develop faster inference methods for non-linear SSMs. However, there are other non-parametric models that do not assume homoscedasticity (Le et al., 2005) and can handle heavy-tailed observations by assuming Student's t noise (Shah et al., 2014).

Rendering photorealistic images

In computer graphics, an important problem is to design good methods for rendering objects into an existing scene. This is typically used for special effects in Hollywood films and for advertisement. A standard approach for this is the *image-based lighting* (IBL) method

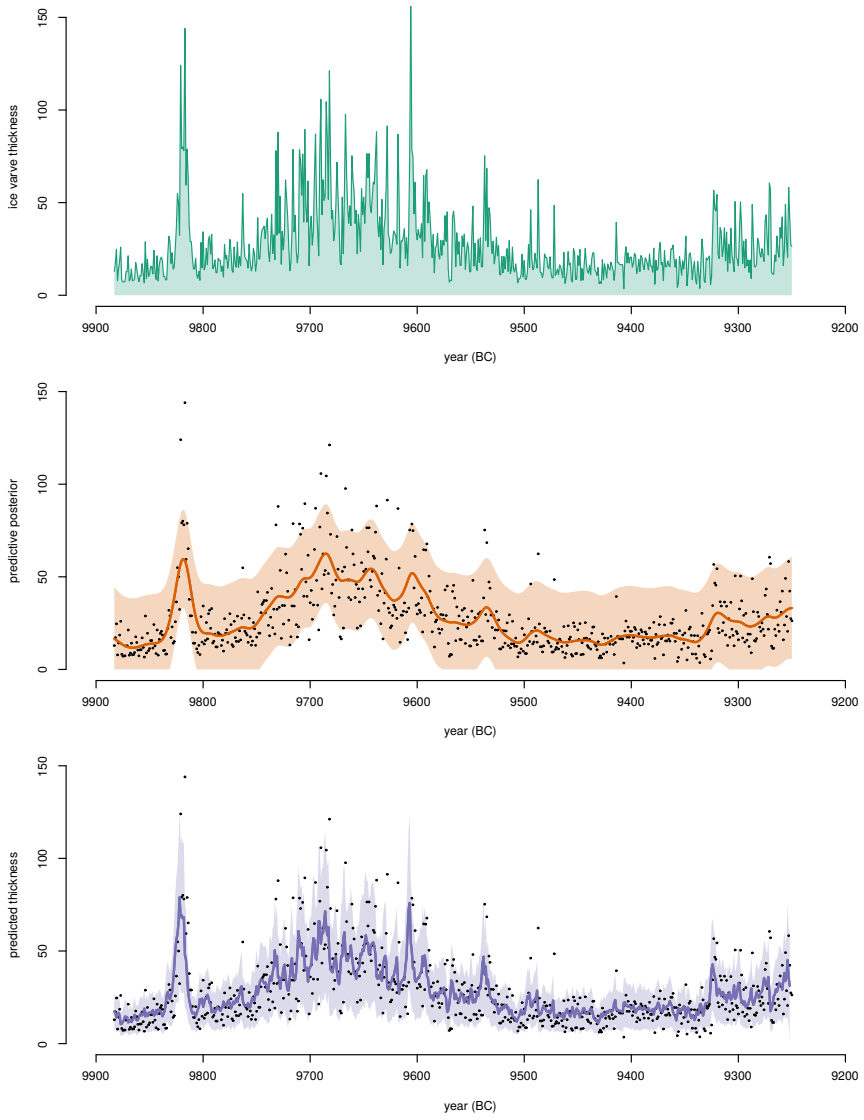


Figure 1.1. Upper: the thickness of ice varves formed at a location in Massachusetts between years 9,883 and 9,250 BC. A non-parametric model (middle) and parametric model (lower) of the thickness presented with the mean value (solid) lines and 95% confidence intervals (shaded areas). The dots indicate the original data.

(Debevec, 1998; Pharr and Humphreys, 2010), where a model of how light behaves is used in combination with information about the scene. The latter so-called *environment map* is often a panoramic image taken by a *high dynamic range* (HDR), which can record much larger variations in brightness than a standard camera. This is required to capture all the different light sources within the scene.

Two concrete examples of renderings using the IBL method are presented in Figures 1.2 and 1.3 taken from Kronander et al. (2014a) and Unger et al. (2013). Note that, we have rendered several photorealistic objects into the two scenes, such as a sphere, helicopter and some furniture. In Figure 1.3, we present the scene before (left) and after (right) adding the rendered furniture. This is a real-world example from IKEA catalogues in which scenes are often rendered using this technique to decrease the cost. The alternative would be to build kitchens and similar environments customized for different countries. The IBL method instead allows for taking an image of a basic set-up, which then can be augmented by computer rendering to create different country-specific variations of the complete scene.

To be able to make use of IBL, we additionally require a geometric description of the objects to be rendered and the properties of the different materials in the objects. All of this information is then combined using the *light transport equation* (LTE), which is a physical model expressed as an integral of how light rays propagates through space and reflects off surfaces. The LTE model cannot be solved analytically, but it can be approximated using Monte Carlo methods as it is an integral.

A further complication is that there are infinitely many rays that bounce around in the scene before they hit the pixels in the image plan. As a result, it is computationally infeasible to simulate all the possible light rays. Instead, we need to find an approach to only simulate the ones that contributes the most to the brightness and colour of each pixel in the image plane. This is especially a problem 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 environment maps are similar, this could lead to a decrease in the total computational cost. Hence, strategies for accelerating Monte Carlo methods could be useful in this context to improve the rendering times and decrease the cost of special effects in films. For more information, see Kronander et al. (2014a) and Ghosh et al. (2006).

Main contribution

We have now seen some specific examples of when dynamical models and Monte Carlo methods can be of use. As stated before, Monte Carlo methods are very useful and often acts as an enabler to solve otherwise intractable or infeasible problems. However, the main drawback is that the Monte Carlo methods have a large computational cost and this could limit their practical utility. Hence, accelerating Monte Carlo methods is an important endeavour with applications in may different domains. There are many different strategies to attain this goal proposed in the literature. These include solving an approximate but simpler problem, utilising parallel hardware such as graphical processing units (GPUs) and modifying the algorithms themselves. In this thesis, we focus on the first and the third approach by using a number of different strategies. This effort has resulted in:

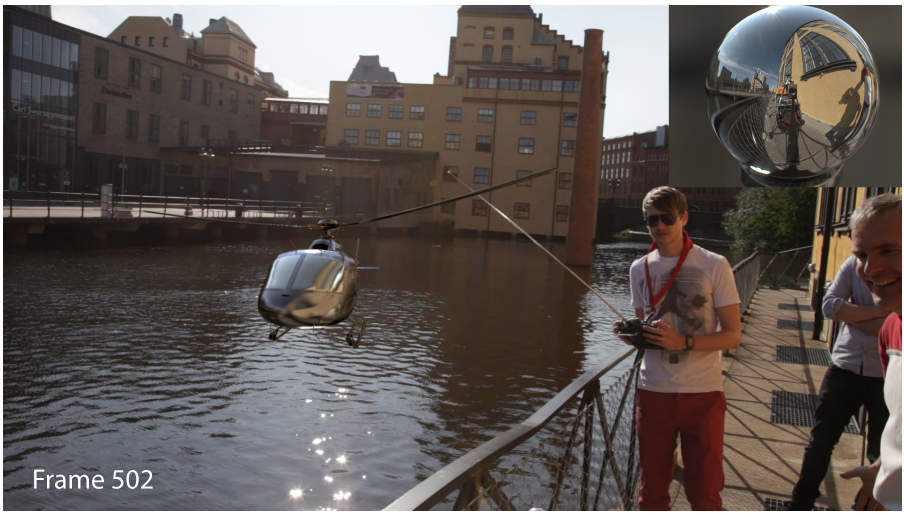


Figure 1.2. A helicopter and sphere rendered using sequential IBL (Kronander et al., 2014a). The image is part of Kronander et al. (2014a) first published in the Proceedings of the 22nd European Signal Processing Conference (EUSIPCO 2014) in 2014, published by EURASIP.

Figure 1.3. Scene from Unger et al. (2013) before (left) and after (right) rendering by IBL. These images are unaltered reproductions of the originals in Unger et al. (2013) and are used under a CC-NC-SA licence (`http://creativecommons.org/licenses/by-nc-sa/3.0/`). The original work is available via Elsevier at `http://dx.doi.org/10.1016/j.cag.2013.07.001`.

- A number of new alternative versions of the particle Metropolis-Hastings (PMH) algorithm, where we incorporate gradient and Hessian information about the target into the proposal. This results in better behaviour during the burn-in, improved mixing of the Markov chain and simplified tuning of the algorithm for the user. (Papers B and C).
- A method for introducing a positive correlation into the auxiliary variables generated in the pseudo-marginal Metropolis-Hastings (PMMH) algorithm for estimating the target. This results in around three times better mixing of the Markov chain for some models, which results in a similar decrease of the computational cost (Paper D).
- A method to perform approximate Bayesian inference in non-linear SSMs, which is especially useful when the likelihood is intractable. The proposed method gives similar estimates compared with the PMH algorithm but can reduce the computational time from days to about an hour. (Paper E).
- A pedagogical and self-contained introduction to the PHM algorithm with supporting software implementations in three different languages (Paper A).
- An evaluation of approximate inference in mixed effects models with a Bayesian mixture model for the heterogeneity of the random effects. (Paper G).
- A method for input design in non-linear SSMs (Paper H). The proposed method increases the accuracy in the parameter estimates by applying a carefully designed input signal to the system.
- An evaluation of two Bayesian ARX models capable of dealing with outliers by modelling the observations as Student's t distributed. The proposed inference methods also include automatic model order selection (Paper F).

Thesis outline

The thesis consists of two parts. The first part introduces some background material regarding modelling of dynamical data and different approaches for inference. We also highlight some problems with existing approaches and propose a number of strategies to mitigate these. These strategies are applied and evaluated in the second part of the thesis in a collection of scientific contributions both as peer-reviewed papers and technical reports.

Paper A

Paper A of this thesis is an edited version of

> J. Dahlin and T. B. Schön. Getting started with particle Metropolis-Hastings for inference in nonlinear models. *Pre-print*, 2015. arXiv:1511.01707v4.

Source code and data: `https://github.com/compops/pmh-tutorial`

Summary: We provide a gentle introduction to the PMH algorithm for parameter inference in non-linear SSMs. Throughout this paper, we develop an implementation of the PMH algorithm in the statistical programming language R. We provide the reader with some intuition

for how the algorithm operates and provide some solutions to numerical problems that might occur in practice. Furthermore, we make use of the implementation for parameter inference in models using real-world data and provide a small survey of the field.

Background and contribution: The idea for the paper originated from Thomas Schön during the spring of 2015. The main aim was to provide an overview of the PMH algorithm together with step-by-step instructions on how to implement it in some common programming languages. The paper was written during the autumn of 2015 and example code for MATLAB, R and Python are provided via GitHub. The author of this thesis wrote most of the paper, made all implementations in software and carried out all numerical experiments.

Paper B

Paper B of this thesis is an edited version of

> J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015b.

which is a development of the two earlier contributions:

> 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.
>
> 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.

Source code and data: `https://github.com/compops/pmh-stco2015`

Summary: We develop an extension to the standard PMH algorithm, which incorporates information about the gradient and the Hessian of the log-posterior into the parameter proposal. This information can be extracted from the output generated by the particle filter when estimating the likelihood. The gradient information is used to add a drift in the proposal towards areas of high posterior probability. The Hessian information is useful to scale the step lengths of the proposal to improve the exploration of non-isotropic posteriors. We provide numerical experiments that indicates that the novel proposal makes the algorithm scale invariant, increases mixing and decreases the number of pilot runs.

Background and contribution: The idea for the first paper originated from Fredrik Lindsten during the autumn of 2012. The paper was developed in three stages and resulted in a journal publication after an invitation to a special issue connected to new results presented at the workshop MCMSki 2014. The first paper only made use of gradient information and was a proof of concept. Hessian information was added in the second paper and another particle smoother was used to decrease the computational cost. In the final paper, we introduced an improved method for estimating the gradient and Hessian together with an approach to handle cases when the estimate of the Hessian is not a valid covariance matrix. The author of this thesis wrote most parts of the conference papers, about half of the journal paper, made all implementations in software and carried out all numerical experiments. A similar idea was developed independently by Nemeth et al. (2014) during the same period.

Paper C

Paper C of this thesis is an edited version of

> J. Dahlin, F. Lindsten, and T. B. Schön. Quasi-Newton particle Metropolis-Hastings. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 981–986, Beijing, China, October 2015c.

Source code and data: `https://github.com/compops/qpmh2-sysid2015`

Summary: We develop the ideas from Paper B further by constructing an estimate of the Hessian directly from gradient information. This is useful as it often is difficult to obtain positive semi-definite estimates of the Hessian using particle smoothing. This problem can be mitigated by increasing the number of particles but this results in that the computational cost also increases. Instead, we propose to construct a local approximation of the Hessian using ideas from quasi-Newton optimisations, which often results in a positive semi-definite estimate. The novel approach only requires estimates of the gradient, which usually are more accurate compared with the estimates of the Hessian. We make use of the algorithm for inference in a challenging class of models known as SSMs with intractable likelihoods. The results indicate that the proposed algorithm can in some cases increase the mixing by a factor of four, when the gradients can be accurately estimated.

Background and contribution: The idea for the paper originated from the author of this thesis during the spring of 2014, when preparing an example in the presentation for the defence of his Licentiate's thesis. The proposed algorithm is an attempt to increase the mixing of PMH when the likelihood is estimated using particle filtering with approximate Bayesian computations (ABC). It was later used to compare with the approximate method proposed in Paper E. The author of this thesis wrote most of the paper, made all implementations in software and carried out all numerical experiments.

Paper D

Paper D of this thesis is an edited version of

> J. Dahlin, F. Lindsten, J. Kronander, and T. B. Schön. Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables. *Pre-print*, 2015a. arXiv:1512.05483v1.

Source code and data: `https://github.com/compops/pmmh-correlated2015`

Summary: The standard formulation of the PMMH algorithm makes use of independent estimators for the value of the target distribution. However, in theory we can increase the acceptance rate of the algorithm by introducing a positive correlation between two consecutive target estimates. We explore this idea by introducing a Crank-Nicolson proposal for the random variables which are used to construct the estimator of the target. We provide some numerical experiments indicating that this small change in the PMMH algorithm can increase mixing and allow for a decrease in the number of particles. The typical increase in mixing results in that the number of iterations can be decreased to a third compared with using non-correlated random variables. Furthermore, we can often also decrease the

number of random variables in the estimator, which results in a further decrease of the computational cost.

Background and contribution: The original idea for the paper originated from discussions between the author of this thesis and Joel Kronander during the summer of 2015. The idea was then extended and refined during discussions with Fredrik Lindsten during the autumn of 2015. The author of this thesis wrote about half of the paper, made all implementations in software and carried out all numerical experiments. A similar idea was developed independently by Deligiannidis et al. (2015) published on the pre-print library arXiv one day before our own paper.

Paper E

Paper E of this thesis is an edited version of

> J. Dahlin, M. Villani, and T. B. Schön. Efficient approximate Bayesian inference for models with intractable likelihoods. *Pre-print*, 2015d. arXiv:1506.06975v1.

which is the development of the two earlier contributions:

> 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.
>
> 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 2014b.

Source code and data: `https://github.com/compops/gpo-abc2015`

Summary: We propose a method for approximate Bayesian inference in SSMs with intractable likelihoods. The posterior in this type of models can be approximated point-wise using ABC. However, the resulting sequential Monte Carlo algorithm with ABC (SMC-ABC) for approximating the likelihood in SSMs often requires more particles than the standard SMC implementation to achieve reasonable accuracy. To decrease the resulting large computational cost, we propose a combination of SMC-ABC and Gaussian process optimisation (GPO) to estimate the parameters by maximising a surrogate function mimicking the posterior distribution. We provide numerical experiments indicating that the constructed surrogate function is similar to the true posterior around the mode and results in similar parameter estimates. Furthermore, the use of GPO can decrease the computational cost with one or two orders of magnitude compared with the PMH algorithm.

Background and contribution: The original idea was proposed by Fredrik Lindsten during the summer of 2013. In the first paper, we combined GPO with a standard particle filter for maximum likelihood estimation in SSMs. During the fall of 2013, the author of this thesis attended a course in Bayesian inference given by Mattias Villani. The idea of making use of GPO in combination with ABC was born during this course and resulted in a technical report as part of the course project. This report was reworked and extended twice to its

current form during the spring of 2015. The author of this thesis wrote most parts of the papers, made all implementations in software and carried out all numerical experiments. A similar idea was developed independently by Gutmann and Corander (2015) during the same period.

Paper F

Paper F 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.

Source code and data: `https://github.com/compops/rjmcmc-sysid2012`

Summary: 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 (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.

Background and contribution: The original idea was proposed by Fredrik Lindsten during the autumn of 2011. It was the first project undertaken by the author of this thesis during his PhD studies. The author of this thesis wrote the latter half of the paper, made some implementations in software and carried out most of the numerical experiments. The EEG data was kindly provided by Eline Borch Petersen and Thomas Lunner at Eriksholm Research Centre, Oticon A/S, Denmark.

Paper G

Paper G of this thesis is an edited version of

> J. Dahlin, R. Kohn, and T. B. Schön. Bayesian inference for mixed effects models with heterogeneity. Technical Report LiTH-ISY-R-3091, Department of Electrical Engineering, Linköping University, Linköping, Sweden, March 2016.

Source code and data: `https://github.com/compops/panel-dpm2016`

Summary: Mixture modelling is an important problem in many scientific fields. In this paper, we are interested in modelling panel data, i.e., a few sequential observations gathered from many individuals. This type of data sets provides little information about a specific individual and the main challenge is to pool information from similar individuals to obtain accurate estimates of the parameters of the model. We compare two different approaches

to pool the individuals together using a Dirichlet process mixture (DPM) and a finite mixture model with a sparseness prior. In this setting, we can see the latter approach as an approximation of the DPM, which results in simpler and sometimes more efficient inference algorithms. We conclude via numerical experiments that the posteriors obtained from the two approaches are very similar. Therefore, the approximate model can be beneficial for inference in big data problems.

Background and contribution: The idea of the paper originated from discussions between the author of this thesis and Robert Kohn during the autumn of 2014. Some preliminary work was carried out during author's PreDoc at University of New South Wales Business School in Sydney, Australia. The present paper is the result of work during the spring of 2016. The author of this thesis wrote most of the paper, made all implementations in software and carried out all numerical experiments.

Paper H

Paper H of this thesis is an edited version of,

> P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. On robust input design for nonlinear dynamical models. *Automatica*, 2016a. (provisionally accepted).

which is a development of the earlier contribution

> 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.

Summary: Input design is an important sub-field of system identification. Its main aim is to to determine an input that maximises a scalar function of the Fisher information matrix. In this work, we make use of graph theory to create a model for the input signal based on a convex combination of different basis inputs. The resulting input signal is given as a solution to an optimisation problem, which depends on estimates of the Fisher information matrix for each basis input. We develop a particle smoothing technique to obtain these estimates in a more efficient and accurate manner than previously. Finally, we present numerical illustrations indicating that the use of the designed input decreases the uncertainty in the estimates and improves the convergence speed of the expectation maximisation algorithm.

Background and contribution: The idea of the first paper originated from discussions between the author of these papers during the spring of 2013. The main aim was to combine recent developments in particle smoothing with input design. This idea was implemented in the first paper as a proof of concept. It was later extended in a second paper with a robust formulation and a better estimator of the Fisher information matrix. The author of this thesis wrote parts of the sections regarding particle methods in two the papers, made all particle-based implementations in software and carried out most of experiments.

Publications

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

P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. Particle-based Gaussian process optimization for input design in nonlinear dynamical models. *Pre-print*, 2016b. arXiv:1603.05445v1.

⋆ J. Dahlin, R. Kohn, and T. B. Schön. Bayesian inference for mixed effects models with heterogeneity. Technical Report LiTH-ISY-R-3091, Department of Electrical Engineering, Linköping University, Linköping, Sweden, March 2016.

⋆ P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. On robust input design for nonlinear dynamical models. *Automatica*, 2016a. (provisionally accepted).

⋆ J. Dahlin and T. B. Schön. Getting started with particle Metropolis-Hastings for inference in nonlinear models. *Pre-print*, 2015. arXiv:1511.01707v4.

⋆ J. Dahlin, F. Lindsten, J. Kronander, and T. B. Schön. Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables. *Pre-print*, 2015a. arXiv:1512.05483v1.

A. Svensson, J. Dahlin, and T. B. Schön. Marginalizing Gaussian process hyperparameters using sequential Monte Carlo. In *Proceedings of the 6th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Cancun, Mexico, December 2015.

⋆ J. Dahlin, F. Lindsten, and T. B. Schön. Quasi-Newton particle Metropolis-Hastings. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 981–986, Beijing, China, October 2015c.

M. Kok, J. Dahlin, , T. B. Schön, and A. Wills. Newton-based maximum likelihood estimation in nonlinear state space models. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 398–403, Beijing, China, October 2015.

T. B. Schön, F. Lindsten, J. Dahlin, J. Wågberg, C. A. Naesseth, A. Svensson, and L. Dai. Sequential Monte Carlo methods for system identification. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 775–786, Beijing, China, October 2015.

⋆ J. Dahlin, M. Villani, and T. B. Schön. Efficient approximate Bayesian inference for models with intractable likelihoods. *Pre-print*, 2015d. arXiv:1506.06975v1.

⋆ J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015b

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

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.

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.

J. Dahlin. *Sequential Monte Carlo for inference in nonlinear state space models*. Licentiate's thesis no. 1652, Linköping University, May 2014.

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 publications 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.

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.

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, USA, June 2014.

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

Bayesian modelling and inference

Bayesian modelling and inference is a popular and growing tool in statistics, machine learning and data mining. It is one of the two dominating perspectives used in probabilistic modelling and has certain interesting features for handling over-fitting, prior information and uncertainty, which can be useful in applications. Bayesian statistics has its origins with the English reverend Thomas Bayes [1701-1761]. He discussed the first known use of Bayesian inference in Bayes (1764) for the Bernoulli model with what is now known as a uniform prior. However, the general formulation and many important theorems are due to the French mathematician Pierre-Simon Laplace [1749-1827]. He proposed the well-known theorem named after Bayes in Laplace (1886). As a consequence, Bayesian inference is also known as Laplacian statistics or inverse probability.

However, the popularity of Bayesian inference faded during the early 20th century when the English statistician Ronald Fisher [1890-1962] proposed the Frequentisitic paradigm for statistical inference. This view is based on the optimisation of the likelihood function, which was first proposed in Fisher (1922). The resulting method is known as maximum likelihood and can be carried out in closed-form for many interesting applications. This in contrast with Bayesian inference, which often is analytically intractable and requires approximations to compute estimates. This is perhaps the reason that Bayesian statistics took the back seat in statistics for some time.

This changed with the advent of the electronical computer in the 1940s and onwards. Computational methods known as statistical simulation started to be applied to approximate the estimates obtained by Bayesian inference. Monte Carlo methods emerged as a good alternative for solving integration problems in high dimensions. These methods eventually found their use in Bayesian inference during the 1980s as most problems in this paradigm are posed as problems of computing integrals.

Figure 2.1. The basic inference process.

This chapter provides an overview of Bayesian modelling and inference with the aim to introduce the statistical inference process following the steps presented in Figure 2.1. The first step in any inference procedure is to collect data from the system of interest, e.g., a machine, the weather, the stock market or a human body. To describe the data, we require a statistical model which usually depends on a number of unknown parameters. In the last step, we make use of the data to infer the parameters of interest. After the model has been inferred, we can make use of it to make forecasts or for making decisions by taking the uncertainty into account.

Furthermore, we introduce two examples of applications in this chapter and these are analysed throughout the introductory part of this thesis. We also give an overview of non-parametric methods, where the number of parameters is infinite or grows with the number of observations. This property makes this type of models more flexible compared with the aforementioned parametric type of models. We conclude this chapter by providing the reader with an outlook and references for further study.

Three examples of statistical data

The first step in constructing a probabilistic model of a phenomenon is to collect data from the system, individual or some other source. In this section, we discuss three different types of data: (i) cross-sectional, (ii) time series and (iii) panel/longitudinal. We also presents two data sets that are later used to exemplify modelling and inference using different approaches.

Cross-sectional data

In cross-sectional data, we obtain a collection of observations $y = \{y_i\}_{i=1}^n$ from n different sources/individuals. Furthermore, we often assume that these observations are independent from each other and that they are recorded at the same time or that the observations are independent of time. Three examples of cross-sectional data are: (i) the length of students in a class room, (ii) the monthly wages at a company and (iii) the chemical factors that influence the quality of wine.

These observations are typically recorded together with additional information which is assumed to be able to explain the outcome. In the wage example, we would like to take into account the age, the educational background and the number of years that each person has worked for the company. We usually refer to the observation y_i as the *dependent variable* and the additional information as the independent or explaining variables. The *independent variables* are denoted by $x_i = \{x_{ij}\}_{j=1}^p$, where p is the number of different attributes recorded for each observation y_i. A typical question that the statistician would like to

answer is which independent variables influence the observation and by how much. We return to modelling this type of data using a regression model in Section 2.2.1.

Time series data

In time series data, we obtain multiple sequential observations $\{y_t\}_{t=1}^T$ from a single source or individual. We typically assume that the observations are dependent and that the correlation increases when the observations are closer (in time) to each other. The main objective is therefore to capture this correlation using a statistical model. Three typical examples of time series data are: (i) the ice varve thickness from Section 1.1.1, (ii) the price of coffee beans in Papers C and F and (iii) the blood sugar level of a patient. Another type of time series data is presented in Example 2.1.

We often assume that the current observation can be explained by previous observations. However, we can also add independent variables as in the cross-sectional case. For the blood sugar example, it can be useful to also take into account the amount of physical exercise, what the patient has eaten and if he/she is a diabetic when trying to forecast the future blood sugar level. We refer to these variables as the *exogenous* variables and denote them by $u_t = \{u_{tj}\}_{j=1}^p$. A typical problem that the statistician would like to solve is to determine the value of y_{t+m} given $\{y_t, u_t\}_{t=1}^T$ for $m > 0$. That is, to make a so-called m-step predication of the observation given all the previous observations and independent variables available at the present. We return to model this type of data using two different time series models in Sections 2.2.2 and 2.2.3.

Example 2.1: How does unemployment affect inflation?

Consider the scenario that the Swedish parliament has launched a big campaign against unemployment. The unemployment rate is expected to decrease rapidly during the coming 24 months. At the same time the Swedish Riksbank (central bank) is worried that this might increase the inflation rate above its two percent target. They would like us to analyse this scenario by providing them with a forecast to determine if any action is required.

The reasoning of the Riksbank is based on the Phillips curve hypothesis proposed by Phillips (1958). It states that there is an inverse relationship between unemployment and inflation rates in the short run. That is, a rapid decrease in unemployment tends to correlate with a increased rate of inflation. The intuition for this is that it is difficult for companies to attract workers if the unemployment rate is too low. As a consequence, the employees gain bargaining power which results in increased wages and therefore increased inflation as well. The opposite occurs when the unemployment rate is high as it is easy for companies to recruit new workers. Therefore, no wage increases are required to attract new workers.

Furthermore, the Phillips curve assumes that there exists an equilibrium point in the unemployment known as the *non-accelerating inflation rate unemployment* (NAIRU) or the natural rate of unemployment. The reason for a non-zero NAIRU is the *matching problem* on the labour market. That is, not all individuals can take any available position due to e.g., geographical or educational constraints. The NAIRU determines if the inflation rate increases or decreases given the current unemployment rate. The inflation increases if the

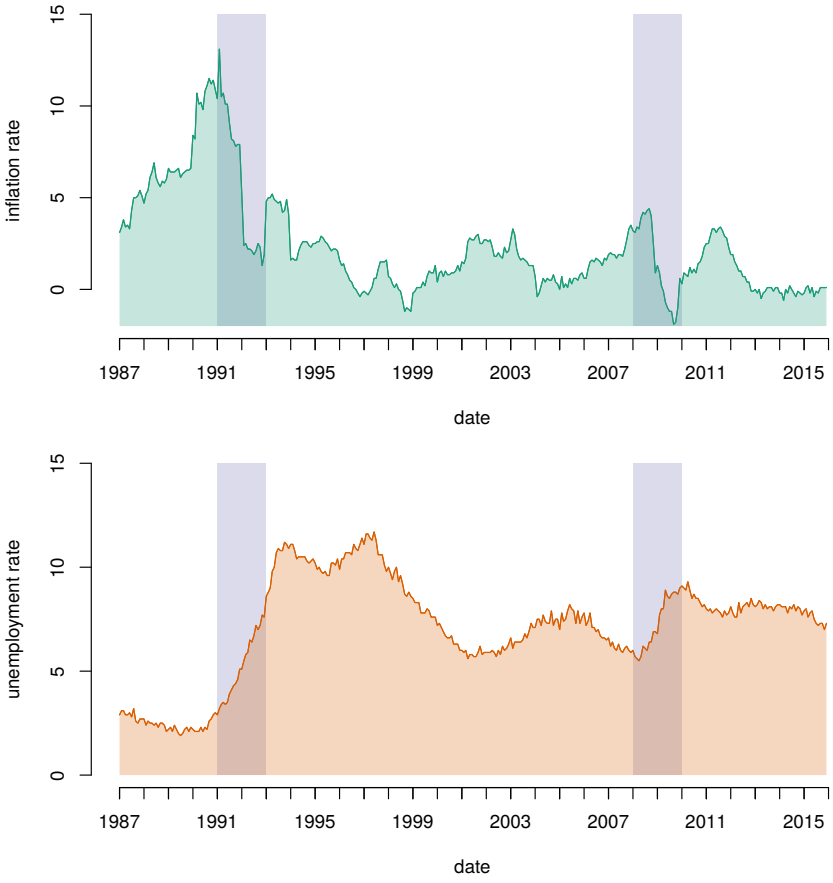


Figure 2.2. The inflation rate (upper) and unemployment rate (lower) for Sweden during the period January, 1987 to December, 2015. The purple areas indicate the financial crises of 1991-1992 and 2008-2010. The data is obtained from Statistiska centralbyrån (SCB).

unemployment rate is smaller than the NAIRU and vice versa. Estimating the NAIRU is therefore important for making predictions of the future inflation.

In Figure 2.2, we present the unemployment and inflation rates in Sweden between January, 1987 and December, 2015. The data is obtained from Statistiska centralbyrån[1] (SCB), which is the governmental agency responsible for collecting statistics in Sweden. We note that the inflation rate changed rapidly during the financial crises in 1991-1992 and 2008-2010 and at the same time the unemployment rate increased. This suggests that a negative correlation between these two variables could exist. We would like to determine the support for this claim using a probabilistic model, which also is required to make the forecast asked for by the Riksbank.

We return to this data set in Example 2.3 on page 27.

Panel and longitudinal data

Panel data (also known as longitudinal data) can be seen as a combination of time series and cross-sectional data. In this setting, we typically obtain a data set $y = \{\{y_{it}\}_{i=1}^{n}\}_{t=1}^{T}$ for individual i at time t with $T \ll n$. We assume that the observations are independent between individuals but correlated between observations of the same individual. For each observation y_{it} of individual i at time t, we usually record p independent variables denoted by $\{x_{ijt}\}$ for $j = 1, \ldots, p$.

One example of panel data is socio-economic studies such as the *Sozio-oekonomisches Panel* (SOEF[2]), where a selection of German households have been interviewed annually since 1984. Topics included in the annual interviews are economical factors such as employment and earnings as well as social factors such as family composition, health and general life satisfaction. Analysis of such data is important to e.g., investigate how household incomes correlate with university attendance. This can be useful to guide interventions and policy decisions considered by the government.

Two other applications of panel data were already discussed in Chapter 1: (i) genome-wide association studies and (ii) recommendation systems. In application (i), scientists are making use of rapid scanners to search for markers connected with a particular disease in a DNA sample, see Bush and Moore (2012), Zhang et al. (2010) and Yang et al. (2014). This information is useful for diagnoses, treatment and prevention of e.g., cancer, diabetes and heart disease. In application (ii), the dynamics of users' preferences can be seen as panel data, see Condliff et al. (1999) and Stern et al. (2009). We return to discussing how to model panel data in Section 2.2.4.

Example 2.2: Voting behaviour in the US Supreme court

In February 2016, the conservative justice Antonin Scalia of the US Supreme Court justice died and a replacement is therefore required. The US President Barack Obama is faced with a dilemma to either appoint a new justice with the same ideological leaning or one who is more liberal. The US Supreme Court is made up by nine different justices, which are

[1] See `http://www.scb.se/en_/` for more information.
[2] See `http://www.diw.de/en/soep` for more information.

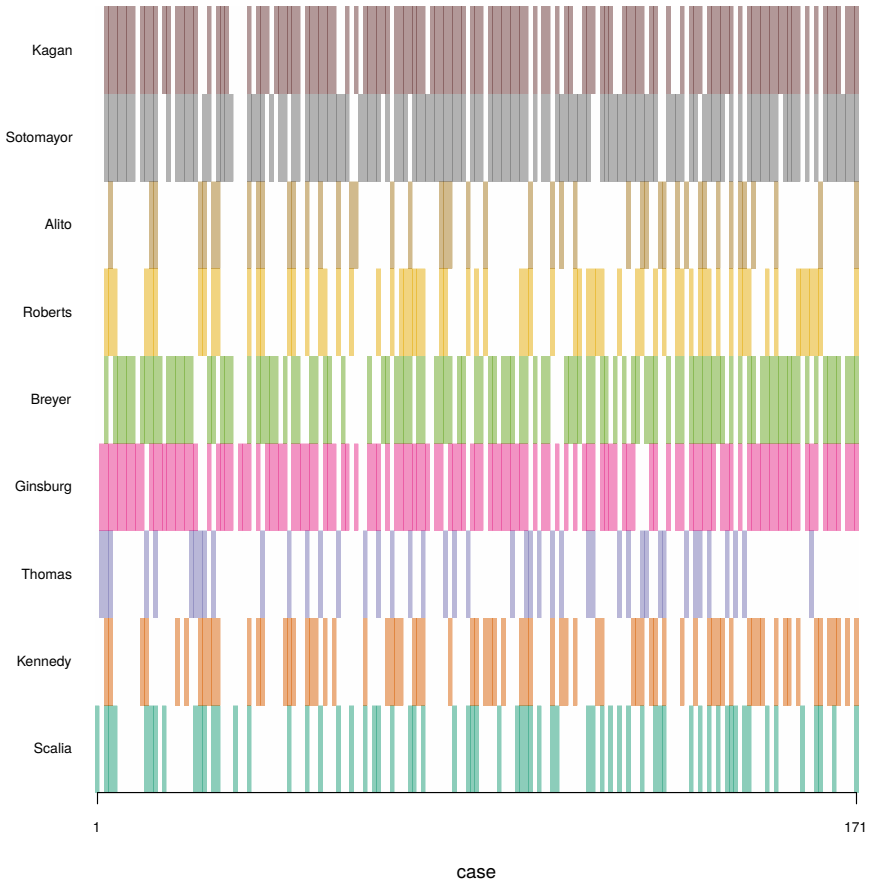


Figure 2.3. The rulings in $T = 171$ non-unanimous cases in the US Supreme Court during the terms between 2010 and 2015. Liberal votes are indicated by coloured fields and conservative by white for each justice and case.

nominated by the President and approved by the Senate. The nomination is an important political decision as each justice often serves for the remainder of his/her life or until he/she resigns. The political view of each justice can therefore influence rulings during many years. How will appointing a more liberal judge affect the rulings of the court?

We consider the data set provided by Spaeth et al. (2016) of $T = 171$ non-unanimous rules from the terms between 2010 and 2015. At the time, the supreme court justices were: Kagan, Sotomayor, Alito, Roberts, Breyer, Ginsburg, Thomas, Kennedy and Scalia. The vote of each justice is categorised as either liberal (1) or conservative (0) depending on the topic at hand. The data set is presented in Figure 2.3 for each justice and case, respectively.

We would like to model the relative level of conservatism/liberalism between the justices. A quick look at the data seems to indicate that (i) Kagan, Sotomayor, Breyer and Ginsberg seem to be more liberal and (ii) Alito, Thomas and Scalia seem to be more conservative. However, we would like to verify this using a probabilistic model. This model can later be used to simulate votes by each justice to estimate the ideological leaning of the court.

We return to this data set in Example 2.4 on page 28.

Parametric models

The next step in probabilistic modelling after the data has been collected is to choose a suitable model structure. In this section, we present a few different structures aimed at modelling the three different kinds of statistical data discussed in the previous section. All of the models presented are members of the family of *parametric models*. Hence, we assume that they are specified by a finite number of parameters denoted by $\theta \in \Theta \subset \mathbb{R}^p$, where Θ denotes the parameter space which we typically assume to be the p-dimensional real space.

The choice of model structure is often difficult and greatly influences the predictions and decisions made from the combination of the data and the model. Hence, model choice is an important problem in statistics but it is not discussed at any length in this thesis. Two approaches are likelihood ratio tests and Bayes factors, see Casella and Berger (2001) and Robert (2007) for more information.

Linear regression and generalised linear models

Generalised linear models (GLMs) are the work-horse of statistical modelling of cross-sectional data. This is a type of regression model, where we would like to infer the observation (dependent) variable given the independent variables. The latter are typically referred to as the regressors in this model. The basic GLM is given by the *linear regression* model, which can be expressed by

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \sigma e_i, \tag{2.1}$$

where it is typically assumed that errors are independent and distributed according to the standard Gaussian distribution, i.e., $e_i \sim \mathcal{N}(0, 1)$. Note that this implies that the noise

variance is constant for all the observations, i.e., the errors are *homoscedastic*. Furthermore, we assume that the regressors are linearly independent of each other and with the error, i.e., $\mathbb{E}[x_{ij}e_i] = 0$ for every i and j.

In this model, the parameters are given by $\theta = \{\beta_{0:p}, \sigma\}$, where $\beta_0 \in \mathbb{R}$ determines the so-called intercept (or bias) of the model. The standard deviation of the noise is determined by $\sigma > 0$. The remaining parameters $\beta_{1:p} \in \mathbb{R}^p$ determines the linear relationship between the regressors and the observation. A standard method for estimating θ from data is to make use of the least squares (LS) approach. The main objective in LS is to minimise the squared prediction error, i.e.,

$$\widehat{\beta}_{\text{LS}} \triangleq \operatorname*{argmin}_{\beta \in \Theta} \|y - X\beta\|_2^2, \tag{2.2}$$

where $\|\cdot\|_2$ denotes the L_2-norm. Here, we introduce

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \qquad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}, \qquad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}.$$

The solution to (2.2) can be computed using a closed-form expression (Casella and Berger, 2001) given by

$$\widehat{\beta}_{\text{LS}} = \left(X^\top X\right)^{-1}\left(X^\top y\right), \tag{2.3}$$

which we refer to as the *normal equations*. The noise variance σ^2 can be estimated directly using the standard sample estimator by

$$s^2 = \widehat{\sigma}^2 = \frac{1}{n-p-1}\sum_{i=1}^{n}\left(y_i - X_i\widehat{\beta}_{\text{LS}}\right)^2, \tag{2.4}$$

where X_i denotes row i of the matrix X. It is possible to show that the LS estimator is the best linear un-biased estimator (BLUE) when the assumptions of the error term and the independence between the error and the regressors are fulfilled. This result is known as the Gauss-Markov theorem (Casella and Berger, 2001; Lehmann and Casella, 1998) and it holds for other types of linear models as well.

An alternative to the least squares formulation (2.2) is the *elastic net* (Zou and Hastie, 2005; Hastie et al., 2009). The resulting loss function is given by

$$\widehat{\beta}_{\text{elastic}} \triangleq \operatorname*{argmin}_{\beta \in \Theta} \|y - X\beta\|_2^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2, \tag{2.5}$$

where $\|\cdot\|_1$ denotes the L_1-norm and $\lambda_1, \lambda_2 > 0$ denote tuning parameters. This type of loss function is sometimes referred to as *regularised least squares* (RLS) and is particularly useful for the case $p > n$, i.e., when we have more parameters than observations. We can recover two important special cases from (2.5): (i) L1-RLS when $\lambda_2 = 0$ and (ii) L2-RLS when $\lambda_1 = 0$.

The advantage of this alternative formulation is that it penalises the inclusion of regressors

that does not contribute to explain the observations. That is, these regression coefficients are shrunk towards zero and therefore the regressors are practically removed from the model. This is a form of model selection, which as previously mentioned is a challenging issue.

Another popular name for (i) is the LASSO (Tibshirani, 1996) and it has the property to shrink regression coefficients to be exactly zero. The drawback with the LASSO is bad performance when the regressions exhibit *multicollinearity*, i.e., linear dependence between some regressors. In this case, a better alternative is (ii) known also as *ridge regression* introduced in statistics by Hoerl and Kennard (1970). The drawback with this regularisation is that is only shrinks the coefficients towards zero and do not remove them completely. We return to the use of regularisation for automatic model order selection in Chapter 4.

In the linear regression model, we assume that the response y_i is a continuous random variable. However, many other types of responses can be found in applications. Some examples are: (a) binary response (success/failure), (b) Bernoulli response (no. successes in M attempts) or (c) count response (no. occurrences during some time period). The GLM is a useful approach to model these kinds of observations. This is done by transforming the linear predictor $\eta_i = X_i\beta$ with a so-called link function h such that $\mathbb{E}[y_i] = h^{-1}(\eta_i)$. For example in (a), we can use the logistic function or the Gaussian cumulative distribution function (CDF) to map the linear predictor onto the unit interval $(0, 1)$.

Autoregressive models

There are a large number of different models for time series data. The simplest is probably the *autoregressive process* of order p denoted by AR(p), see e.g., Brockwell and Davis (2002). This model can be expressed using densities by

$$y_t \,|\, y_{t-p:t-1} \sim \mathcal{N}\left(y_t; \mu + \sum_{k=1}^{p} \phi_k (y_{t-k} - \mu), \sigma^2\right), \tag{2.6}$$

where y_t denotes the observation at time t. The model is specified by the parameters $\theta = \{\mu, \phi_{1:p}, \sigma\}$ and the noise is assumed to be independent and Gaussian. The latter can be relaxed to account for outliers by assuming Student's t distributed noise, which is considered in Paper F. Note that we make use of densities to define the AR process, which is slightly different from the equation form of the LS model. However, it is possible to rewrite the AR process on the difference form corresponding to the LS model (2.1) and vice versa.

In the AR model, the model order $p \in \mathbb{N}$ influences the number of past observations included into the model. Therefore, p together with ϕ determines the persistence and correlation structure of the process. The mean of the observations is determined by $\mu \in \mathbb{R}$ and the standard deviation of the noise is determined by $\sigma > 0$. We require that all the poles of the characteristic polynomial

$$q^p - \sum_{k=1}^{p} \phi_k q^{p-k} = 0,$$

lie within the unit circle to obtain a stable AR(p) process, i.e., it does not diverge to infinity when T increases. Here, q denotes the back shift (lag) operator.

Given the order p, we can reformulate the problem of estimating θ in (2.6) when $\mu = 0$ as a LS problem using the observations $y_{1:T}$. The estimates are obtained by rewriting the model to obtain

$$y = \begin{bmatrix} y_{p+1} \\ y_{p+2} \\ \vdots \\ y_T \end{bmatrix}, \qquad X = \begin{bmatrix} y_p & y_{p-1} & \cdots & y_1 \\ y_{p+1} & y_p & \cdots & y_2 \\ \vdots & \vdots & \ddots & \vdots \\ y_T & y_{T-1} & \cdots & y_{T-p} \end{bmatrix}, \qquad \phi = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix}.$$

In this case, the LS estimate corresponds to the maximum likelihood estimate. However, it is also possible to estimate the parameters in a Bayesian setting. Furthermore, we can apply regularisation for selecting the model order using a loss function similar to (2.5). We investigate this in Paper F for AR with exogenous inputs (ARX) models, where the known input $\{u_t\}_{t=1}^T$ (possibly lagged) is included in (2.6).

State space models

In the AR model, we obtained direct observations of the quantity of interest $y_{1:T}$. In some cases, we cannot directly observe the cause of the observation as it is a function or random variable depending on some latent variables. A standard model for time series data using latent variables is the SSM, which is also known as the hidden Markov model (HMM). This type of model is used e.g., in statistics (Brockwell and Davis, 2002; Langrock, 2011), control (Ljung, 1999), econometrics (Durbin and Koopman, 2012) and finance (Tsay, 2005).

An SSM with latent states $x_{0:T}$ and observations $y_{1:T}$ can be expressed as

$$x_0 \sim \mu_\theta(x_0), \quad x_t \,|\, x_{t-1} \sim f_\theta(x_t \,|\, x_{t-1}), \quad y_t \,|\, x_t \sim g_\theta(y_t \,|\, x_t), \tag{2.7}$$

where θ denotes the parameters of the model. Here, we assume that the model can be described by probability density functions (pdfs) denoted μ_θ, f_θ and g_θ.

We say that the SSM is fully dominated (by the Lebesgue measure) when we can write the model on the form in (2.7). That is, when we can find a density for each of the Markov kernels in the model. In practice, this can often be done when the states and observations are real-valued and the state or observation equations are not deterministic. However, the methods presented in this thesis can be applied even when the densities are degenerate (states are deterministic) and when the states/observations are integers. The reason for adopting the density formulation is to keep the notation simple and avoid the measure-theoretic formulation of stochastic processes.

The parameters of interest in the SSM are the latent states $x_{0:T}$ and the parameters of the densities θ. We refer to the problem of estimating the former as the *state inference problem* and the latter as the *parameter inference problem*. For this type of model, we cannot form a simple optimisation problem as for the LS or AR models as the states are not directly observed. Instead, we require more advanced maximum likelihood or Bayesian inference methods to solve these two inference problems jointly. We return to this in Section 2.3.

Example 2.3: How does unemployment affect inflation? (cont. from p. 19)

We consider the Phillips curve with non-linear dynamics and rational expectations introduced by Zhou (2013) to model the Swedish data and to make forecasts. Let y_t and u_t denote the inflation rate and unemployment rate (in percent) at time t. Furthermore, let x_t denote the NAIRU (the equilibrium point in the unemployment rate), which changes with time depending on previous rates of inflation and unemployment. We can write a slightly modified version of this model as an SSM given by

$$x_0 \sim \mathcal{N}\big(x_0; 2, 4\big), \tag{2.8a}$$

$$x_t \,|\, x_{t-1} \sim \mathcal{N}\Big(x_t; \phi x_{t-1} + \mu(x_{t-1}), \sigma_v^2(x_{t-1}, u_{t-1})\Big), \tag{2.8b}$$

$$y_t \,|\, x_t \sim \mathcal{N}\big(y_t; y_{t-1} + \beta(u_t - x_t), \sigma_e^2\big). \tag{2.8c}$$

We introduce the mean function of the state process and its variance function given by

$$\mu(x_{t-1}) \triangleq \alpha\Big[1 + \exp\big(-x_{t-1}\big)\Big]^{-1}, \quad \sigma_v^{-1}(x_{t-1}, u_{t-1}) \triangleq 1 + \exp\Big[-\big|u_{t-1} - x_{t-1}\big|\Big].$$

The parameters of this Phillips curve model are $\theta = \{\alpha, \beta, \phi, \sigma_e\}$. Here, $\alpha \in \mathbb{R}$ and $\phi \in (-1, 1)$ determine the mean and persistence of the state, respectively. The inflation rate is determined by $\beta \in \mathbb{R}$ and $\sigma_e > 0$. The parameter β is of great interest to us as its sign determines the correlation between the inflation and the unemployment gap (the difference between the unemployment rate and the state). The Phillips curve hypothesis suggests that this parameter is negative.

We continue by analysing the mean and the variance of the state process to obtain some insight into the model and its dynamics. Note that the term $\mu(x_{t-1})$ makes the state process *mean-reverting*. That is, the average value of the process is given by $\mu(x_{t-1})$ and therefore the process occasionally reverts to this value. Furthermore, $\mu(x_{t-1})$ can vary between

$$\mu \to \alpha, \text{ when } x_{t-1} \gg 0, \quad \mu \to \frac{\alpha}{2}, \text{ when } x_{t-1} \approx 0.$$

That is, if x_t grows large, then the long-term mean of the process also grows, making it difficult for x_t to decrease again at a later stage. Hence, the state is sticky and if the unemployment rate is larger than the state of the process (the NAIRU) for a long time, then the mean of the latter grows. An explanation for this effect is that companies tend to streamline their organisations at the same time as employees are laid off, which can increase the matching problem.

For the noise standard deviation, we have

$$\sigma_v \to 0.5, \text{ when } |u_{t-1} - x_{t-1}| \to 0, \quad \sigma_v \to 1.0, \text{ when } |u_{t-1} - x_{t-1}| \to \infty,$$

so the process noise decreases as the unemployment rate u_{t-1} approach the NAIRU x_{t-1}. The reason for this is that the NAIRU (according to some economists) can be seen as the equilibrium state of the economy. Therefore, the inflation rate does not change if the unemployment rate is close to the latent state. However, it can increase when the unemployment rate is lower than the NAIRU, i.e., if $u_t - x_t < 0$. This imposes the condition that $\beta < 0$ for the negative correlation between inflation and unemployment to exist.

We return to this model in Example 3.3 on page 54.

Mixed effect models

In many panel data applications, we would like to separate the mean population behaviour and the individual deviations from this mean. This can be done using a *mixed effects model*, where the population behaviour is captured using so-called *fixed effects* and individual deviations are captured using *random effects*. A mixed effects model can be expressed by

$$y_{it} = \alpha_t x_{it} + \beta_i z_{it} + e_{it}, \tag{2.9}$$

where e_{it} denotes some error term, e.g., a Gaussian IID random variable or an AR(1) process. Here, $\alpha_t \in \mathbb{R}^d$ denotes the time-dependent fixed effects and $\beta_i \in \mathbb{R}^p$ denotes the individual-dependent random effects. The design matrices x_{it} and z_{it} for the two effects contain the intercept and the relevant regressors. For a general introduction to mixed effects models, see Fitzmaurice et al. (2008) and Greene (2008).

A common assumption for the individual random effects are

$$\beta_i \sim \mathcal{N}(\beta_i; \beta_0, \Sigma_{\beta_0}),$$

for some mean vector $\beta_0 \in \mathbb{R}^p$ and covariance matrix $\Sigma_{\beta_0} \in \mathbb{R}^{p \times p}$. However, this can be restrictive in some applications when the distribution of the random effects is multi-modal. An alternative approach that we consider in Paper G is therefore to replace this assumption a mixture of Gaussians. This generalisation allows for so-called *heterogeneity* in the individual random effects. The mixed effects model differs from the linear regression model as some of its parameters vary between individuals and some can vary over time. Inference in mixed effects models is therefore more complicated than for the linear regression model (2.1).

Finally, it is also possible to make use of a model similar to (2.9) when the observations are binary or integer. The resulting class of models is known as the generalised linear mixed model (GLMM), which is based on the same type of link functions as for the GLM. GLMMs are common in many different applications ranging from marketing and econometrics to health and medicine, see e.g., Baltagi (2008) and Fitzmaurice et al. (2008).

Example 2.4: Voting behaviour in the US Supreme court (cont. from p. 21)

Let the observation $y_{it} = 1$ denote a liberal vote and $y_{it} = 0$ denote a conservative vote of judge $i = 1, \ldots, n$ in case $t = 1, \ldots, T$. We model the votes using an *item response model* (IRM; Fox, 2010), which is similar to a GLMM with a probit link function, given by

$$u_{it} = \begin{bmatrix} \alpha_{t,1} & \alpha_{t,2} \end{bmatrix} \begin{bmatrix} -1 \\ x_i \end{bmatrix} + e_{it}, \qquad y_{it} = \begin{cases} 1, & u_{it} > 0, \\ 0, & u_{it} \leq 0, \end{cases}$$

where $\alpha_{t,1} \in \mathbb{R}$ and $\alpha_{t,2} \in \mathbb{R}$ denote the *difficulty* and *discrimination* parameter of case t respectively. Here, e_{it} denotes an independent standard Gaussian random variable. The quantity $x_i \in \mathbb{R}$ captures the relative liberal/conservative score for each justice.

Note that the score x_i and the so-called utility u_{it} are unknown *latent variables* in this model. The task is therefore to reconstruct these latent variables from the observations. However, this is difficult as we do not know the parameters of the model α_t.

We return to this model in Example 3.8 on page 63.

Computing the posterior and making decisions

We have discussed the first two steps of the inference process in some details. In this section, we introduce Bayesian inference for determining the unknown parameter θ from the data. The basic premise in Bayesian inference is that we treat the unknown parameter as a random variable. This parameter is assumed to be distributed according to some density denoted by $p(\theta)$. This object is known as the prior distribution of the parameter and encodes our *subjective beliefs* before looking at the data. From this prior distribution, we would like to compute the posterior distribution $p(\theta \mid y)$, where y denotes some data. This procedure is known as the *prior-posterior update* and is carried out using Bayes' theorem given by

$$p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{p(y)} \propto p_\theta(y)p(\theta). \tag{2.10}$$

Here, $p(y \mid \theta) \triangleq p_\theta(y)$ denotes the likelihood, which is a function of the data and summarises all the information about θ available in the observations. This is known as the *likelihood principle* in statistics. The posterior is therefore a combination of our prior beliefs about θ and the information about the parameter that is available in the observations. After computing the posterior, we can extract point estimates of θ and their uncertainties.

In maximum likelihood inference, it is assumed that the parameter θ is fixed and that the uncertainty comes from the data. The point estimate of θ is obtained by maximising the likelihood function $p_\theta(y)$. Moreover, it is possible to prove that this procedure gives the desired result in the limit of infinitely many observations (Lehmann and Casella, 1998), i.e., that the estimate equals the true parameter. In Bayesian statistics, we do not rely on asymptotic results as for the maximum likelihood estimator. However, it is known that the maximum likelihood estimator is the best un-biased estimator in many cases and can be difficult to beat in terms of statistical accuracy. However, this does not necessarily hold true when the number of observations is small.

Prior distributions

A major decision for the user of Bayesian statistics is the choice of $p(\theta)$. As previously mentioned, the prior distribution is often determined by expert knowledge about the current setting. A common type of prior is the *conjugate prior*, i.e., when the prior and posterior are given by the same type of distribution. This is a convenient choice for carrying out inference as the prior-posterior update amounts to recomputing some sufficient statistics. Conjugate priors can be found for many members of the exponential family of distribution when the data is IID, see Robert (2007) and Bishop (2006).

In many other cases, we cannot make use of conjugacy for selecting the prior. Instead, parametric distributions such as the Gaussian, Gamma and similar are used as prior distributions. This often results in that approximations are required to compute the posterior distribution. Two other popular alternatives for prior distributions are *improper priors* and *non-informative priors*. In former, we cannot normalise the prior and express it as a density. However, it is possible in some cases to obtain a valid posterior distribution, which integrates to one. A popular example is the uniform distribution over the positive real numbers, which forces the posterior to only have probability mass on this interval. This is useful to

encode stability properties in SSMs and other dynamical models. Non-informative priors try to introduce as small amount of prior information as possible. However, the construction of such priors is difficult and just applying flat priors in order to encode ignorance can have unforeseen effects on the posterior. For more information regarding non-informative priors, see Robert (2007) and Gelman (1996).

The choice of prior can greatly influence the shape and properties of the posterior distribution. Especially, when the amount of information in the data is small. It is therefore advisable to make use of a couple of different prior distributions and compare the resulting posterior. This approach is advocated by Spiegelhalter (2004). Another approach is *posterior predictive checks*, where data is simulated from the posterior and compared to the actual observations. This can be done *in-sample* (on the data used for estimation) or *out-of-sample* (on fresh data not used in the estimation). This is similar to *cross-validation*, which is useful for model order selection and model validation in e.g., machine learning and system identification. For more information about posterior predictive checks, see Gelman et al. (1996) and Gelman et al. (2013).

Finally, note that one of the strengths of Bayesian inference comes from the prior. For example, we can make use of prior knowledge to narrow the possible range for the parameter. This could be helpful in settings when the amount of data is limited or when we have identifiability issues. Furthermore, priors can be used to promote smoothness and sparsity in the parameter (vector), see Remark 2.5.

Remark 2.5 (Prior distribution for promoting smoothness and sparsity). Prior distributions are particularly useful in promoting: (i) smoothness and (ii) sparsity in the parameter posterior. Smoothness is an interesting property when modelling cross-sectional or time series data as the posterior estimates should vary slowly and smoothly between nearby data points. An example of this is the GP regression model introduced in Section 1.1.1 for modelling the thickness of ice varves. We return to GP regression models and their applications in Section 2.4.

RLS was introduced in the form of the elastic net (2.5) in Section 2.2.1 by adding two terms to the loss function. A more intuitive approach to RLS is to view it as Bayesian linear regression with specific choices of prior distributions. Two choices for promoting sparsity are given by

$$p(\beta_j) = \mathcal{L}(\beta_j; 0, \sigma), \qquad p(\beta_j) = \mathcal{N}(\beta_j; 0, \sigma^2),$$

for $j = 1, \ldots, p$. Here, $\mathcal{L}(\beta_j; 0, \sigma)$ and $\mathcal{N}(\beta_j; 0, \sigma^2)$ denote the zero-mean Laplace and Gaussian distributions with scale $\sigma > 0$, respectively. These two choices of priors corresponds to L1-RLS (LASSO) and to L2-RLS (ridge regression), respectively. As previously mentioned, the primary benefit of these priors is that they shrink regression coefficients towards zero and therefore automatically select the most important regressors. We return to the use of sparseness priors for model order selection in Section 4.2 and in Papers F and G.

The likelihood

As previously mentioned, the likelihood contains all the information in the observations about the parameter. The form of the likelihood is determined by the model of the data.

For example if the data is assumed IID, then the likelihood is given by

$$p_\theta(y_{1:n}) = \prod_{i=1}^{n} p_\theta(y_i),$$

where e.g., $p_\theta(y_i) = \mathcal{N}(y_i; \mu, \sigma^2)$ if the data has a Gaussian distribution. We can write similar expressions for the linear regression model and the AR(p) process.

Example 2.6: Voting behaviour in the US Supreme court (cont. from p. 28)

We have that the observations are IID Bernoulli from the model. Hence, we can express the likelihood by

$$p(y \mid \theta) = \prod_{i=1}^{n} \prod_{t=1}^{T} p_{it}^{y_{it}} (1 - p_{it}^{y_{it}})^{1-y_{it}},$$

with the *success probability* p_{it} is given by

$$p_{it} = \Phi\left(\begin{bmatrix} \alpha_{t,1} & \alpha_{t,2} \end{bmatrix} \begin{bmatrix} -1 \\ x_i \end{bmatrix} \right),$$

where $\Phi(\cdot)$ denotes the standard Gaussian cumulative distribution function (CDF). Using the likelihood, we can compute the posterior if we assume a prior distribution for each of the parameters $\{\alpha_{1:T}, u_{1:n,1:T}, x_{1:n}\}$. Here, we limit ourselves to computing the *conditional posterior* for the liberal/conservative score using the prior $x_i \sim \mathcal{N}(x_i; 0, 1)$. From Bayes' theorem (2.10), we obtain directly that

$$p(x_i \mid y, \alpha, u) \propto \mathcal{N}(x_i; 0, 1) \prod_{t=1}^{T} \mathcal{N}\big(u_{it}; -\alpha_{t,1} + \alpha_{t,2} x_i, 1\big),$$

which is the Gaussian prior for x_i multiplied with a Gaussian likelihood. Here, we have discarded all terms not depending on x_i. It is possible to rewrite the posterior as a Gaussian distribution with updated sufficient statistics. This is a result of the conjugacy between the likelihood and the prior, see Robert (2007) and Gelman et al. (2013). The calculation is done by completing the square in the exponent of the Gaussian density. The resulting conditional posterior can be written as

$$p(x_i \mid y, \alpha, u) = \mathcal{N}\left(x_i; \Sigma_{\text{post}}^{-1} \sum_{t=1}^{T} \alpha_{t,2}(u_{it} + \alpha_{t-1}), \Sigma_{\text{post}}^{-1} \right), \quad \Sigma_{\text{post}} = 1 + \sum_{t=1}^{T} \alpha_{t,2}^2.$$

It is possible to also compute conditional posteriors for u and α, see Albert (1992). This is done in the next part of this example. It turns out that we can sample from the posterior using Monte Carlo by iteratively sample from each of the three conditional posteriors given the remaining parameters.

We return to this model in Example 3.8 on page 63.

For the SSM, we can express the likelihood by using the decomposition

$$p_\theta(y_{1:T}) = p_\theta(y_1) \prod_{t=2}^{T} p_\theta(y_t \,|\, y_{1:t-1}), \tag{2.11}$$

where $p_\theta(y_t \,|\, y_{1:t-1})$ denotes the so-called *predictive likelihood*. We can express the predictive likelihood as the marginalisation given by

$$p_\theta(y_t \,|\, y_{1:t-1}) = \int_{\mathcal{X}^2} g_\theta(y_t \,|\, x_t) f_\theta(x_t \,|\, x_{t-1}) p_\theta(x_{t-1} \,|\, y_{1:t-1}) \,\mathrm{d}x_{t-1:t}, \tag{2.12}$$

which follows from the Markov property of the SSM. However, we cannot evaluate this integral in closed form for most SSMs as the latent states and thereby $p_\theta(x_{t-1} \,|\, y_{1:t-1})$ are unknown. This can be done in two special cases: (a) when the state space is finite (when the state only assumes a finite collection of values) and (b) when the SSM is linear and Gaussian as discussed in Remark 2.7. Otherwise, the computation of the likelihood is analytically intractable and approximations are required.

Remark 2.7 (Bayesian state inference in a linear Gaussian SSM). In this example, we make use of the properties of the Gaussian distribution to solve the state inference problem exactly for the linear Gaussian state space (LGSS) model. We can express a scalar version of a LGSS model by

$$x_t \,|\, x_{t-1} \sim \mathcal{N}\left(x_t; \mu + \phi(x_{t-1} - \mu), \sigma_v^2\right), \qquad y_t \,|\, x_t \sim \mathcal{N}\left(y_t; x_t, \sigma_e^2\right), \tag{2.13}$$

where the parameters are denoted by $\theta = \{\mu, \phi, \sigma_v, \sigma_e\}$. From (2.12), we know that the filtering distribution $\pi_t(x_t) \triangleq p_\theta(x_t \,|\, y_{1:t})$ is required to compute the likelihood. We can compute $\pi_t(x_t)$ using the *Bayesian filtering recursion* (Anderson and Moore, 2005) given by

$$\pi_t(x_t) = \frac{g_\theta(y_t \,|\, x_t)}{p_\theta(y_t \,|\, y_{1:t-1})} \int_{\mathcal{X}} f_\theta(x_t \,|\, x_{t-1}) \pi_{t-1}(x_{t-1}) \,\mathrm{d}x_{t-1}, \tag{2.14}$$

for $0 < t \leq T$. From the structure of the LGSS model, we assume that the prior distribution can be denoted by $\pi_{t-1}(x_{t-1}) = \mathcal{N}(x_{t-1}; \widehat{x}_{t-1|t-1}, P_{t-1|t-1})$. Here, $\widehat{x}_{t-1|t-1}$ and $P_{t-1|t-1}$ denote the filtered state and its covariance both at time $t-1$, respectively.

We can then solve (2.14) by using the properties of the Gaussian distribution. The solution is a recursion known as the Kalman filter (Kalman, 1960; Kailath et al., 2000). This is an iterative approach with two steps: (i) the *simulation step* computes the predicted state estimate and its covariance and (ii) the *correction step* computes the filtered state estimate and its covariance. The simulation step corresponds to simulating the system one time step forward according to the state process, which during iteration t consists of

$$\widehat{x}_{t|t-1} = \mu + \phi(\widehat{x}_{t-1|t-1} - \mu), \qquad P_{t|t-1} = \phi^2 P_{t-1|t-1} + \sigma_v^2.$$

In the correction step, we compare the predicted state with the observations and correct the state estimate accordingly by

$$\widehat{x}_{t|t} = \widehat{x}_{t|t-1} + K_t(y_t - \widehat{x}_{t|t-1}), \qquad P_{t|t} = P_{t|t-1} - P_{t|t-1} K_t,$$

where $K_t = P_{t|t-1}(P_{t|t-1} + \sigma_e^2)^{-1}$ denotes the so-called *Kalman gain*. Here, we introduce the predicted state estimate and the predicted covariance denoted by $\widehat{x}_{t|t-1}$ and $P_{t|t-1}$, respectively.

Finally, the posteriors for the filtered and predicted states are given by $x_{t|t} \sim \mathcal{N}(x_{t|t}; \widehat{x}_{t|t}, P_{t|t})$ and $x_{t|t-1} \sim \mathcal{N}(x_{t|t-1}; \widehat{x}_{t|t-1}, P_{t|t-1})$, respectively. After a run of the Kalman filter, we can compute the likelihood of the LGSS model given the parameters θ by

$$p_\theta(y_{1:T}) = \prod_{t=1}^{T} \mathcal{N}(y_t; \widehat{x}_{t|t-1}, P_{t|t-1} + \sigma_e^2),$$

which follows from (2.12). The mean and variance of the initial state are typically assumed to be known, e.g., $\widehat{x}_{1|0} = \mu$ and $P_{1|0} = \sigma_v^2(1-\phi^2)^{-1}$ in this particular model.

We continue by presenting some useful quantities connected with the likelihood, which are used of in many of the papers included in this thesis. The first quantity is known as the *score function* and it is defined as the gradient of the log-likelihood given by

$$\mathcal{S}(\theta') = \nabla \log p_\theta(y)\big|_{\theta=\theta'}. \tag{2.15}$$

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, this is not necessarily true when the number of observations is finite.

The second quantity is known as the *observed information matrix* and it is defined by the negative Hessian of the log-likelihood given by

$$\mathcal{J}(\theta') = -\nabla^2 \log p_\theta(y)\big|_{\theta=\theta'}. \tag{2.16}$$

The observed informations matrix can be seen as a measure of the total amount of information available regarding θ in the data. 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 maximum likelihood parameter estimate (c.f. the second-derivative test in basic calculus).

Moreover, there exists a limiting behaviour for the observed information matrix, which tends to the so-called *expected information matrix* as the number of data points approach infinity. This quantity (also known as the *Fisher information matrix*) is defined as the expected value of the observed information matrix given by

$$\mathcal{I}(\theta') = -\mathbb{E}_y\left[\nabla^2 \log p_\theta(y)\big|_{\theta=\theta'}\right] = \mathbb{E}_y\left[\left(\nabla \log p_\theta(y)\big|_{\theta=\theta'}\right)^2\right], \tag{2.17}$$

where the expectation is evaluated with respect to the data. 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 according to the first term in (2.17) can be seen as the variance of the score function.

Point estimates

The Bayesian parameter inference problem is completely described by (2.10) and everything known about θ is encoded in the posterior. However, we are sometimes interested in computing *point estimates* of the parameter vector. This is done by applying *statistical decision theory* to make decisions about what information from the posterior to take into

	Loss function	Bayes point 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 / Maximum a-posteriori (MAP)

Table 2.1. Different loss functions and the resulting Bayes point estimator.

account in the point-estimate. Consider a *loss function* $L : \Theta \times \Theta \to \mathbb{R}_+$, which takes the parameter and its estimate as inputs and returns a real-valued positive loss. The *expected posterior loss* (or posterior risk) is given by

$$\rho\big(p(\theta), \delta \,|\, y\big) = \int_\Theta L\big(\theta, \delta(y)\big) p(\theta \,|\, y)\, \mathrm{d}\theta,$$

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

$$\delta^\star(y) = \underset{\delta(y)\in\Theta}{\operatorname{argmin}}\, \rho\big(p(\theta), \delta \,|\, y\big).$$

Remark 2.8 (Some common loss functions). In Table 2.1, we present three different Bayes estimators resulting from different choices of the loss function. For example when selecting the quadratic loss function, we have

$$\underset{\delta(y)\in\Theta}{\operatorname{argmin}} \int_\Theta (\widehat{\theta} - \theta)^2 p(\theta \,|\, y)\, \mathrm{d}\theta.$$

Expansion and differentiation of the integral with respect to $\widehat{\theta}$ gives

$$\frac{\partial}{\partial \widehat{\theta}} \left[\widehat{\theta}^2 - 2\widehat{\theta} \int_\Theta \theta p(\theta \,|\, y)\, \mathrm{d}\theta + \int_\Theta \theta^2 p(\theta \,|\, y)\, \mathrm{d}\theta \right] = 0,$$

where the derivative is set to zero to obtain the optimum. Hence, we obtain

$$2\widehat{\theta} - 2 \int_\Theta \theta p(\theta \,|\, y)\, \mathrm{d}\theta = 0,$$

where the solution is given by

$$\widehat{\theta} = \mathbb{E}_y[\theta] = \int_\Theta \theta\, p(\theta \,|\, y)\, \mathrm{d}\theta. \tag{2.18}$$

That is, the Bayes estimator is given by the posterior mean for the quadratic loss function. Similar calculations can be done for other loss functions. Furthermore, selecting the 0-1 loss function together with uniform priors recovers the maximum likelihood estimator. Finally, we note that other more complicated loss functions can be of interest in practice. For example, it could be important to limit the number of estimates smaller than the true value or the number of false positives.

The computation in (2.18) is a common integration problem in Bayesian inference. It turns out that most problems in Bayesian inference corresponds to intractable integration problems. On the other hand, maximum likelihood inference often corresponds to solving convex and non-convex optimisation problems.

Asymptotic properties

The statistical properties of the Bayes estimator depend in general on the choice of prior. It is therefore challenging to state anything general regarding the properties of the parameter estimates when n (or T) is finite. However, the *Bernstein-von-Mises theorem* (Van der Vaart, 2000) states that under some mild regularity conditions the influence of the prior distribution diminishes are the amount of information about θ increase. Note that, this only occurs when the amount of informative observations increases. Moreover, the posterior distribution *concentrates* to a Gaussian distribution centred around the true parameters, i.e., the asymptotic maximum likelihood estimate. Therefore, the Bayes estimator enjoys the same strong asymptotic properties as the maximum likelihood estimator.

As a consequence, the Bayes estimator is *consistent*, *asymptotically Gaussian* 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, see Lehmann and Casella (1998) and Casella and Berger (2001). An estimator is said to be *consistent* if

$$\widehat{\theta} \xrightarrow{\text{a.s.}} \theta^\star,$$

where $\theta^\star$ denotes the true parameters and when $n \to \infty$. That is, the estimate almost surely (with probability one) converges to the true value of the parameter in the limit of infinite data. Furthermore as the estimator is asymptotically Gaussian, the error in the estimate satisfies a central limit theorem (CLT) given by

$$\sqrt{n}\left(\widehat{\theta} - \theta^\star\right) \xrightarrow{\text{d}} \mathcal{N}\left(0, \mathcal{I}^{-1}(\theta^\star)\right), \tag{2.19}$$

when $n \to \infty$. This follows from a second-order Taylor expansion of the log-likelihood around $\theta^\star$. Note that, the expected information matrix $\mathcal{I}(\theta)$ determines the asymptotic accuracy of the estimate.

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 maximum likelihood estimator is the best un-biased estimator in the MSE-sense and there are no better un-biased estimators. This last property is appealing and one might be tempted to say that this estimator is the *best* choice for parameter inference. However, this result is only asymptotically valid. Therefore, other estimators (e.g., Bayes estimators) could have better properties in the finite sample regime.

Finally, there can exist biased estimators with a smaller MSE than the maximum likelihood estimator. This is the result of the so-called *bias-variance trade-off*. One example that we already encountered is the RLS in Remark 2.5, where the estimates are biased but sometimes enjoy a much smaller variance compared with the LS solution.

Non-parametric models

Bayesian non-parametrics (BNPs; Hjort et al., 2010) is an active research field in machine learning and computational statistics. The models introduces in Section 2.2 are all parametric, i.e., the number of parameters p does not grow with the number of observations. In non-parametric models, we assume that p grows with the number of observations, which gives more flexibility to the model as more observations are recorded.

Another perspective of BNPs is that they are infinite stochastic processes. Their construction can be carried out by invoking the *Kolmogorov extension theorem* (Billingsley, 2012, p. 517). This theorem states that any finite subset of an infinite dimensional process is distributed according to the marginal of that process. Hence, we can find an infinite stochastic process by fixing a process in a finite subset of points and applying the extension theorem. Conjugate priors are often used to carry out the prior-posterior update. See Orbanz (2009) for how to construct BNPs by starting from finite-dimensional marginals.

In this section, we briefly discuss two useful BNP models that we make use of in Papers E and F. The first model is the GP (Rasmussen and Williams, 2006) and it is useful for regression and classification. We already encountered the GP in the motivating example connected with Figure 1.1. There are many more applications where GPs are useful for modelling, e.g., airline delays (Hensman et al., 2013) and human faces (Titsias and Lawrence, 2010).

The second model is known as a Dirichlet process (DP; Ferguson, 1973, 1974) and it is useful for clustering and to model probability measures (distributions). An overview of the use of DPs and other BNPs is provided by Fox (2009). DPs are employed to model piece-wise affine systems by Wågberg et al. (2015) with applications in automatic control. In economics, Burda and Harding (2013) have applied DPs to model the heterogeneity in the effects of research and development in companies. Finally, this type of model is also used for detecting haplotypes in genomics as discussed by Xing et al. (2007). A haplotype is unit of genetic information and inferring these from data is important to understand genetic variations in populations of individuals.

Gaussian processes

GPs have their origins in *kriging* methods (Cressie, 1993; Matheron, 1963) from spatial statistics, where they are used to construct elevation maps from measurements. Mathematically, a realisation of a GP is an infinite long vector of real-valued random variables. Hence, we can see this vector as a function and this is why GPs can be used as priors over function spaces. Formally, a GP is an infinite-dimensional Gaussian distribution, where any finite subset of points is jointly distributed according to a Gaussian distribution. We denote a GP by $\mathcal{GP}(m, \kappa)$, where $m(x)$ and κ denote the *mean function* and the *covariance function* (kernel), respectively. These two functions fully specifies the GP and can be defined by

$$m(x) = \mathbb{E}[f(x)], \tag{2.20a}$$

$$\kappa(x, x') = \mathbb{E}\Big[\big(f(x) - m(x)\big)\big(f(x') - m(x')\big)^\top\Big], \tag{2.20b}$$

for some function $f : \mathbb{R}^p \to \mathbb{R}$ that we would like to model by the GP. Both m and κ are considered to be prior choices and encode our prior beliefs about the data in terms of e.g.,

trends, cycles and smoothness. The mean function specifies the average value of the process and the covariance function specifies the correlation between (nearby) samples.

In the left half of Figure 2.4, we present a realisation from a GP prior. Furthermore, we indicate two pairs of points by dotted lines. In the right half of the same figure, we present the covariance function corresponding to the points. In the green case, the covariance function has a high correlation and therefore the probable range of values for x_2 given x_1 is quite narrow. In the orange case, the distance between the points is larger and the correlation in the covariance function is therefore smaller. This illustrates the connection between realisations of the GP and the choice of covariance function.

In Figure 2.5, we present three realisations from two different GP priors. Here, we make use of the squared exponential (SE) covariance function and the Matérn 3/2 covariance function, see Rasmussen and Williams (2006) for details. The former encodes the assumption that the function has an infinite number of continuous derivatives. The latter only assumes one continuous derivative and therefore the realisations are less smooth. We also vary the *length scale* l, which encodes assumptions on the rate of change of the underlying function. Typically, l is treated as a hyper-parameter, which is either estimated from the data or determined by the user.

GPs are useful for non-parametric/non-linear regression, where no particular functional form is assumed for the regression function $f(\cdot)$. A non-parametric regression model can be written as

$$y_i = f(x_i) + \sigma_e e_i, \tag{2.21}$$

with e_i as a standard Gaussian random variable with standard deviation $\sigma_e > 0$. To utilise the GP, we assume the prior distribution

$$f \sim \mathcal{GP}(m, \kappa), \tag{2.22}$$

for the regression function. Hence, we have that both the prior (2.22) and the data likelihood (2.21) are distributed according to Gaussian distributions. We can compute the posterior distribution by Bayes' theorem using the conjugate property given some data $\mathcal{D} = \{x, y\} = \{x_i, y_i\}_{i=1}^n$. The resulting *predictive distribution* evaluated at some *test point* $x_\star$ is given by a Gaussian distribution with an updated mean and covariance function computed by

$$f(x_\star)|\mathcal{D} \sim \mathcal{N}\Big(x_\star; \mu_f(x_\star|\mathcal{D}), \sigma_f^2(x_\star|\mathcal{D})\Big), \tag{2.23a}$$

$$\mu_f(x_\star|\mathcal{D}) = \kappa_\star^\top \Big[\kappa\big(x, x\big) + \sigma_e^2 \mathbf{I}_n\Big]^{-1} y, \tag{2.23b}$$

$$\sigma_f^2(x_\star|\mathcal{D}) = \kappa\big(x_\star, x_\star\big) - \kappa_\star^\top \Big[\kappa\big(x, x\big) + \sigma_e^2 \mathbf{I}_n\Big]^{-1} \kappa_\star + \sigma_e^2. \tag{2.23c}$$

Here, we introduce $\kappa_\star = \kappa\big(x_\star, x\big)$ to denote the covariance between the test value and the sampling points. An example of a predictive GP posterior was given in Figure 1.1, where the mean μ_f is plotted as a solid line. The confidence intervals are computed by using the variance in the predictive posterior σ_f^2. We return to using GPs to model the posterior distribution of an SSM in Chapter 4 and in Paper E.

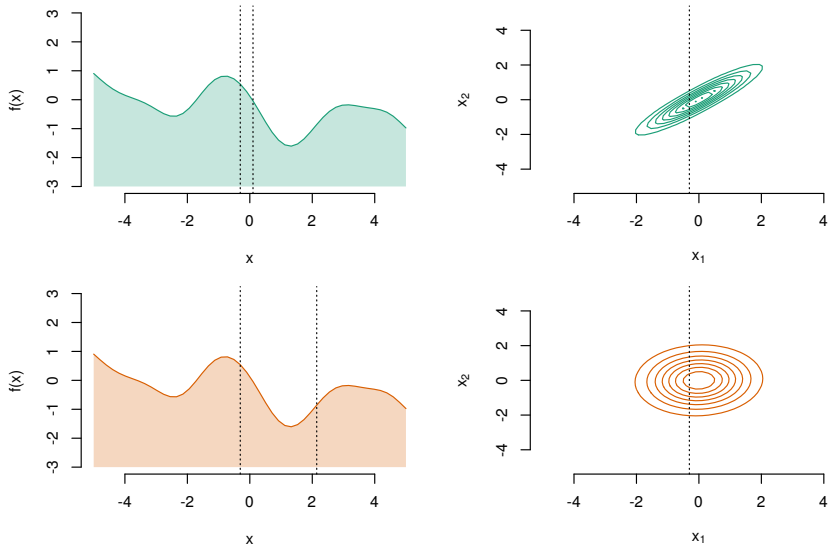


Figure 2.4. A realisation from a GP prior with the SE covariance function for two pairs of points indicated by dotted lines.

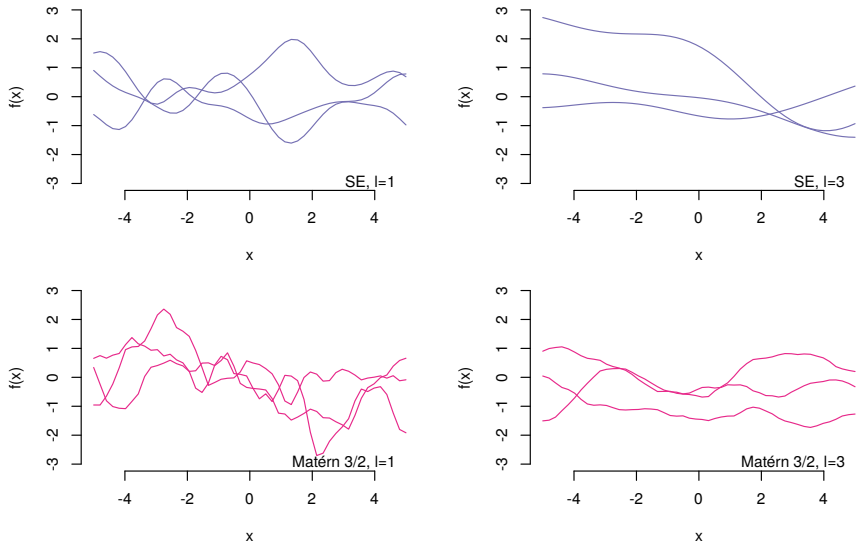


Figure 2.5. Realisations from a GP prior with the SE covariance function (purple) and the Matérn covariance function (magenta) for two different length scales l.

Dirichlet processes

A realisation G of a DP is a random discrete probability distribution in the form of an empirical distribution, i.e.,

$$G(\mathrm{d}\theta) = \sum_{k=1}^{\infty} w_k \delta_{\theta_k}(\mathrm{d}\theta). \tag{2.24}$$

Here, the weights $\{w_k\}_{k=1}^{\infty}$ and locations $\{\theta_k\}_{k=1}^{\infty}$ are random variables. Furthermore, we have that $\sum_{k=1}^{\infty} w_k = 1$ with probability 1, which is why G can be interpreted as a probability measure. Let $\mathcal{DP}(\alpha, G_0)$ denote a DP with *concentration parameter* $\alpha > 0$ and *base measure* G_0. We say that G is distributed according to a DP if all of its marginal distributions are Dirichlet distributed. This was proved by Ferguson (1973) and is in analogue with the Gaussian marginals required for the GP. Hence, if G_0 is a probability measure on the space $(\Omega, \mathcal{F})$, we have that

$$\Big(G(A_1), G(A_2), \ldots, G(A_N)\Big) \sim \mathcal{D}\Big(\alpha G_0(A_1), \alpha G_0(A_2), \ldots, \alpha G_0(A_N)\Big), \tag{2.25}$$

for any finite (measurable) partition $A_{1:N}$ of Ω. Here, $\mathcal{D}(\alpha)$ denotes the Dirichlet distribution with concentration parameter $\alpha > 0$.

Note that the expected value of G is the base measure and therefore G has the same support as G_0. Moreover, G is discrete with probability one even if the base measure is continuous. In Figure 2.6, we present two realisations from a DP using $\alpha = 1$ (green) and $\alpha = 10$ (orange) and the standard Gaussian as G_0. We note that a larger concentration parameter results in more similar weights, where we can almost guess the underlying base measure. Conversely, most probability mass is allocated to a small number of components when the concentration parameter is small.

We can recover these properties analytically by studying the predictive distribution of a DP. Assume that we obtain some data generated from the model given by

$$G \sim \mathcal{DP}(\alpha, G_0), \qquad \theta_i \,|\, G \sim G,$$

for $i = 1, 2, \ldots$. The predictive distribution is given by the marginalisation

$$p(\theta_\star \,|\, \theta_{1:n}) = \int G(\theta_\star) p(G \,|\, \theta_{1:n}) \mathrm{d}G,$$

which is possible to carry out in closed-form. The result is a Pólya urn scheme discussed by Blackwell and MacQueen (1973), which can be expressed mathematically by

$$\theta_\star \,|\, \theta_{1:n} \sim \frac{\alpha}{\alpha + n} G_0 + \frac{1}{\alpha + n} \sum_{i=1}^{n} n_i \delta_{\theta_i}. \tag{2.26}$$

Here, n_i denotes the number of parameters that are identical to θ_i, i.e.,

$$n_i = \sum_{j=1}^{n} \mathbb{I}[\theta_i = \theta_j],$$

where $\mathbb{I}[A]$ denotes the indicator function.

This Pólya urn scheme has an interesting and rather amusing interpretation known as the *Chinese restaurant process* (CRP). In this interpretation, we see each parameter θ_i as a guest arriving to a Chinese restaurant. The first guest choose a random dish from the menu and sits down at some table. The second guest can either select a new random dish from the menu or join the first guest at his/her table and have the same dish. This continues on forever and the probability of joining an existing table is proportional to the number of guest n_i already sitting at that table.

Hence, we can conclude from the CRP and (2.26) that the DP is a discrete process with a non-zero probability of ties. That is, that guests tend to cluster around the existing dishes in the restaurant. Furthermore, we are more likely to sample from the base measure (choose a new dish) if $\alpha \gg n$, which means that the predictive posterior concentrates to the base measure. If $\alpha \ll n$, we often sample from the existing parameters, which gives many ties and a strong clustering behaviour. This corresponds to that a few samples obtain most of the probability mass as seen in Figure 2.6.

A third alternative view of a DPs are to consider them as the results of a *stick-breaking process* (SBP; Sethuraman, 1994). This is useful for generating realisations from a DP by using the empirical distribution in (2.24). The weights and locations can be generated by a SBP, i.e.,

$$w_k = V_k \prod_{i=1}^{k-1}(1 - V_i), \quad V_k \sim \mathcal{B}(1, \alpha), \quad \theta_k \sim G_0,$$

where $\mathcal{B}(a, b)$ denotes the Beta distribution with shape parameters $a > 0$ and $b > 0$. The name SBP comes from that w_k can be seen as a part of a stick of unit length. The product represents the length of the remaining stick at iteration k and V_k denotes the fraction that is broken off the stick. In the left part of Figure 2.7, we present an illustration of the SBP. At each iteration, we break off a proportion of the remaining stick (green) given by V_k. We collect the resulting pieces and combine them with samples from the base measure to obtain the random probability distribution presented in the right part of the figure.

We can create a hierarchical model to utilise the discreteness of the DP for clustering. This results in a DP mixture (DPM; Antoniak, 1974), which can be expressed as

$$G \sim \mathcal{DP}(\alpha, G_0), \qquad \theta_k \sim G, \qquad x_k \sim p(\cdot \,|\, \theta_k), \tag{2.27}$$

for $k = 1, \ldots, n$. That is, we generate data from a distribution which is parametrised by the random parameters drawn from the random probability distribution generated by a DP. In practice, we make use of a parametric distribution for the data and obtain a clustering model as some x_k shares the same parameters. We return to make use of DPMs for modelling the heterogeneity of the individual random effects in a mixed effects model in Paper G.

Outlook and extensions

In this chapter, we have presented a few popular parametric models for different types of data. Regression models are discussed in many different textbooks and the interested reader is referred to Hastie et al. (2009) and McCullagh and Nelder (1989) for more information. Time series models such as AR and autoregressive moving average (ARMA) models with

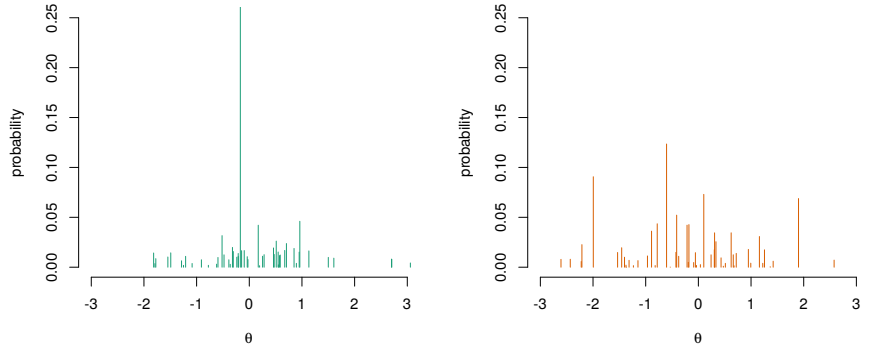


Figure 2.6. Realisations from a DP prior with a standard Gaussian as the base measure and the concentration parameter $\alpha = 1$ (green) and $\alpha = 10$ (orange).

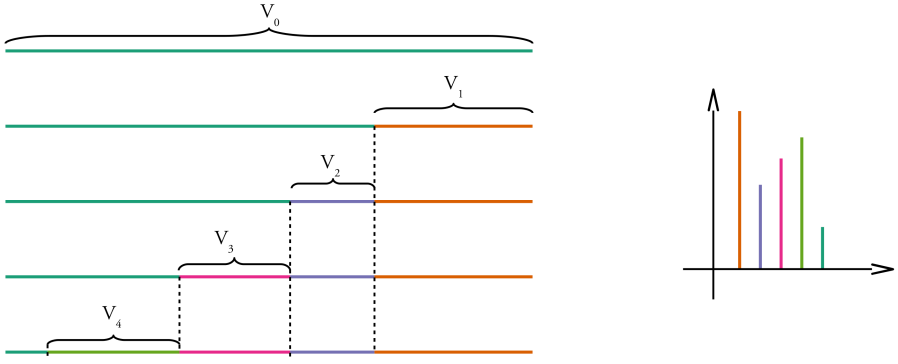


Figure 2.7. Left: illustration of the stick-breaking process in which a proportion V_k of the remaining stick (green) is broken off at each iteration. Right: Illustration of the resulting realisation of the DP.

extensions are thoroughly introduced by Tsay (2005), Shumway and Stoffer (2011) and Brockwell and Davis (2002). For more information about SSMs and interesting extensions, see Douc et al. (2014), Cappé et al. (2005) and Ljung (1999). Kalman filtering is an important topic for LGSS models and a book long treatment is provided by Kailath et al. (2000).

Graphical models are another large family of models, where SSMs corresponds to a specific instance, see Paper A. This type of models is useful in modelling everything from images to text documents. A good introduction to this subject is provided in the book by Koller and Friedman (2009) and Chapter 8 in the book by Bishop (2006). A few more examples of models are presented in the papers included in this thesis. For example, we make use of so-called copula models (Nelsen, 2007) in Section 6.3 (page 253) of Paper E.

Finally, more information regarding Bayesian inferences are found in the books by Robert (2007) and Gelman et al. (2013). The statistical properties of Bayes estimators are further discussed in Lehmann and Casella (1998) and Berger (1985).

3
Monte Carlo methods

A common problem when making use of Bayesian inference for many models of interest is analytical intractability. From Chapter 2, we know that this can be the result of an intractable likelihood or due to the fact that the prior-posterior update cannot be carried out in closed-form. In these situations, we have to resort to approximations which are usually based on variational inference or statistical simulation. In this thesis, we focus on the latter approach by using Monte Carlo methods. This family of methods make use of random sampling for integration, optimisation or to sample from some complicated probability distribution. In Chapter 2, we noted that many problems in Bayesian inference can be expressed as integrals and therefore Monte Carlo methods are useful.

As discussed by Eckhardt (1987), Monte Carlo methods were first introduced by the Polish-American mathematician Stanislaw Ulam [1909-1984] in cooperation with the Hungarian-American mathematician John von Neumann [1903-1957] at the Los Alamos Scientific Laboratory in 1946. The first application was to simulate neutron transports in the shielding material used for nuclear weapons research. These methods were quickly disseminated into physics and chemistry to simulate complicated phenomena.

More elaborate Monte Carlo methods based on the use of Markov chains were later proposed by Metropolis et al. (1953) and extended by Hastings (1970). The resulting algorithm is known as the Metropolis-Hastings (MH) algorithm and is a member of the larger family of Markov chain Monte Carlo (MCMC) methods. Another important MCMC method is known as Gibbs sampling and was proposed by Geman and Geman (1984).

In the late 1980s, Monte Carlo methods became a common tool to approximate the posterior distribution for many interesting problems in statistics. Ever since, it has been an important enabler for Bayesian inference and is usually taught in most courses on the subject. In the beginning of the 1990s, sequential versions of Monte Carlo algorithms were proposed by Stewart and McCarty (1992), Gordon et al. (1993) and Kitagawa (1996). These methods are

usually referred to as particle filters or sequential Monte Carlo (SMC) methods. A useful combination of MCMC and SMC was proposed by Beaumont (2003) based on a heuristic argument. The algorithm was later formalised and analysed by Andrieu and Roberts (2009) and Andrieu et al. (2010). The resulting algorithms are known as pseudo-marginal and particle MCMC algorithms.

In this chapter, we present a number of Monte Carlo methods together with their properties and applications. The main aim of the chapter is to provide the reader with an understanding of the opportunities and problems that are connected with each algorithm. In Chapter 4, we outline some strategies to mitigate these problems, which are applied in the papers included in this thesis.

We begin this chapter by introducing standard Monte Carlo based on a number of different approaches using independent samples. Moreover, we discuss SMC, MCMC and the pseudo-marginal Metropolis-Hastings (PMMH) algorithm for sampling from more complicated models. Finally, we provide the reader with an outlook and references for further study.

Empirical approximations

Monte Carlo methods are a collection of statistical simulation methods based on sampling. They are particularly useful for approximating high-dimensional integration problems. For example, a common problem in Bayesian inference is to compute the expected value of some integrable *test function* $\varphi : \mathcal{X} \to \mathbb{R}$ given by

$$\pi[\varphi] \triangleq \mathbb{E}_\pi\big[\varphi(x)\big] = \int_{\mathcal{X}} \varphi(x)\,\pi(x)\,\mathrm{d}x, \tag{3.1}$$

where $\pi(x)$ denotes a (normalised) *target distribution*. In the basic *vanilla* formulation of Monte Carlo methods, we assume that we can simulate IID *particles* (or samples) from the target distribution. However, we do not require to be able to evaluate the target point-wise. In what follows, we encounter Monte Carlo methods which require point-wise evaluation of the target but not being able to simulate from it directly.

The first step in computing a Monte Carlo estimate of (3.1) is to form an empirical approximation of the target distribution given by

$$\widehat{\pi}^N_{\mathrm{MC}}(\mathrm{d}x) = \sum_{i=1}^{N} \delta_{x^{(i)}}(\mathrm{d}x), \tag{3.2}$$

using the particles $\{x^{(i)}\}_{i=1}^N$ generated from the target distribution π. Here, $\delta_{x'}(\mathrm{d}x)$ denotes the Dirac distribution placed at $x = x'$. The *vanilla* estimator follows from the second step by combining (3.2) into (3.1) to obtain

$$\widehat{\pi}^N_{\mathrm{MC}}[\varphi] \triangleq \int_{\mathcal{X}} \varphi(x)\,\widehat{\pi}(\mathrm{d}x) = \frac{1}{N}\sum_{i=1}^{N} \varphi\big(x^{(i)}\big), \tag{3.3}$$

which follows from the properties of the Dirac distribution.

The main advantage of Monte Carlo methods over some of their alternatives are solid statistical properties. The estimator is un-biased and *strongly consistent* by the strong law of large numbers (SLLN), i.e.,

$$\widehat{\pi}_{\text{MC}}^{N}[\varphi] \xrightarrow{\text{a.s.}} \pi[\varphi],$$

when $N \to \infty$. Moreover, it is possible to construct a central limit theorem (CLT) for the vanilla Monte Carlo estimator given by

$$\sqrt{N}\left(\widehat{\pi}_{\text{MC}}^{N}[\varphi] - \pi[\varphi]\right) \xrightarrow{\text{d}} \mathcal{N}\left(0, \sigma_{\text{MC}}^{2}\right), \qquad \sigma_{\text{MC}}^{2} \triangleq \mathbb{V}_{\pi}[\varphi] < \infty,$$

when $N \to \infty$ and $\varphi(x)$ has a finite second moment. Hence, we see that the Monte Carlo estimator is asymptotically un-biased with Gaussian errors. Furthermore, the variance of the error decreases as $1/N$ independently of the dimension of the problem. This is one of the main advantages of the Monte Carlo methods compared with common numerical integration methods based on quadratures, see e.g., Stoer and Bulirsch (1993).

Three sampling strategies

The main difficulty with applying vanilla Monte Carlo to many interesting problems is generating good samples from the target distribution. In this section, we present three different approaches for generating samples or approximating (3.3) directly using: (i) independent sampling, (ii) sequential sampling and (iii) Markov chain sampling.

Independent Monte Carlo

All Monte Carlo methods rely on generating random numbers. In practice, we cannot generate truly random numbers using computers. Instead, we make use of *pseudo-random numbers*, which are constructed to pass many statistical tests for randomness. In the following, we refer to pseudo-random numbers simply as random numbers. A *linear congruential generator* can be applied to generate uniform random numbers by

$$x^{(i)} = \left[ax^{(i-1)} + b\right] (\text{mod}\, m),$$

for $i = 1, 2, \ldots$. Here, a, b and m are integers (usually large) determined by the specific generator. For example, the programming language Python makes use of the Mersenne Twister (Matsumoto and Nishimura, 1998), which has a period of $2^{19937} - 1$. In the left part of Figure 3.1, we present random samples generated by the Mersenne Twister on the unit square, i.e., $u^{(i)} \sim \mathcal{U}[0, 1]^2$. We note that the samples do not fill the square evenly but seem to concentrate in certain areas. This is a drawback with pseudo-random numbers and can result in slow convergence rates when the dimension of the problem increases.

There is an alternative method for generating a random sample called *quasi-random number generators*. In the right part of Figure 3.1, we present the output from one such algorithm based on Sobol sequences (Sobol, 1967). The main benefit with this type of sequence is that it fills the space more evenly and this can improve convergence in many applications. The main drawback is that analysis of estimators based on Sobol sequences are challenging due to their deterministic nature, see Owen (2013). Quasi-random numbers are useful in

quantitative finance (Glasserman, 2004; Niederreiter, 2010) and for sequential Monte Carlo sampling, see Section 3.4.

Quantile transformation

For many distributions, we can obtain random samples given uniform random numbers and by using the inverse CDF method. We illustrate this by sampling from the exponential distribution with rate $\lambda > 0$ for which the CDF is given by

$$G(x) = \mathbb{P}(X \leq x) = 1 - \exp(-\lambda x),$$

when $x \geq 0$ and zero otherwise. We can directly compute the inverse CDF as

$$G^{-1}(p) = -\frac{\log(1-p)}{\lambda},$$

which is known as the *quantile function* evaluated at $p \in (0, 1)$. Hence, we can obtain an exponentially distributed random number by

$$x^{(i)} = -\frac{\log\left(u^{(i)}\right)}{\lambda},$$

where $u^{(i)}$ denotes a uniform random number. We present an illustration of the inverse CDF method in the left part of Figure 3.2. Here, we can directly obtain the random sample 1.9 from the uniform random variable 0.86. It is also possible to sample from an empirical CDF constructed from some data, which is presented in the right part of the same figure. This is the basis of an useful approach to non-parametric statistics known as the *bootstrap method* (Efron, 1979; Davison and Hinkley, 1997). The main benefit with the bootstrap is that it does not rely on asymptotics to compute confidence intervals and to carry out tests. We return to resampling schemes like the bootstrap in Section 3.2.2.

Importance sampling

One approach to approximate (3.3) using independent samples is *importance sampling* (Marshall, 1956). This algorithm makes use of a proposal distribution to simulate from the target, which is useful when direct simulation from the target is difficult or impossible. The proposal distribution $q(x)$ is usually selected to be simple to simulate from and allow for cheap point-wise evaluations. The discrepancy between the target and proposal is then compensated for by an importance weight. The main idea is to rewrite (3.1) by

$$\pi[\varphi] = \int_{\mathcal{X}} \varphi(x)\,\pi(x)\,\mathrm{d}x = \int_{\mathcal{X}} \varphi(x)\,\underbrace{\frac{\pi(x)}{q(x)}}_{\triangleq w(x)}\,q(x)\,\mathrm{d}x = q[w\varphi],$$

where $w(x)$ denotes the importance weight. Note that, we require to be able to evaluate the target point-wise, c.f. vanilla Monte Carlo. We can form an analogue estimator to (3.3) by writing

$$\widehat{\pi}^N_{\text{IS}}[\varphi] \triangleq \sum_{i=1}^{N} w^{(i)}\varphi(x^{(i)}), \tag{3.4}$$

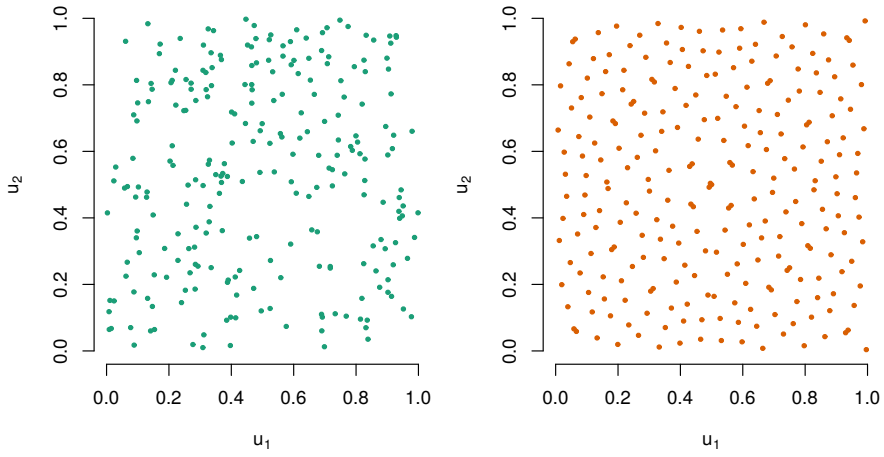


Figure 3.1. Pseudo-random samples from $\mathcal{U}[0,1]^2$ generated using Mersenne Twister (left) and Sobol sequences (right).

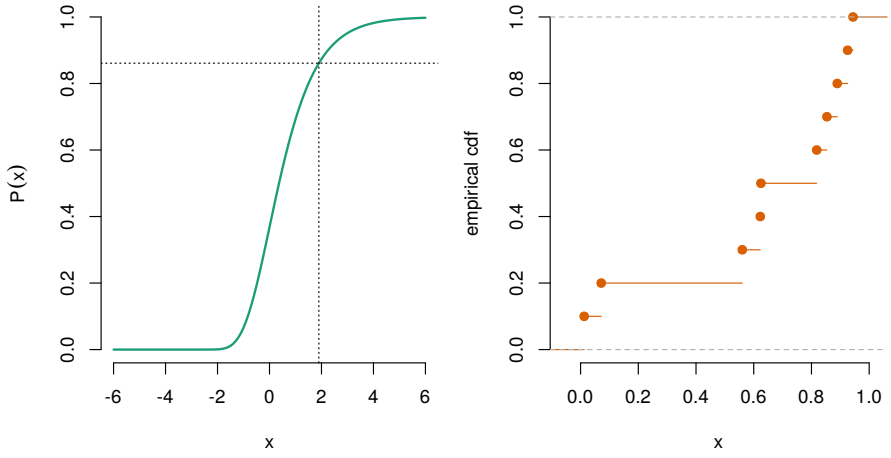


Figure 3.2. Illustration of the quantile transformation method to generate random variables. A uniform variable is generated corresponding to $P(x)$ for which x is determined by quantile transformation (dotted lines).

where $x^{(i)} \sim q(x)$ and $w^{(i)} \triangleq w(x^{(i)})$. In Figure 3.3, we present two different cases where importance sampling can be useful. In the left plot, we consider sampling from the entire target using a similar Gaussian proposal which is simple to sample from. The difference between the two distributions is indicated by the purple area. The main dissimilarity lies in the right tail, which falls off slower for the target than for the proposal. In the right plot, we are interested in computing e.g., the probability of obtaining large values of the target. Therefore, we construct a proposal that focuses on the upper tail behaviour. This is useful in many applications where extreme values are important, e.g., in survival models, hydrology and quantitative finance, see McNeil et al. (2010) and Embrechts et al. (1997).

For importance sampling to work, the support of $q(x)$ has to contain the support of $\varphi(x)\pi(x)$, i.e., $\mathsf{supp}(\varphi\pi) \subset \mathsf{supp}(q)$. In that case, the estimator (3.4) inherits all of the properties from the vanilla Monte Carlo estimator, i.e., it is un-biased, consistent and asymptotically Gaussian. However, the asymptotic variance can be computed by

$$\sigma_{\text{IS}}^2 = \int_{\mathcal{X}} \frac{\big(\varphi(x)\pi(x)\big)^2}{q(x)} \,\mathrm{d}x - \pi[\varphi]^2 = \int_{\mathcal{X}} \frac{\big(\varphi(x)\pi(x) - \pi[\varphi]q(x)\big)^2}{q(x)} \,\mathrm{d}x.$$

A good choice of $q(x)$ should therefore minimise this expression, i.e., by using a proposal proportional to $\varphi(x)\pi(x)$. However, in practice this is often difficult due to the requirement that the proposal should be simple to simulate from and to evaluate point-wise.

In the following, we consider importance sampling from un-normalised target distributions $\gamma(x)$. In this case, we can write the target as $\pi(x) = \gamma(x)\mathcal{Z}^{-1}$ where the normalisation constant $\mathcal{Z}$ is unknown. However, we can make use of importance sampling in this settings as well. The main difference is that the weights are un-normalised and computed by

$$\widetilde{w}(x) = \frac{\gamma(x)}{q(x)}.$$

The resulting estimator is given by

$$\widehat{\pi}_{\text{SNIS}}^N[\varphi] \triangleq \sum_{i=1}^N \underbrace{\frac{\widetilde{w}^{(i)}}{\sum_{j=1}^N \widetilde{w}^{(j)}}}_{\triangleq w^{(i)}} \varphi\big(x^{(i)}\big), \tag{3.5}$$

which is known as the self-normalised importance sampling (SNIS) estimator. The estimator (3.5) is strongly consistent and asymptotically Gaussian, see Owen (2013) for details. However, it is biased for finite N and the bias is proportional to $\mathcal{O}(N^{-1})$.

Sequential Monte Carlo

In some applications, we would like to compute the expectation of a test function with respect to a sequence of probability distributions $\{\pi_t(x_{0:t})\}_{t=0}^T$, where $x_t \in \mathcal{X}$. As for importance sampling, we assume that the target can be written as $\pi_t(x_{0:t}) = \gamma_t(x_{0:t})\mathcal{Z}_t^{-1}$, where $\gamma_t(x_{0:t})$ denotes the un-normalised target and $\mathcal{Z}_t$ denotes the normalisation constant. Here, we assume that it is possible evaluate the un-normalised target point-wise but the

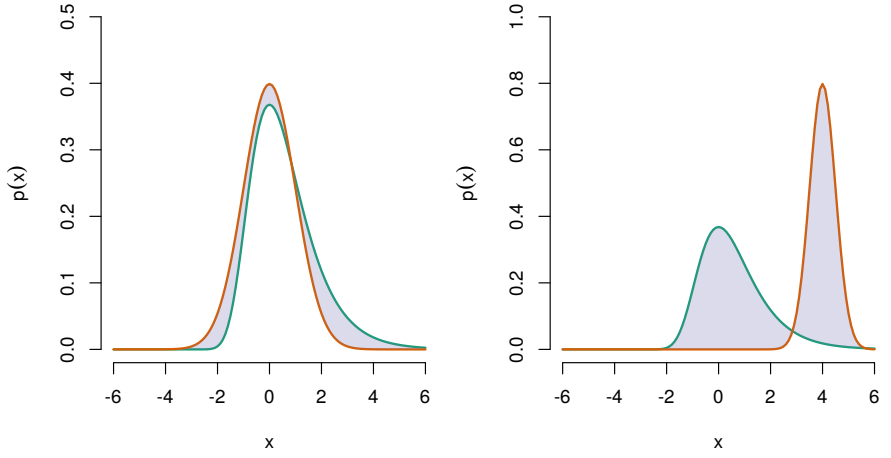


Figure 3.3. Illustration of importance sampling of the target (green) using a proposal (orange). The difference between the target and proposal densities is indicated by the purple area. Two cases are considered: sampling the entire distribution (left) and its upper tail (right).

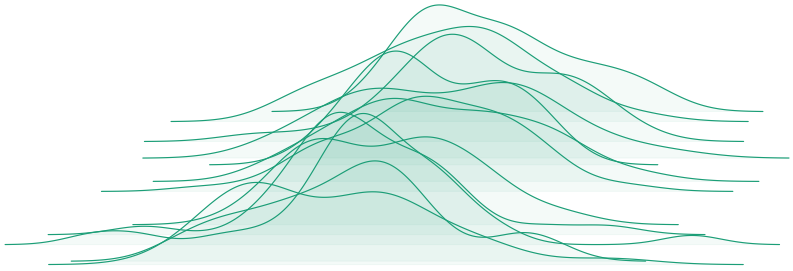


Figure 3.4. An illustration of the evolution of the target distribution $\pi_t(x_{0:t})$ over time.

normalisation constant can be unknown, c.f. SNIS. An illustration of the sequence of targets is given in Figure 3.4.

This set-up is useful in online settings where observations arrives sequentially or when the number of observations is large. Note that the online setting can also be artificially introduced by so-called *tempering methods* where a sequential target can be obtain from a static. A simple annealing scheme is often used for this and it consists of setting $\pi_t(x) = \pi^{\phi_t}(x)$, where ϕ_t is a parameter varying from zero to one as t increases.

A powerful approach for taking advantage of the sequential structure in certain target distributions is to make use of SMC samplers (Del Moral et al., 2006). These methods can be seen as an extension of importance sampling algorithms, where the proposal is constructed sequentially. That is, we assume that the proposal can be expressed by

$$q_t(x_{0:t}) = q_{t-1}(x_{0:t-1})q_t(x_t \mid x_{t-1}) = q_0(x_0)\prod_{r=1}^{t} q_r(x_r \mid x_{r-1}).$$

Hence, we can apply importance sampling to first sample x_0 and then sequentially propose samples conditioned on the previous, i.e., $x_r \sim q_t(x_r \mid x_{0:r-1})$ for $r = 1, \ldots, t$. The resulting importance weights are also computed sequentially by

$$w_t(x_{0:t}) = \frac{\gamma_t(x_{0:t})}{q_t(x_{0:t})} = \frac{\gamma_{t-1}(x_{0:t-1})}{q_{t-1}(x_{0:t-1})} \frac{\gamma_t(x_{0:t})}{\gamma_{t-1}(x_{0:t-1})\, q_t(x_t \mid x_{0:t-1})}.$$

This set-up is more efficient compared with standard importance sampling when x is a high-dimensional vector. This is due to the aforementioned problems with constructing good proposals for the importance sampling algorithms. The benefit with SMC algorithms is that each proposal only extends the state from one time step to another, which simplifies the construction of the proposal.

The resulting algorithm is known as *sequential importance sampling* (SIS). The main drawback with this method is that the variance of the estimates increases rapidly with t. This is the result of *particle depletion*, where only a single particle (or sample) carries all the importance weight. Instead, we can introduce a resampling step to mitigate this effect and to only keep the particles with large weights. This is the major development in the first papers about particle filtering also known as *sequential importance sampling with resampling* (SIR). Later, the SIR algorithm was generalised to the SMC algorithm, which can make use of more elaborate proposal distributions.

The SIR algorithm is based on carrying out three steps during each iteration: (i) resampling, (ii) propagation and (iii) weighting. The output from each iteration is a particle system given by the particles (samples) $\{x_{0:t}^{(i)}\}_{i=1}^N$ and their corresponding self-normalised importance weights $\{w_t^{(i)}\}_{i=1}^N$. From this system, we can construct an empirical approximation of $\pi_t(x_{0:t})$ by

$$\widehat{\pi}_{t,\mathrm{SMC}}^N(\mathrm{d}x_{0:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{x_{0:t}^{(i)}}(\mathrm{d}x_{0:t}), \tag{3.6}$$

together with an estimator in analogue with (3.4) given by

$$\widehat{\pi}^N_{t,\mathrm{SMC}}[\varphi] \triangleq \sum_{i=1}^{N} w_t^{(i)} \varphi(x_{0:t}^{(i)}), \tag{3.7}$$

where $w_t^{(i)} \triangleq w_t(x_{0:t}^{(i)})$. We proceed by briefly presenting each step and refer the interested reader to Section 3 (page 123) in Paper A, Doucet and Johansen (2011) and Del Moral et al. (2006) for further details.

(**Resampling**) The resampling step multiplies particles with large importance weights and discard particles with a small weight. This is done in a stochastic manner to focus the attention of the SIR algorithm to the relevant part of the state space, i.e., in locations with high probability under the target distribution. This operation is carried out by sampling *ancestor indices* denoted $a_t^{(i)}$ for each of the particles. Here, $a_t^{(i)}$ is interpreted as the index of the particle at time $t-1$ from which particle i at time t originates from. This can be expressed as simulating from a multinomial distribution with probabilities given by

$$\mathbb{P}(a_t^{(i)} = j) = w_{t-1}^{(j)}, \qquad j = 1, \ldots, N, \tag{3.8}$$

for $i = 1, \ldots, N$. This operation is known as *multinomial resampling* as it corresponds to sampling from the distribution with the same name. This can also be seen as from the empirical CDF in the right of Figure 3.2 generated from the normalised particle weights.

In Figure 3.5, we present an illustration of the effect of resampling and the meaning of the ancestor indices. The green line indicates the surviving particle genealogy during a run of the SIR algorithm. We see that the the genealogy collapses into a single trajectory at time $t = 8$. This corresponds to a single surviving particle and therefore only one sample in the empirical approximation of the target. This is known as *particle degeneracy*, which results in that estimates of expected values of test functions with respect to $x_{0:t}$ can suffer from a large variance when t is large. However, expectations with respect to only x_t can be estimated using many samples, which results in a lower variance. The particle degeneracy problem is a result of the fact that resampling discards particles with a non-zero probability at every iteration of the algorithm.

To mitigate this problem, we can use alternative resampling schemes or particle smoothers as discussed in Remark 3.2. Better alternatives to multinomial resampling are based on stratified sampling from the CDF of the weights. In this thesis, we make use of systematic resampling, which exhibits good properties in many applications. The interested reader is referred to Douc and Cappé (2005), Hol et al. (2006) and Murray et al. (2015) for more information and examples of resampling algorithms.

(**Propagation**) In the propagation step, we simulate the particle system from time $t-1$ one step forward in time to obtain the particle system at time t. Each particle is propagated using the proposal distribution by

$$x_t^{(i)} \sim q_t\left(x_t \,|\, x_{t-1}^{a_t^{(i)}}\right), \qquad x_{0:t}^{(i)} \triangleq \left\{x_{0:t-1}^{a_t^{(i)}}, x_t^{(i)}\right\}, \tag{3.9}$$

for $i = 1, \ldots, N$. There are many different choices for $q_t(\cdot)$ and tailoring them for the

problem at hand is important to achieve efficient algorithms. We return to this problem in Section 3.4 and in Section 6.3 (page 142) of Paper A.

(**Weighting**) Each particle is assigned an importance weight computed by a weight function defined by

$$W_\theta(x_t, x_{0:t-1}) \triangleq \frac{\gamma_t(x_{0:t})}{\gamma_{t-1}(x_{0:t-1})\, q_t(x_t \mid x_{0:t-1})} w_{t-1}.$$

The resulting un-normalised and normalised weights are computed by

$$\widetilde{w}_t^{(i)} = W_\theta\left(x_t^{(i)}, x_{0:t-1}^{a_t^{(i)}}\right), \qquad w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_{j=1}^N \widetilde{w}_t^{(j)}}, \tag{3.10}$$

for $i = 1, \ldots, N$. The weights account for the discrepancy between the proposal and the target distribution in analogue with importance sampling. Furthermore, it is possible to estimate the unknown normalisation constant $\mathcal{Z}$ for the target by making use of the un-normalised importance weights $\widetilde{w}_t^{(i)}$, see Del Moral et al. (2006) for details.

In Kronander et al. (2014a) and Svensson et al. (2015), we make use of SMC for rendering animations in computer graphics and marginalising hyperparameters in GP priors, respectively. SMC algorithms have also been proposed for inference in mixture models (Fearnhead, 2004; Ulker et al., 2010), SSMs (see Remark 3.1) and graphical models (Naesseth et al., 2014).

Remark 3.1 (Particle filtering). The filtering solution to the state inference problem for the LGSS model is given by the recursion in Example 2.7 on page 32. Although, this recursion is intractable for most SSMs it can be approximated using SMC methods. We refer to the resulting algorithm as the particle filter (Gordon et al., 1993), which is an integral component in most of the papers included in this thesis. The particle filter is the SMC algorithm targeting the filtering distribution in an SSM, i.e., $\pi_t(x_{0:t}) = p_\theta(x_{0:t} \mid y_{1:t})$. We can compute the mean of this distribution at time t by using $\varphi(x_t) = x_t$ in (3.7),

$$\widehat{x}_{t|t} \triangleq \widehat{\pi}_{t,\mathrm{SMC}}^N[x_t] = \int_{\mathcal{X}^{t+1}} x_t\, \widehat{p}_\theta^N(x_{0:t} \mid y_{1:t})\, \mathrm{d}x_{0:t} = \sum_{i=1}^N w_t^{(i)} x_t^{(i)}, \tag{3.11}$$

where $\{\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N\}_{t=0}^T$ denotes the particle system generated by the particle filter.

The main problem is that we cannot evaluate $p_\theta(x_{0:t} \mid y_{1:t})$ directly due to an unknown normalisation factor. Instead, we let the SMC algorithm target $\gamma_t(x_{0:t}) = f_\theta(x_t \mid x_{t-1}) g_\theta(y_t \mid x_t) \gamma_{t-1}(x_{0:t-1})$, which follows from the Bayesian filtering recursions (2.14). A simple choice for the proposal in this setting is given by

$$q_t(x_t \mid x_{0:t-1}) = f_\theta(x_t \mid x_{t-1}), \qquad W_\theta(x_t, x_{0:t-1}) = g_\theta(y_t \mid x_t),$$

where the weight function follows directly from the target and the choice of proposal by (3.10). That is, using the state dynamics as the proposal and the density of the observations as the weight function. We refer to this version of the algorithm as the *bootstrap particle filter* (BPF).

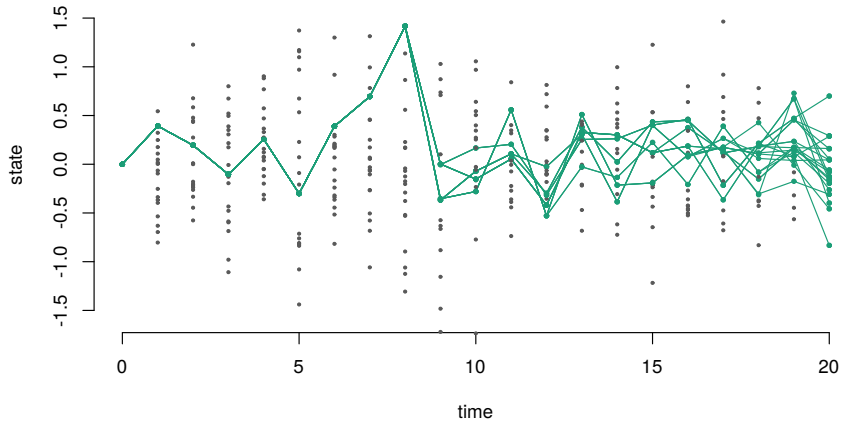


Figure 3.5. The particle genealogy generated by the resampling step in the SMC algorithm. The green line/dots are the particles that survive the repeated resampling steps. The discarded particles are presented as grey dots.

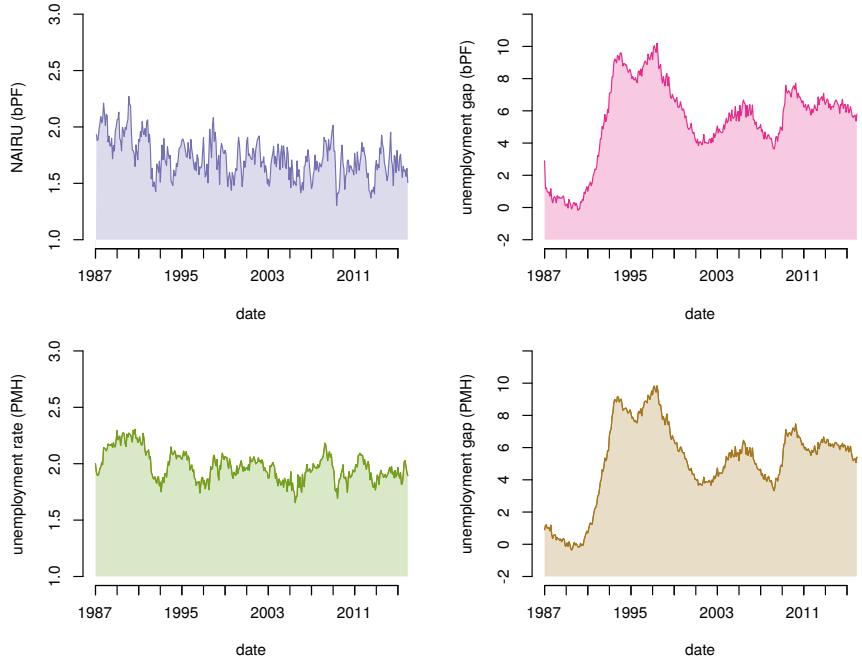


Figure 3.6. The estimated NAIRU (left) together with the estimated unemployment gap (right) for Sweden during the period January, 1987 to December, 2015. The estimates obtain by a BPF (Example 3.3) are indicated by purple/magenta and the estimates from the PMH algorithm (Example 3.13) are indicated by green/brown.

The BPF can be applied to estimate the likelihood $p_\theta(y_{1:T})$, which is useful for parameter inference in SSMs. The predictive likelihood (2.12) can be approximated using the particle system by

$$\widehat{p}_\theta^N(y_t \mid y_{1:t-1}) = \int_{\mathcal{X}} g_\theta(y_t \mid x_t)\widehat{p}_\theta^N(x_t \mid y_{1:t-1})\,\mathrm{d}x_t = \frac{1}{N}\sum_{i=1}^{N} w_t^{(i)}.$$

The estimator of the likelihood follows from the decomposition in (2.11) and is given by

$$\widehat{p}_\theta^N(y_{1:T}) = \frac{1}{N^{T+1}}\prod_{t=0}^{T}\sum_{i=1}^{N} w_t^{(i)}. \tag{3.12}$$

We refer the interested reader to Section 4.1 (page 128) in Paper A and Doucet and Johansen (2011) for more information about the bPF and other particle filtering algorithms.

Remark 3.2 (Particle smoothing). In Remark 2.7, we introduced the Bayesian filtering recursions to sequentially compute the marginal filtering distribution $\pi_t(x_t) \triangleq p_\theta(x_t \mid y_{1:t})$. We already know that the SMC algorithm approximating $\pi_t(x_t)$ is known as the particle filter. A related estimation problem is to approximate the marginal smoothing distribution $\pi_T(x_t) \triangleq p_\theta(x_t \mid y_{1:T})$ for some $t \in [0, T]$. The resulting SMC algorithms are known as particle smoothers, which usually operates by extending the forward pass in the particle filter with a backward update. This enables the algorithm to make use of both past and future observations, which reduces the problem with particle degeneracy.

The simplest particle smoother is to make use of the BPF to approximate the joint smoothing distribution $\pi_T(x_{1:T}) \triangleq p_\theta(x_{1:T} \mid y_{1:T})$. 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 bPF to estimate the fixed-lag smoothing distribution $\pi_t(x_{t-\Delta:t}) \triangleq p_\theta(x_{t-\Delta:t} \mid y_{1:t})$ for some $\Delta > 0$. In this thesis, we make frequent use of this smoother as it has a low computational cost and a reasonable accuracy.

Some of the alternatives to FL particle smoothing are based on approximations of the Bayesian smoothing recursion (Anderson and Moore, 2005) in which the aforementioned backward pass is added. Two popular examples from this family of algorithms are the forward filter backward smoother (FFBSM; Doucet et al., 2000) and the forward filtering backward simulator (FFBSI; Godsill et al., 2004). For an extensive survey of particle smoothers, see Lindsten and Schön (2013).

Example 3.3: How does unemployment affect inflation? (cont. from p. 27)

We employ a BPF from Remark 3.1 to estimate the NAIRU using data from Sweden. This corresponds to the proposal and weight function given by

$$q_t(x_t \mid x_{0:t-1}) = \mathcal{N}\Big(x_t; \phi x_{t-1} + \mu(x_{t-1}), \sigma_v^2(x_{t-1}, u_{t-1})\Big),$$
$$W_\theta(x_t, x_{0:t-1}) = \mathcal{N}\Big(y_t; y_{t-1} + \beta(u_t - x_t), \sigma_e^2\Big),$$

with $\theta = \{\phi, \alpha, \beta, \sigma_e\} = \{0.76, 0.43, 0.01, 0.28\}$ and $N = 100$ particles. The resulting estimate of the NAIRU is presented in the upper part of Figure 3.6 together with the unemployment gap (the difference between the unemployment rate and the NAIRU). We note that the unemployment gap is mostly positive during this period. The NAIRU seems to be constant at two percent and therefore the unemployment needs to decrease considerably for the inflation the increase.

Furthermore, we perform 100 Monte Carlo simulations (independent runs on the same data) to estimate the log-likelihood of the data for this model. We record the mean estimate

	$N = 50$	$N = 100$	$N = 250$	$N = 500$	$N = 1,000$
Mean	-45.90	-45.89	-45.88	-45.88	-45.89
Standard deviation	0.15	0.11	0.06	0.05	0.03

Table 3.1. The mean and variance of log-likelihood estimates in the Phillips curve model computed using the BPF while varying the number of particles N.

and its variance while varying the number of particles N between 50 and 1,000. The results are presented in Table 3.1, where we note that the mean is essentially the same and the variance decreases when N increases. This is an example of the general properties of the log-likelihood estimator discussed in Remark 3.4.

We return to this model in Example 3.6 on page 57.

Statistical properties

The analysis of SMC algorithms is rather complicated compared to standard Monte Carlo estimators as the generated particle system does not consist of independent samples due to the resampling step. However, there are many strong results regarding non-asymptotic stability and asymptotic properties, see the book long treatments by Del Moral (2013) and Del Moral (2004). Here, we only provide a short overview of the results summarised by Crisan and Doucet (2002) and Doucet and Johansen (2011).

For the asymptotic settings, it is possible to show that the empirical distribution (3.6) and the estimator in (3.7) are strongly consistent, i.e.,

$$\lim_{N\to\infty} \widehat{\pi}^N_{t,\text{SMC}} = \pi_t, \qquad \widehat{\pi}^N_{t,\text{SMC}}[\varphi] \xrightarrow{\text{a.s.}} \pi_t[\varphi], \tag{3.13}$$

where the first property holds almost surely (a.s.) and the second property holds for any integrable test function φ when $N \to \infty$. However, note that $\widehat{\pi}^N_{t,\text{SMC}}[\varphi]$ is in general biased for finite N, see Remark 3.4 for an important exception. It is also possible to derive a CLT for (3.7) given by

$$\sqrt{N}\left[\widehat{\pi}^N_{t,\text{SMC}}[\varphi] - \pi_t[\varphi]\right] \xrightarrow{\text{d}} \mathcal{N}\left(0, \sigma^2_{\text{SMC}}\right),$$

when using multinomial resampling and where σ^2_{SMC} denotes the asymptotic variance computed in Del Moral et al. (2006).

Furthermore, assume that the function $\varphi(x)$ is bounded[1] for all $x \in \mathcal{X}$ and some additional assumptions that are stated by Crisan and Doucet (2002). It then follows that the MSE of the estimator (when we make use of the bPF with multinomial resampling) can be upper bounded for any $N \geq 1$ by

$$\mathbb{E}\left[\left(\widehat{\pi}^N_{t,\text{SMC}}[\varphi] - \pi_t[\varphi]\right)^2\right] \leq C_t \frac{\|\varphi\|^2}{N}, \tag{3.14}$$

[1]This is a rather restrictive assumption as it is not satisfied by the function $\varphi(x) = x$, which is used to compute the estimate of the filtered state $\widehat{x}_{t|t}$ in Remark 3.1.

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

Note that (3.14) implies that the SMC algorithm is *stable*, i.e., that the variance of the estimator does not *blow up* as t increases for any $N \geq 1$. Intuitively, this could be a problem as the SMC algorithm makes approximations based on approximations. It turns out that the resampling step takes care of this and prevents the rapid accumulation of errors to occur. For more stability results, see Chopin (2004) and Whiteley (2013).

It is possible to relax the assumption that $\varphi(x)$ should be bounded and that we only use the BPF. The resulting upper bounds have a similar structure to (3.14) but with different functions replacing the constant C_t. It is also possible to establish uniform convergence of the estimator if the SSM is fast mixing, i.e., forgets its past fast enough, see Crisan and Doucet (2002) for details.

Remark 3.4 (Particle filtering (cont. from p. 52)). It turns out that the likelihood estimator (3.12) based on the bPF is un-biased for any $N \geq 1$, c.f. with the state estimate in (3.13). Furthermore, under some mixing assumptions for the SSM, the error of the estimate satisfies a CLT given by

$$\sqrt{N}\left[p_\theta(y_{1:T}) - \widehat{p}_\theta^N(y_{1:T})\right] \xrightarrow{\mathrm{d}} \mathcal{N}\left(0, \sigma_L^2\right), \tag{3.15}$$

for some asymptotic variance σ_L^2, see Proposition 9.4.1 in Del Moral (2004) or Pitt et al. (2012). Note that the estimator for the log-likelihood is biased for a finite N, but strongly consistent and asymptotically Gaussian. This follows from the second-order delta method (Casella and Berger, 2001).

Estimating additive functionals

In this section, we consider the use of SMC algorithms to estimate the expected value of an *additive functional*. This type of functional can be expressed as

$$S_\theta(x_{0:T}) = \sum_{t=1}^{T} \xi_{\theta,t}(x_{t-1:t}), \tag{3.16}$$

which means that a function that depends on the entire state trajectory can be decomposed into a sum of functionals. Here, $\xi_{\theta,t}(x_{t-1:t})$ denotes some general functional that depends on only two states of the trajectory. This type of additive functional occurs frequently in SSMs when computing functions that depend on the densities $f_\theta(x_{t+1} \mid x_t)$ and $g_\theta(y_t \mid x_y)$ due to the Markov property. The resulting expectation can be expressed by

$$\pi_T[S_\theta] = \sum_{t=1}^{T} \int_{\mathcal{X}^2} \xi_{\theta,t}(x_{t-1:t}) p_\theta(x_{t-1:t} \mid y_{1:T}) \, \mathrm{d}x_{t-1:t}, \tag{3.17}$$

where $p_\theta(x_{t-1:t} \mid y_{1:T})$ denotes the two-step smoothing distribution, which is analytically intractable for a general SSM. However, it can be estimated by a particle filtering or smoothing, see Remark 3.2 and Poyiadjis et al. (2011). In Remark 3.5, we show how to make use of (3.17) to estimate the score function and observed information matrix for an SSM. The same setup can also be used in the expectation maximisation (EM; Dempster et al., 1977; McLachlan and Krishnan, 2008) algorithm as discussed by Del Moral et al. (2010).

Remark 3.5 (Estimating the score function and information matrix for an SSM). The score function (2.15) for an SSM can be estimated using the *Fisher identity* (Cappé et al., 2005) given by

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

where $\log p_\theta(x_{0:T}, y_{1:T})$ denotes the complete data log-likelihood given by

$$\log p_\theta(x_{0:T}, y_{1:T}) = \log \mu(x_0) + \sum_{t=1}^{T} \Big[\log f_\theta(x_t \,|\, x_{t-1}) + \log g_\theta(y_t \,|\, x_t)\Big]. \tag{3.18}$$

This results in the additive functional

$$\xi_{\theta',t}(x_{t-1:t}) = \nabla \log f_\theta(x_t \,|\, x_{t-1})\big|_{\theta=\theta'} + \nabla \log g_\theta(y_t \,|\, x_t)\big|_{\theta=\theta'}, \tag{3.19}$$

corresponding to part of the gradient of (3.18) evaluated at $\theta = \theta'$. The observed information matrix (2.16) can be estimated using the *Louis identity* (Cappé et al., 2005) given by

$$\mathcal{J}(\theta') = \left[\mathcal{S}(\theta')\right]^2 - \left[\nabla^2 p_\theta(y_{1:T})\big|_{\theta=\theta'}\right]\left[p_\theta(y_{1:T})\right]^{-1}, \tag{3.20}$$

where the second term can be expressed as

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

Note that the first term in (3.20) is the square of the score function, which can be estimated by (3.19). The term $\mathbb{E}[\nabla^2 \log p_\theta(x_{0:T}, y_{1:T}) \,|\, y_{1:T}]$ can be estimated using an analogue additive functional. However, this cannot be done for the remaining term as it consists of terms with the structure $\xi_{\theta,t}(x_{t-1:t})\xi_{\theta,s}(x_{s-1:s})$ for all $s, t \in \{1, \ldots, T\}$. We return to this problem and how to estimate these two identities in Section 3 (page 162) of Paper B.

Example 3.6: How does unemployment affect inflation? (cont. from p. 54)

From the previous, we know how to estimate the log-likelihood and the gradient of the log-posterior. It is possible to make use of this information in a gradient ascent or quasi-Newton algorithm for parameter inference by maximising the log-likelihood/posterior. Unfortunately, this can be difficult as the log-likelihood and the gradient can be quite noisy.

In Figure 3.7, we present estimates of these two quantities obtained using a BPF with $N = 50$ particles while varying ϕ. The gradient of the log-posterior is estimated using a FL particle smoother, which makes use of the particle system generated by the BPF, see Remark 3.2. We note that the log-likelihood estimate is quite noisy for this model and the optimum around 0.75 is indistinguishable from the surroundings. However, the gradient estimate is less noisy and it is zero at 0.81, which indicates the MAP estimate of ϕ.

Here, it could be possible to make use of a direct optimisation method for estimating ϕ based on the gradient estimate, see Kok et al. (2015) and Poyiadjis et al. (2011). In other models, the noise can be even larger and increasing N could make inference computationally prohibitive. We return to discuss other remedies for this problem in Chapter 4.

We return to this model in Example 3.13 on page 70.

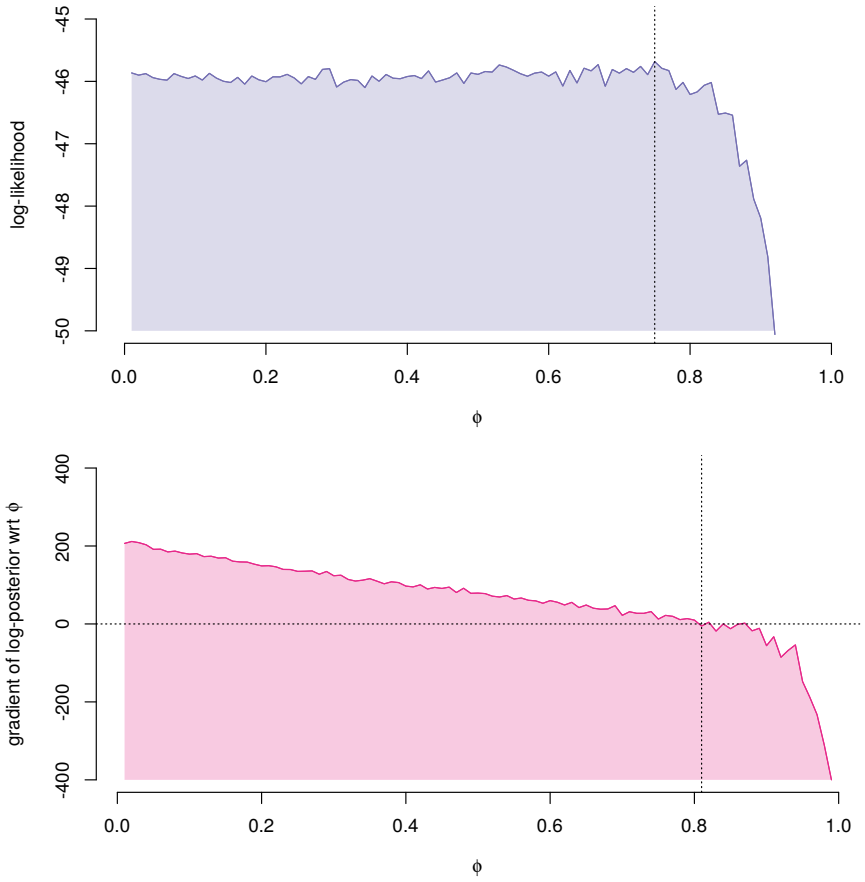


Figure 3.7. The estimates of the log-likelihood (upper) and the gradient of $\log p(\theta|y)$ with respect to ϕ (lower) in the Phillips curve model computed by the BPF and FL particle smoothing when varying ϕ. The dotted vertical lines indicate the maximum likelihood estimate (upper) and MAP estimate (lower) and the dotted horizontal line indicate the value zero for the gradient of the log-posterior.

Markov chain Monte Carlo

Another approach for sampling from complicated target distributions is to make use of MCMC algorithms. The main idea is to construct a Markov chain with the target as its stationary distribution. An advantage of MCMC compared to e.g. importance sampling is that it can be easier to find a good proposal distribution for an MCMC algorithm. The reason for this is that it is possible to use the current state of the Markov chain to make a more informed proposal for the next state.

To explain what this means, we need to introduce some essential concepts of Markov chains. More general introductions to the subject are found in Meyn and Tweedie (2009) and Billingsley (2012). Introductions to Markov chains for MCMC are available in Tierney (1994) and Robert and Casella (2004).

A sequence of random variables $\{x_k\}_{k=0}^K$ is a Markov chain if

$$\mathbb{P}[x_k \in A \,|\, x_{0:k-1}] = \mathbb{P}[x_k \in A \,|\, x_{k-1}] = \int_A R(x_{k-1}, \mathrm{d}x_k) = \int_A R(x_{k-1}, x_k)\,\mathrm{d}x_k,$$

where $x_k \in \mathcal{X}$ denotes the state of the Markov chain at time k and A denotes some (measurable) subset of $\mathcal{X}$. Here, the Markov kernel $R : \mathcal{X} \times \mathcal{X} \to [0, 1]$ assigns a probability to any (measurable) subset given the current state x_{k-1}.

In this thesis, we assume that $R(x_{k-1}, \mathrm{d}x_k)$ admits a density denoted $R(x_{k-1}, x_k)$ with respect to the Lebesgue measure. However, all results presented in the following carries over to when the Markov kernel is not absolutely continuous, e.g., when the kernel is a combination of a density and a Dirac distribution.

An important property of the Markov chain is the concept of an *invariant distribution*. We say that a distribution μ is invariant to the Markov chain if

$$\mu(x_k) = \int_{\mathcal{X}} \mu(x_{k-1}) R(x_{k-1}, x_k)\,\mathrm{d}x_{k-1}, \tag{3.21}$$

which we write in short as $\mu = \mu R$. The concept of invariance means that $x_k \sim \mu$ for all future values of k if $x_{k-1} \sim \mu$. Hence if the above holds for any k, then the Markov chain has entered its stationary regime. Note that, we make use of R as an operator or transform in (3.21). Hence, we can see Markov kernel as a mapping between the distribution of x_{k-1} and the distribution of x_k.

The invariance property is important as we are interested in sampling from a target distribution using a Markov chain with a specific stationary distribution. The remaining question is how to construct R such that the stationary distribution matches the target π. It turns out that a sufficient condition for the chain to have a stationary distribution μ is

$$\mu(x_{k-1}) R(x_{k-1}, x_k) = \mu(x_k) R(x_k, x_{k-1}), \quad \text{for any } x_{k-1}, x_k \in \mathcal{X}. \tag{3.22}$$

This condition is known as *detailed balance* and it ensures that the Markov chain is reversible. That is, the statistics of the Markov chain are the same if the direction of time is reversed. To confirm this, we can integrate both sides to recover the invariance property in (3.21),

i.e.,

$$\begin{aligned}\int_{\mathcal{X}} \mu(x_{k-1})R(x_{k-1}, x_k)\,\mathrm{d}x_{k-1} &= \int_{\mathcal{X}} \mu(x_k)R(x_k, x_{k-1})\,\mathrm{d}x_{k-1} \\ &= \mu(x_k)\int_{\mathcal{X}} R(x_k, x_{k-1})\,\mathrm{d}x_{k-1} \\ &= \mu(x_k),\end{aligned}$$

where the invariance property $\mu R = \mu$ is recovered. Here, the detailed balance (3.22) gives the first step and the property $\int_{\mathcal{X}} R(x_k, x_{k-1})\,\mathrm{d}x_{k-1} = 1$ is used in the third step.

Another important concept in Markov chain theory is *ergodicity*, which is required later to establish certain statistical properties of MCMC algorithms. Intuitively, an ergodic Markov chain can visit any (interesting) part of the state space at any point in time. This is necessary as we would like to make use of the Markov chain to sample the target. Hence, it must be able to visit all parts of the target with a non-zero probability mass to be able to explore it fully. We might end up with problems if the Markov chain is not ergodic as it then can get stuck in certain parts of the state space and therefore not revisit all areas of the target with some regularity.

It is possible to show that if the Markov chain is ergodic, then it also has a unique invariant distribution satisfying

$$\mu = \lim_{k\to\infty} \mu_0 R^k,$$

for (almost) any initial distribution μ_0. Hence, we have a unique limiting distribution and it is the stationary or invariant distribution of the Markov chain governed by the kernel R. The requirements for ergodicity are *irreducibility*, *aperiodicity* and *positive recurrence*. We say that a chain is (strongly) irreducible if

$$\int_A R(x_{k-1}, x_k)\,\mathrm{d}x_k > 0,$$

for any $x_{k-1}, x_k \in \mathcal{X}$. Hence, the Markov chain can reach any part of the state space in one step. Aperiodicity roughly means that the Markov chain does not get stuck in cycles where it returns in a number of steps with a certain period. Finally, a Markov chain is positive recurrent if the expected number of visits to any subset of the state space is infinite. The opposite case is known as transience, where the expected number of returns to a certain set is zero after $M < \infty$ steps. We return to what these properties mean in practice when introducing some specific examples of MCMC algorithms.

The remaining question is how to construct the Markov kernel R in practice such that it has some sought target distribution π as its stationary distribution. That is, a ergodic kernel R that fulfils detailed balance for π. We present two different approaches: the MH algorithm and Gibbs sampling. After this, we return to discussing how to make use of the Markov chain generated by these approaches to estimate expectations of test functions with respect to some target.

Metropolis-Hastings

We can construct a Markov chain to sample from some target distribution $\pi(x) = \gamma(x)\mathcal{Z}^{-1}$ using the MH algorithm (Metropolis et al., 1953; Hastings, 1970). Here, we remind the reader that γ denotes the un-normalised target distribution and $\mathcal{Z}$ denotes the often unknown normalisation constant.

Remark 3.7 (Bayesian parameter inference using the MH algorithm). In many applications, we have the parameters θ as the state and the target as the parameter posterior $\pi(\theta) = p(\theta|y)$ given by Bayes' theorem (2.10). In this setting, the normalisation constant is the marginal likelihood $p(y)$. Furthermore, we have the un-normalised target $\gamma(\theta) = p(y|\theta)p(\theta)$, where $p(y|\theta)$ and $p(\theta)$ denote the likelihood and the prior distribution, respectively.

The MH algorithm consists of an iterative scheme in which we propose a new state of the Markov chain called the candidate state x' from some proposal distribution $q(x'|x_{k-1})$ admitting a density with respect to the Lebesgue measure. The candidate state is then accepted or rejected according to some acceptance probability. In summary, we carry out the following operations during iteration k:

a) Sample the proposal $x' \sim q(x'|x_{k-1})$.

b) Set the next state

$$x_k = \begin{cases} x' & \text{with probability } \alpha(x_{k-1}, x'), \\ x_{k-1} & \text{with probability } 1 - \alpha(x_{k-1}, x'). \end{cases}$$

These steps are repeated until we obtain K samples from π denoted by $\{x_k\}_{k=1}^K$. The *acceptance probability* is given by

$$\alpha(x_{k-1}, x') = 1 \wedge \frac{\pi(x')}{\pi(x_{k-1})} \frac{q(x_{k-1}|x')}{q(x'|x_{k-1})} = 1 \wedge \frac{\gamma(x')}{\gamma(x_{k-1})} \frac{q(x_{k-1}|x')}{q(x'|x_{k-1})}, \tag{3.23}$$

where $a \wedge b \triangleq \min(a, b)$ and when the proposal can be expressed as a density. Note that the normalisation constant $\mathcal{Z}$ cancels and only point-wise evaluation of the un-normalised target is required for computing the acceptance probability.

Typically, the Markov chain is constructed by the MH algorithm such that it explores the posterior distribution by local moves thus exploiting the previously accepted state. Hence, it focuses its attention to areas of the state space in which the posterior assigns a relatively large probability mass. We can see this from (3.23), a proposed state x' is always accepted (neglecting the influence of the proposal) if it results in larger value of the target compare with x. Furthermore, we accept a proposed state 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 Markov chain to find and for it to explore areas of high posterior probability. Hence, the MH algorithm can possibly escape local extrema if the target is multi-modal which is a problem for many local optimisation algorithms used in numerical maximum likelihood inference. It is also easier to construct a good proposal for a high-dimensional target using local moves as in the MH algorithm compared with the importance sampling algorithm.

In the MH algorithm, we can express the Markov kernel by

$$R(x_{k-1}, \mathrm{d}x_k) = \alpha(x_{k-1}, x_k) q(x_k \,|\, x_{k-1}) \,\mathrm{d}x_k + \left[1 - \int_{\mathcal{X}} \alpha(x_{k-1}, z) q(z \,|\, x_{k-1}) \,\mathrm{d}z \right] \delta_{x_{k-1}}(\mathrm{d}x_k), \tag{3.24}$$

which cannot be expressed as a density due to the fact that the Dirac distribution is not absolutely continuous. Hence, we write the kernel and the proposal as measures in this case. This kernel satisfies the detail balance condition (3.22), which can be verified by

$$\begin{aligned} \pi(x_{k-1}) q(x_k \,|\, x_{k-1}) \alpha(x_{k-1}, x_k) &= \pi(x_{k-1}) q(x_k \,|\, x_{k-1}) \left[1 \wedge \frac{\pi(x_k)}{\pi(x_{k-1})} \frac{q(x_{k-1} \,|\, x_k)}{q(x_k \,|\, x_{k-1})} \right], \\ &= \pi(x_{k-1}) q(x_k \,|\, x_{k-1}) \wedge \pi(x_k) q(x_{k-1} \,|\, x_k), \\ &= \pi(x_k) q(x_{k-1} \,|\, x_k) \alpha(x_k, x_{k-1}), \end{aligned}$$

by using the definition of the acceptance probability (3.23). This results in that the target distribution is the stationary distribution of the Markov chain. Finally, the Markov chain generated by the MH algorithm is ergodic if the proposal $q(x' \,|\, x) > 0$ for all $x', x \in \mathcal{X}$. That is, the probability of reaching any set of the state space in one or a few steps is non-zero. Further details are provided by Tierney (1994) and Robert and Casella (2004).

The performance of the MH algorithm is dependent on the choice of the proposal and the resulting acceptance rate. Two common choices for proposals are the independent Gaussian proposal and Gaussian random walk given by

$$q(x' \,|\, x) = q(x') = \mathcal{N}(x'; \mu, \Sigma), \qquad q(x' \,|\, x) = \mathcal{N}(x'; x, \Sigma),$$

for some mean vector μ and covariance matrix Σ. Note that the independent proposal cannot exploit the previous accepted state, which can be problematic if the proposal does not match the target well. If the match is good, it is always possible to instead use an importance sampling algorithm, which is probably the better choice.

The Gaussian random walk is the typical choice in applications but a version based on the Student's t distribution is also common. The choice of Σ determines the performance of the MH algorithm. If it is too small, the algorithm tends to accept proposed steps but as each step is small this results in a large autocorrelation in the chain. Moreover, the probability of accepting a large step is also low, which results in a large autocorrelation as well. In the MCMC literature, this is referred to bad mixing of the Markov chain, see Figure 5 (page 136) in Paper A for an illustration of good and bad mixing. A good choice of the proposal is therefore crucial to obtain a small autocorrelation in the Markov chain and (as we shall see) accurate estimates of the target.

Gibbs sampling

Another approach to construct a Markov kernel to sample the target distribution of interest is to use Gibbs sampling (Geman and Geman, 1984). A major difference with the MH algorithm is that all the draws from the proposal distribution are accepted by the algorithm. However, this does not mean that Gibbs sampling is superior to other types of MCMC

algorithms. It is actually possible to show that Gibbs sampling is a particular special case of the MH algorithm, see Robert and Casella (2004, p. 381). Gibbs sampling can only be implemented for some models that admit conditional distributions with a special structure. It can also get stuck if some states are highly correlated with each other. The main advantage is that no tuning is required by the user which make implementation easier.

In Gibbs sampling, we sample the full conditional distribution for each element of the state vector while keeping the other elements fixed to their current value. Hence, we repeat the following step during iteration k:

$$x_{k,i} \sim \pi_i\big(x_{ki} \,|\, x_{k,1}, \ldots, x_{k,i-1}, x_{k-1,i+1}, \ldots, x_{k-1,p}\big),$$

for each element of the state $i = 1, \ldots, p$. Here, we denote the full condition distribution for x_i of the target distribution $\pi(x)$ as $\pi_i(x_i \,|\, \cdot)$. The resulting Markov kernel is given by

$$R(x_{k-1}, x_k) = \prod_{i=1}^{p} \pi_i(x_{k,i} \,|\, x_{k,1}, \ldots, x_{k,i-1}, x_{k-1,i+1}, \ldots, x_{k-1,p}). \tag{3.25}$$

It is possible to show (Robert and Casella, 2004, p. 345) that $\pi(x) = \pi(x_1, x_2, \ldots, x_p)$ is the invariant distribution of the Markov chain governed by (3.25). Furthermore, the kernel generates an ergodic chain if the density $\pi(x)$ satisfies the positivity condition. That is, if $\pi(x) > 0$ for all x_i such that $\pi_i(x_i) > 0$ for $i = 1, 2, \ldots, p$. Here, we denote the marginal density of x_i under $\pi(x)$ by $\pi_i(x_i)$. In summary, we have that the Gibbs sampler generates samples from the target distribution in its stationary regime.

Compared with the MH algorithm, the full conditionals depend on the problem at hand and Gibbs sampling cannot be used for all models. It is common to make use of conjugate priors to obtain some posterior $\pi(x)$ with the necessary conditionals. In Example 3.8, we show how to apply Gibbs sampling for inference in the model regarding the ideological leanings of US Supreme Court justices.

In practice, *blocking* and *partial collapsing* are used to increase the performance. Blocking means that we sample more than one parameter at the same time, e.g.,

$$x_{k,1}, x_{k,2} \sim \pi_{1,2}(x_{k,1}, x_{k,2} \,|\, x_{k-1,3}), \qquad x_{k,3} \sim \pi_3(x_{k,3} \,|\, x_{k,1}, x_{k,2}).$$

An example of partial collapsing is the update

$$x_{k,1} \sim \pi_1(x_{k,1} \,|\, x_{k-1,2}), \qquad x_{k,2} \sim \pi_2(x_{k,2} \,|\, x_{k,1}), \qquad x_{k,3} \sim \pi_3(x_{k,3} \,|\, x_{k,1}, x_{k,2}).$$

Both approaches can result in a significant increase in performance. However, the design is problem specific and care needs to be taken to ensure that the sampler converge to the desired target.

Example 3.8: Voting behaviour in the US Supreme court (cont. from p. 31)

We follow Martin et al. (2011) and assume the following priors

$$\alpha_t \sim \mathcal{N}(\alpha_t; 0_2, 0.25\mathbf{I}_2), \qquad x_i \sim \mathcal{N}(x_i; 0, 1),$$

for each justice $i = 1, \ldots, n$ and case $t = 1, \ldots, T$. From these choices of prior, we can compute the conditional posteriors as presented in Example 2.6, see Albert (1992) for the

complete derivation. The resulting update for the Gibbs sampler is given by

$$
\begin{aligned}
u'_{it} &\sim \begin{cases} \text{TN}_{[0,\infty)}(u'_{it}; -\alpha_{t,1} + \alpha_{t,2} x_i, 1) & \text{if } y_{it} = 1, \\ \text{TN}_{(-\infty,0]}(u'_{it}; -\alpha_{t,1} + \alpha_{t,2} x_i, 1) & \text{if } y_{it} = 0, \end{cases} \\
\alpha'_t &\sim \mathcal{N}(\alpha'_t; m_{\alpha,t}, (X^\top X)^{-1}), \\
x'_i &\sim \mathcal{N}(x'_i; m_{x,i}, \sigma_x^{-2}),
\end{aligned}
$$

for each justice $i = 1, \ldots, n$ and case $t = 1, \ldots, T$.

We introduce the notation $X \triangleq [-1 \quad x_{1:n}]^\top$ for the design matrix. Furthermore, we use $u_i = \{u_{it}\}_{t=1}^T$ and introduce $\text{TN}_{(a,b)}(\mu, \sigma^2)$ to denote the truncated Gaussian distribution on the interval (a, b) with location $\mu \in \mathbb{R}$ and scale $\sigma > 0$. Finally, we introduce the following auxiliary quantities

$$
m_{\alpha,t} = (X^\top X)^{-1} X u_t, \quad m_{x,i} = \sigma_x^{-2} \sum_{t=1}^T \alpha'_{t,2}(u_{it} + \alpha'_{t,1}), \quad \sigma_x^2 = 1 + \sum_{t=1}^T (\alpha'_{t,2})^2,
$$

where $m_{\alpha,t}$ is the same as the ordinary LS estimate (2.3).

We make use of the command `MCMCirt1d` implemented in the R package `MCMCpack` (Martin et al., 2011) for the inference. The Gibbs sampler is run for $K = 50,000$ iterations (discarding the first $10,000$ as burn-in). In Figure 3.8, we present the resulting marginal posteriors for $x_{1:9}$. From this, we note that: (i) Ginsberg, Sotomayor, Breyer and Kagan are more liberal and (ii) Scalia, Thomas and Altio are more conservative. This is the same result as we predicted in Example 2.2 when first introducing the data set.

We return to this model in Example 5.2 on page 88.

Statistical properties

In this section, we summarise some of the statistical results that MCMC algorithms relies upon and briefly mention their underlying assumptions. For more information about the properties of MCMC algorithms in general, see e.g., Tierney (1994), Robert and Casella (2004) and Meyn and Tweedie (2009). A natural estimator of $\pi[\varphi]$ for any integrable test function φ using the Markov chain generated by a MCMC algorithm is given by

$$
\widehat{\pi}^K_{\text{MCMC}}[\varphi] = \frac{1}{K} \sum_{k=1}^K \varphi(x_k), \tag{3.26}
$$

where x_k denotes the state of the Markov chain at time step k with π as its stationary distribution. Note that this intuitively means that time spent by the Markov chain in particular region is proportional to the probability mass allocated to the region. Hence, we can map the distribution of probability mass in a target distribution but observing how the Markov chain spends its time in the state space by e.g., a histogram.

A practical problem is that the estimator (3.26) only holds for samples from the Markov chain when it is in stationarity. That is, when the distribution of the Markov chain is the limiting or stationary distribution, which usually is the target distribution of interest. In

practice, it is difficult to assert when this occurs and often the Markov chain does not reach stationarity for many iteration.

However, it is common to run the Markov chain during a *burn-in* (or *warm-up*) phase and discard the samples from this period. After the *burn-in*, we assume that all the samples are from the target distribution. Note that there are a number of convergence tests and similar to make use of for diagnostics. More information is found in Section 6 (page 142) of Paper A and in Robert and Casella (2009, p. 242).

By the *ergodic theorem* (Tierney, 1994; Robert and Casella, 2004), we know that the estimator (3.26) is *strongly consistent*, i.e.,

$$\widehat{\pi}^K_{\text{MCMC}}[\varphi] \xrightarrow{\text{a.s.}} \pi[\varphi],$$

when $K \to \infty$. Note that this property does not follow directly from the SLLN as the samples obtained for the target are not IID, due to the fact that the states of the Markov chain are correlated.

Furthermore, it is possible to form a CLT for the estimator with some additional assumptions, see Jones (2004). Usually, we assume that the Markov chain is *uniformly ergodic*, i.e.,

$$\|R^k(x_0, \cdot) - \pi\|_{\text{TV}} < C\rho^{-k},$$

for some $C < \infty$, $\rho > 1$ and any $x_0 \in \mathcal{X}$. Here, $\|\cdot\|_{\text{TV}}$ denotes the total variational (TV) norm given by

$$\|\mu - \pi\|_{\text{TV}} \triangleq \frac{1}{2} \int_{\mathcal{X}} |\mu(x) - \pi(x)| \, \mathrm{d}x,$$

when both densities are dominated by the Lebesgue measure. This means that for any starting point x_0, we converge to the stationary distribution with a geometric rate. This is a stronger condition compared with geometric convergence, where the same convergence holds for most $x_0 \in \mathcal{X}$. Given uniform/geometric convergence, we can find a CLT given by

$$\sqrt{K}\left[\widehat{\pi}^K_{\text{MCMC}}[\varphi] - \pi[\varphi]\right] \xrightarrow{\text{d}} \mathcal{N}(0, \sigma^2_{\text{MCMC}}),$$

when $K \to \infty$. Here, σ^2_{MCMC} denotes the variance of the estimator given by

$$\sigma^2_{\text{MCMC}} = \mathbb{V}\left[\widehat{\pi}^K_{\text{MCMC}}[\varphi]\right] = \pi\left[\widetilde{\varphi}^2(x_0)\right] + 2\sum_{k=1}^{\infty} \pi\left[\widetilde{\varphi}(x_0)\widetilde{\varphi}(x_k)\right],$$

where we introduce $\widetilde{\varphi}(x) \triangleq \varphi(x) - \pi[\varphi(x)]$ for brevity. We can rewrite this as

$$\sigma^2_{\text{MCMC}} = \pi\left[\widetilde{\varphi}^2(x_0)\right] \cdot \text{IACT}(x_{1:K}),$$

where the *integrated autocorrelation time* (IACT) is given by

$$\text{IACT}(x_{1:K}) \triangleq 1 + 2\sum_{k=1}^{\infty} \rho_k(x_{1:K}).$$

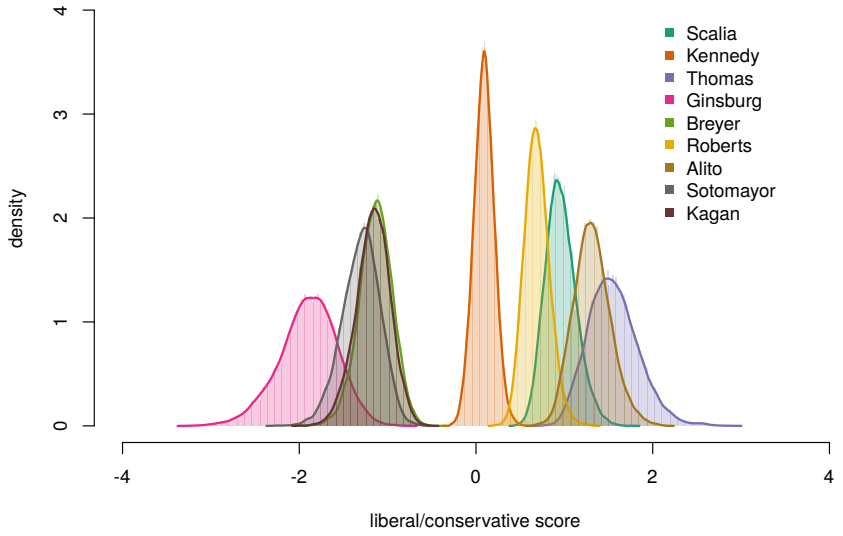


Figure 3.8. The estimates of marginal posteriors for $x_{1:9}$ in the IRM for the supreme court justice data, which represent the liberal/conservative score for each justice.

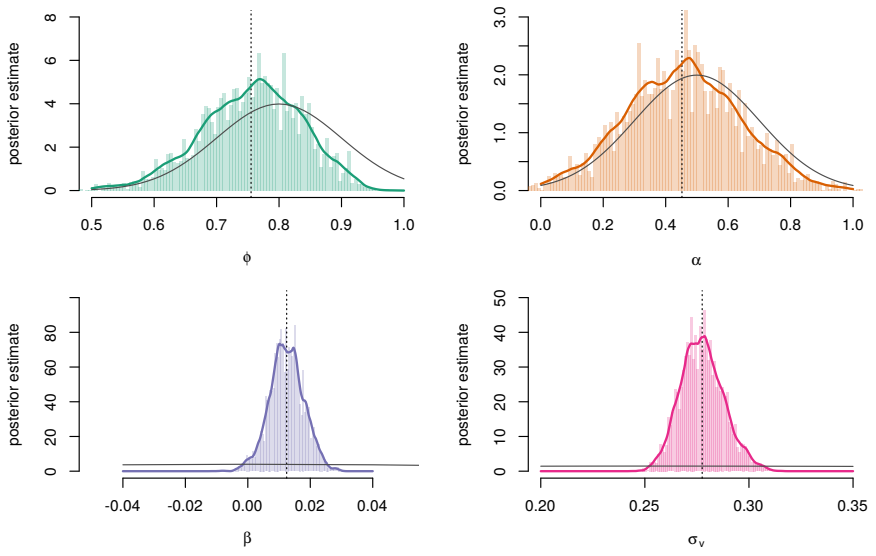


Figure 3.9. Estimates of the parameter posterior distribution for ϕ (green), α (orange), β (purple) and σ_e (magenta) in the Phillips curve model using the Swedish data. The prior distributions are indicated by grey cures and the posterior means by dotted lines.

Here, $\rho_k(x)$ denotes the k-lag autocorrelation function (ACF) of $\varphi(x_k)$. In practice, we have to estimate the IACT by approximating it using the sample estimates, see Section 6.3 (page 142) in Paper A for more information.

The IACT can be interpreted as the number of iterations between each independent sample and it is connected with the mixing property. Hence, a small value of the IACT means that the Markov chain mixes well, nearby samples are almost uncorrelated and that the asymptotic variance is small. Note that an importance sampling algorithm makes use of independently proposed particles and its IACT is therefore one. However, the non-equal weights in the importance sampling estimator results in other inefficiencies.

Remark 3.9 (Optimal Gaussian random walk proposal for the MH algorithm). It is possible to compute the optimal Gaussian random walk proposal for a Gaussian target distribution by analytically minimising the IACT. The resulting proposal (Roberts et al., 1997) is given by

$$q(x \mid x') = \mathcal{N}\left(x'; x, \frac{2.38^2}{p}\widehat{\Sigma}\right), \tag{3.27}$$

where $\widehat{\Sigma}$ denotes an estimate of the posterior covariance and p denotes the number of parameters in the model. The covariance is often unknown but it can be estimated using pilot runs. However, this is problematic as its estimation essentially means that we already have a Markov chain with good mixing.

It is also possible to calculate the resulting acceptance probability, which is 0.234 in the case of a Gaussian target when using the optimal proposal. The performance of (3.27) can be poor if the target deviates from a Gaussian distribution, e.g., if the target is more heavy-tailed than a Gaussian or is multi-modal. In this case, it is possible to find better proposals. However by the Bernstein-von-Mises theorem from Section 2.3, the target concentrates asymptotically to a Gaussian as the amount of data increases and then the Gaussian assumption is valid.

Pseudo-marginal Metropolis-Hastings

In Section 3.2.3, we illustrated how to make use of Markov chains to sample from some complicated target distribution denoted by $\pi(x)$. In the MH algorithm, it is required that we can evaluate $\pi(x)$ or its un-normalised version $\gamma(x)$ point-wise in the computation of the acceptance probability (3.23). However, we cannot implement the MH algorithm when this is not possible.

Instead, we can often make use of importance sampling or SMC to obtain un-biased but noisy estimates of the target. These estimators are exploited by Beaumont (2003) to approximate the acceptance probability in the MH algorithm by replacing the unknown target distribution by a non-negative and un-biased estimate. We refer to the resulting algorithm as a pseudo-marginal MH (PMMH) algorithm. It is proved by Andrieu and Roberts (2009) that this is a valid approach and that the PMMH algorithm has similar convergence properties as its exact counterpart. For example, the distribution of the Markov chain convergences to the sought target distribution.

Note that the assumption of non-negativity is important in what follows to obtain a so-called *exact approximation* of the MH algorithm. That is, an algorithm that returns samples

from the sought posterior. As previously mentioned, SMC algorithms can provide one such valid estimator but finding such estimators is difficult in general. Assume that we have an un-biased estimator of some quantity $\alpha \in \mathbb{R}$, then there does not exist any algorithm that can provide an un-biased estimator of $h(\alpha) \in \mathbb{R}^+$ for some non-constant function $h : \mathbb{R} \to \mathbb{R}^+$. This fundamental result was recently established by Jacob and Thiery (2015). The implications of this result is that exact approximation of algorithms can be difficult using approximate Bayesian computations (ABC). As a consequence, we might have to scarify un-biasedness to obtain inference algorithms with computational tractability. We return to discussing this problem in Section 4.2.

Remark 3.10 (Bayesian parameter inference in non-linear SSMs). From Chapter 2, we know that the likelihood is intractable for a non-linear SSM. Therefore, we cannot evaluate the target distribution or its un-normalised version as $p(y \mid \theta)$ is unknown. The key point with PMMH is that the un-biased estimate from the bPF discussed in Remark 3.1 can be used as a plug-in estimator for $p(y \mid \theta)$.

One approach to show the validity of the PMMH algorithm is to consider it to be a standard MH algorithm targeting an extended target distribution. To see this, we assume that there exists a non-negative and un-biased estimator of the un-normalised target $\gamma(x)$ such that

$$\mathbb{E}_u\left[\widehat{\gamma}^N(x \mid u)\right] = \int_{\mathcal{U}} \widehat{\gamma}^N(x \mid u)\, m(u)\, \mathrm{d}u = \gamma(x), \tag{3.28}$$

where $u \in \mathcal{U}$ denotes the multivariate random samples with density $m(u)$ with respect to the Lebesgue measure used to construct this estimator. Remember that the normalisation constant $\mathcal{Z}$ cancels in the acceptance probability for the MH algorithm. Hence, it does not matter that we can only estimate the value of the un-normalised target in this setting.

Remark 3.11 (The random samples u). In the PMMH algorithm, the variable u contains the random variables used to estimate the target. These corresponds to samples from the proposal distribution when an importance sampler is applied to estimate the target. In the SMC algorithm, the random variables are used in the resampling step and for propagation. Hence, we can see a SMC algorithm such as the BPF algorithm as a deterministic algorithm given u. In this case, we refer to the resulting algorithm as the particle MH (PMH) algorithm. It was first introduced by Fernández-Villaverde and Rubio-Ramírez (2007) and later analysed by Andrieu et al. (2010).

Furthermore, we assume a proposal distribution for x and u such that

$$q(x', u' \mid x, u) = q_x(x' \mid u, x)\, q_u(u' \mid u). \tag{3.29}$$

Note that the random variables u can be used when proposing the candidate state x'. We will find this useful in Section 4, when we propose extensions of the PMMH algorithm to improve mixing.

Finally, we can introduce the MH algorithm operating on the extended space $\mathcal{X} \times \mathcal{U}$ with the *extended target* given by

$$\gamma(x, u) = \widehat{\gamma}^N(x \mid u)\, m(u). \tag{3.30}$$

As a result, we can recover the original target by the marginalisation given by the un-biasedness property in (3.28). Note that the resulting MH algorithm with the target (3.30)

and proposal (3.29) has an acceptance probability given by

$$\alpha(x, u, x', u') = 1 \wedge \frac{\widehat{\gamma}^N(x' \mid u')}{\widehat{\gamma}^N(x \mid u)} \frac{q(x, u \mid x', u')}{q(x', u' \mid x, u)}. \tag{3.31}$$

This acceptance probability is the same as for the MH algorithm but replacing the target with an un-biased estimator of it and including an *extended proposal*. Note that this is not a formal proof of the validity and the interested reader is referred to Andrieu and Roberts (2009) for a more detailed presentation.

To recapitulate, the PMMH algorithm consists of an iterative scheme in which we propose x' and u' using (3.29) and accept with the probability given by (3.31). Hence, we carry out the following operations during iteration k:

a) Sample the state proposal $x' \sim q_x(x' \mid x_{k-1}, u_{k-1})$.

b) Sample the auxiliary variable proposal $u' \sim q_u(u' \mid u_{k-1})$.

c) Compute the estimate of the un-normalised target $\widehat{\gamma}^N(x' \mid u')$.

d) Set the next state

$$\{x_k, u_k\} = \begin{cases} \{x', u'\} & \text{with probability } \alpha(x_{k-1}, u_{k-1}, x', u'), \\ \{x_{k-1}, u_{k-1}\} & \text{with probability } 1 - \alpha(x_{k-1}, u_{k-1}, x', u'). \end{cases}$$

These steps are repeated until we obtain K samples from $\pi(x, u)$ denoted by $\{x_k, u_k\}_{k=1}^K$.

The statistical properties of the PMMH algorithms are similar to the MH algorithm. However, the performance of PMMH often depends on the number of samples N, which are used to compute point-wise estimates of the un-normalised target. If N is too small, then the variance of the 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 to the computational cost of estimating the target.

Remark 3.12 (Optimal Gaussian random walk proposals and selecting N in the PMH algorithm). There is a trade-off between the number of PMMH/PMH iterations and the computational complexity in the estimator of the target. The overall aim is to minimise the total computational cost of the algorithm. This problem is analysed for the PMH algorithm by Doucet et al. (2015) and Pitt et al. (2012). Their conclusion is to select N such that the variance of the target estimates is between 1.0 and 1.7 depending on the efficiency of the proposal for θ.

Furthermore, it is possible to construct optimal Gaussian random walk proposals (minimising the IACT for a Gaussian target) for the PMH algorithm in analogue with the MH algorithm. This is investigated by Sherlock et al. (2015) and the resulting proposal is given by

$$q(x' \mid x) = \mathcal{N}\left(x'; x, \frac{2.562^2}{p}\widehat{\Sigma}\right), \tag{3.32}$$

where $\widehat{\Sigma}$ again denotes an estimate of the posterior covariance and p denotes the dimension of the parameter vector. The resulting acceptance probability is 0.07 in the case of a Gaussian target when using the optimal proposal. Compared with the MH algorithm, this acceptance rate is much smaller and this is due to the noise in the likelihood estimator.

Example 3.13: How does unemployment affect inflation? (cont. from p. 57)

We are now interested in estimating the parameter posterior $\pi(\theta)$ where $\theta = \{\phi, \alpha, \beta, \sigma_e\}$. To do this, we need to augment the model with prior distributions for each parameters, where we choose

$$\phi \sim \text{TN}_{(-1,1)}(\phi; 0.8, 0.1^2), \qquad \alpha \sim \mathcal{N}(\alpha; 0.5, 0.2^2),$$
$$\beta \sim \mathcal{N}(\beta; 0, 0.1^2), \qquad \sigma_e \sim \mathcal{G}(\sigma_e; 2, 4),$$

where $\mathcal{G}(a, b)$ denote the Gamma distribution with expected value a/b.

To sample from the posterior, we employ the PMH algorithm with the bPF to provide estimates of the likelihood. We make use of $N = 1,000$ particles in the BPF, $K = 15,000$ iterations in the PMH algorithm and discard the first $5,000$ iterations as burn-in. We select an independent proposal for u given by $u \sim \mathcal{N}(0, 1)$ and the random walk proposal in (3.32) for θ. The posterior covariance is estimated using a pilot run and it is given by

$$\widehat{\Sigma} = 10^{-3} \cdot \begin{bmatrix} 7 & 2 & 1 & 0 \\ 2 & 48 & 3 & -1 \\ 1 & 3 & 1 & 0 \\ 0 & -1 & 0 & 0.05 \end{bmatrix}.$$

In Figure 3.9, we present the marginal parameter posterior for the four parameters in the model. We note that the Phillips curve hypothesis is not supported by this data set as there seems to be little support for that β is negative. Remember that the sign of β determines the correlation between the inflation and unemployment rates. The Phillips curve hypothesis stated that this correlation should be negative and this then implies a negative value for β.

Furthermore, it is possible to make use of the PMH algorithm to estimate the NAIRU by averaging over the state of the Markov chain. This means, that we can compute the posterior of the latent state with the parameters marginalised out. Hence, we take into account the uncertainty in the parameter posterior when estimating the state. This is different from the approach in Example 3.3 on page 54, where the parameters are fixed to a single value.

We present the resulting estimate of the NAIRU using PMH in the lower part of Figure 3.6 on page 53. Note that the estimates from the bPF and the PMH algorithms are similar.

We return to this model in Example 4.1 on page 83.

Outlook and extensions

We conclude this chapter with an outlook and some possible extensions to the methods that we have introduced. As discussed in the beginning of the chapter, there are two different main approaches to approximate intractable posterior distributions. We have presented the approach based on statistical simulation in the form of Monte Carlo methods. However, variational inference can be useful in applications when speed is important. This approach is often based on approximating the posterior by a Gaussian (or some other member of the exponential family of distributions) and make use of simple updates in analogue with conjugate priors. Good general introductions to these methods are provided in the books

by Bishop (2006) and Murphy (2012). A recent survey aimed for statisticians is provided by Blei et al. (2016).

The main difficulty in using importance sampling is to find a suitable proposal distribution to sample from. This is especially difficult in high-dimensional problems, where SMC and MCMC can be better alternatives. However, there are interesting methods in the literature to adapt and/or combine several proposal distributions in importance sampling, see Cornuet et al. (2011) and Veach and Guibas (1995). This idea can also be used in SMC algorithms as proposed by Kronander and Schön (2014). Another approach to construct better proposals for SMC algorithm is introduced by Naesseth et al. (2015). Population Monte Carlo makes use of similar ideas and can be an interesting alternative, see Cappé et al. (2004). It is also possible to make use of SMC algorithms for parameter inference in SSMs and other interesting models. One such algorithm is the SMC^2 algorithm proposed by Chopin et al. (2013). Finally, Quasi-random numbers can be useful in both importance sampling and SMC to decrease the variance in the estimates, see Gerber and Chopin (2015).

Adaptive algorithms have also been developed for the MH and PMH algorithms, see Andrieu and Thoms (2008) and Peters et al. (2010). These approaches are often based on the rule-of-thumb for selecting the optimal proposal and computes the unknown covariance matrix on-the-fly. It is also possible to make use of information regarding the gradient and Hessian of the log-posterior to tune the proposal. This approach was proposed by Girolami and Calderhead (2011) for the MH algorithm and we return to it in the context of the PMH algorithm in Chapter 4.

Finally, there are a large number of additional MCMC algorithms, which are not discussed in this thesis. Hamiltonian MCMC (HMC; Duane et al., 1987) is a interesting approach to sample from high-dimensional targets using simulated Hamiltonian dynamics, see Neal (2010). A big challenge is to create a pseudo-marginal version of this algorithm for parameter inference in latent variable models such as the SSM. Sequential MCMC has been discussed by e.g., Brockwell et al. (2010) as an alternative to PMH. Slice sampling introduced by Neal (2003) is also an interesting and popular alternative. A pseudo-marginal version of slice sampling was recently proposed by Murray and Graham (2015). Finally, MCMC algorithms can also be useful for optimisation and a pseudo-marginal scheme for this is proposed by Finke (2015).

4

Strategies for accelerating inference

We know from the previous chapters that Monte Carlo methods are useful for enabling Bayesian inference in many interesting models. A drawback with these approaches are that a large number of samples from the posterior are usually required to obtain reasonable accuracy. This is often not a problem for simpler Monte Carlo methods, where samples can be generated efficiently. However, for more complicated models it can take considerable time to generate a sample.

There are two main approaches to mitigate this problem: (i) decreasing the computational cost of generating a sample or (ii) decreasing the required number of samples by making better use of them. We explore both strategies in this chapter to accelerate inference, i.e., decrease the total time for the inference, for some problems. We later show how these strategies are employed in the papers included in this thesis. Another aspect for the user is the time and complexity of the implementation of an inference algorithm for a new model. It can therefore be beneficial to apply simpler methods that are quicker to implement and tune than more elaborate algorithms that may be more efficient but takes longer time to get up an running. The total time for the entire inference can therefore be shorter in the former case than in the latter, even if the advanced algorithm is more efficient.

To statistical procedure to fit a model to data consists of three steps as discussed in the beginning of Chapter 2: collecting data, selecting a model and inferring the parameters of the model given the data. In this chapter, we propose some strategies accelerating inference in these three steps. That is, starting with how to generate and collect the data, then how to change the model and lastly how to change the inference algorithm to make faster and easier inference.

Increasing the amount of information in the data

The first approach to accelerate inference is to make data more informative about the parameters. If this is accomplished, we can obtain accurate estimates of the parameters using a smaller amount of observations T. In the SMC algorithm, this results in a linear[1] decrease of the computational cost that also carries over to the PMH algorithm and similar methods. In Paper H, we present results that illustrates how more informative data can actually increase the convergence rate of the EM algorithm.

A common approach to make data more informative is the use of an input signal which excites the system. In the field of input design, the main objective is to construct this input $u = \{u_t\}_{t=1}^T$ such that the observations $y = \{y_t\}_{t=1}^T$ are as informative as possible. The amount of information in the observations y is described by the Fisher information matrix $\mathcal{I}(\theta^\star)$, see (2.17), which is the curvature of the log-likelihood at the true parameter $\theta^\star$. A larger size of the Fisher information matrix also results in a smaller asymptotic variance of the Bayes and maximum likelihood estimators, see (2.19). To quantify the size of this matrix, we usually employ the logarithm of its determinant denoted by $h(\mathcal{I}(\theta^\star)) = \log\det\mathcal{I}(\theta^\star)$. A suitable input can then be computed by

$$u^\star = u(\alpha^\star), \quad \alpha^\star = \operatorname*{argmax}_\alpha h\big(\mathcal{I}(\theta^\star, u(\alpha))\big),$$

for some family of inputs $u(\alpha)$ parametrised by α and where $\mathcal{I}(\theta^\star, u)$ denotes the Fisher information matrix when the input u is applied.

We encounter two main problems with this approach for finding $u^\star$ namely: (i) how do we parametrise a good input $u(\alpha)$ for a specific model and (ii) how do we compute/estimate $\mathcal{I}(\theta^\star, u)$. We revisit these problems in Paper H. For problem (i), we make use of graph theory to construct a number of basis inputs. The optimal input signal is formed by finding the convex combination of these basis inputs with weighting α such that the size of the Fisher information matrix is maximised.

For problem (ii), the Fisher information matrix is estimated for each basis input by using SMC methods and the Fisher identity introduced in Remark 3.5. The major challenge is to obtain accurate estimates of $\mathcal{I}(\theta^\star, u)$ with a reasonable computational cost. We propose to accomplish this by adapting an estimator introduced by Segal and Weinstein (1989) for the Kalman filter to the SMC framework. However, the Louis identity from Remark 3.5 can also be used. The main problem with this alternative is that the estimates of the information matrix often are negative definite. The estimates from the approach proposed by Segal and Weinstein (1989) are always PD.

Approximating the model

The second approach that we consider to accelerate inference is to approximate the model and then make use of standard algorithms for inference. Note that the approximation of the model typically results in a bias in the estimates. However, it could potentially have a large

[1] Actually quadratic since the number of particles N typically needs to scale linearly with T, which results in a computational cost proportional to NT $\propto T^2$.

impact on the computational cost and can enable inference in otherwise intractable models. Here, we discuss using: (a) sparseness priors for model order selection, (b) approximate Bayesian computations for SSMs with intractable likelihoods and (c) using a surrogate function to approximate the log-likelihood and/or log-posterior.

Over-parametrised models with sparseness priors

Model order selection is an important problem in statistics. For example, it is a challenging task to determine a suitable model order p of an AR process (2.6) to best fit a given data set. In Bayesian inference, we can introduce the model order as a parameter to be inferred together with the parameters of the model. This type of inference can be carried out by the reversible-jump Metropolis-Hastings (RJMH) algorithm introduced by Green (1995). However, it can be challenging to find proposal distributions for p and θ that result in reasonable mixing. Furthermore, implementation can be challenging even for simple models. See Paper F for a concrete example where we infer p for an ARX process using RJMH.

An alternative approach is to make use of an over-parametrised model and employ sparseness priors to penalise using more parameters than supported by the data. This is similar to the use of regularisation in linear regression as discussed in Remark 2.5. The surprising result is that this can provide consistent estimates in some cases as proven by Rousseau and Mengersen (2011). That is, the posterior distributions of the parameters and the model order are asymptotically the same compared with using an exact approach such as the RJMH algorithm. The main benefit is that sparsity in some cases can be induced by using conjugate priors. This enables making use of Gibbs sampling, which can be simpler to implement compared with the RJMH algorithm and can also result in better mixing. Hence, we can say that this accelerates inference in some type of models, where the model order is unknown.

A concrete example of this is again model order selection in ARX models as discussed in Paper F. Here, we make use of ARD priors to induce sparsity in this case. This corresponds to an hierarchical model where a zero-mean Gaussian prior is selected for the AR coefficients, i.e., $\phi_k \sim \mathcal{N}(0, \sigma_0^2)$. Moreover, we assume that $\sigma_0^2 \sim \text{IG}(a_0^\sigma, b_0^\sigma)$, where the shape $a_0^\sigma > 0$ and scale $b_0^\sigma > 0$ are so-called hyperparameters. This is similar to the L2-RLS discussed in Remark 2.5 on page 30 with the major difference that the prior variance is not fixed but estimated from the data.

Another example is mixture models to capture heterogeneity in the individual random effects in panel data. In the mixed effects model (2.9), we assumed that the random effects were distributed according to a Gaussian distribution with some unknown mean and variance. In some applications, it would be interesting to allow for multi-modal distributions of the random effects where each mode captures the behaviour of a certain subgroup of the individuals. Two illustrations of this are presented in Figure 4.1, where the distributions clearly indicate that there are a number of clusters of random effects in the data.

One possible approach to model the distributions of the random effects is by using a Dirichlet process mixture (DPM), see (2.27). However, inference in this type of models can be challenging both from implementation perspective and because of poor mixing, see Hastie et al. (2015). An alternative proposed by Ishwaran and Zarepour (2002) is to make use of an over-parametrised Gaussian finite mixture model and put a sparseness prior on the number

of components. The resulting inference algorithm is easier to implement and performs well in many problems, see e.g., Chapter 22 in Gelman et al. (2013). We make use of this approach and compare with using DPM in Paper G. The main objective is to compare the posterior estimates to validate if the findings in Rousseau and Mengersen (2011) carries over to this settings.

Approximate Bayesian computations

For some models, it is not possible to approximate the posterior as the target cannot be evaluated point-wise. This could be the result of that the likelihood cannot be expressed in closed-form or that the computation cost is high. In an SSM, this problem corresponds to it is not possible evaluate $g_\theta(y_t \mid x_t)$ point-wise. For a BPF, this results in that the weights cannot be computed. An example of an SSM with an intractable likelihood is when the observations are modelled using an α-stable distribution (Nolan, 2003). This is a popular choice in finance as discussed in Papers C and E.

It turns out that we can reformulate the SSMs with intractable likelihoods by introducing a small perturbation, which allows us to apply a standard particle filter. This approach is part of a family of methods known as ABC (Marin et al., 2012). For the particle filter, we can construct a SMC-ABC algorithm (Jasra et al., 2012) to estimate the log-likelihood. In this algorithm, we assume that the observations are perturbed by

$$y_t^\star = y_t + \epsilon z_t,$$

where $\epsilon \geq 0$ denote the tolerance parameter and z_t denote a standard Gaussian random variable. Hence, we make use of $y^\star = \{y_t^\star\}_{t=1}^T$ as the observations from the model and $\{x_t, y_t\}_{t=1}^T$ as the latent states. This results in that it is only required to be able to generate samples from $g_\theta(y_t \mid x_t)$ and not evaluate it point-wise, which is usually less restrictive. The weighting function for the resulting ABC version of the BPF algorithm is given by $w_t^{(i)} = \mathcal{N}(w_t^{(i)}; y_t - y_t^{(i),\star}, \epsilon^2)$, where $y_t^{(i),\star}$ denotes a sample from $g_\theta(\cdot \mid x_t^{(i)})$.

Note that we recover the original model when $\epsilon \to 0$, see Dean and Singh (2011). In practice, it is required that $\epsilon > 0$ to balance the computational cost with the accuracy of the estimate. We can employ standard parameter inference methods like the PMH algorithm to estimate the parameters of the approximate model implied by the SMC-ABC algorithm. The main problem with this is that the variance of the log-likelihood estimates often is larger than for a standard SMC algorithm. This can result in bad mixing in the PMH algorithm. We return to this in Section 4.3 and in Section 5.2 (page 197) of Paper C.

Note that using ABC can in itself lead to an acceleration of the inference when the likelihood is computationally prohibitive to evaluate. This could be a problem when the number of observations is large. Then evaluating the likelihood at every iteration of a MH algorithm could be problematic. We return to discussing Bayesian methods for big data in Chapter 5.2.

Building a surrogate of the posterior

From Example 3.6, we know that the estimates of the log-target and its gradients can be obtained by particle methods. However, they can be quite noisy when N is small and

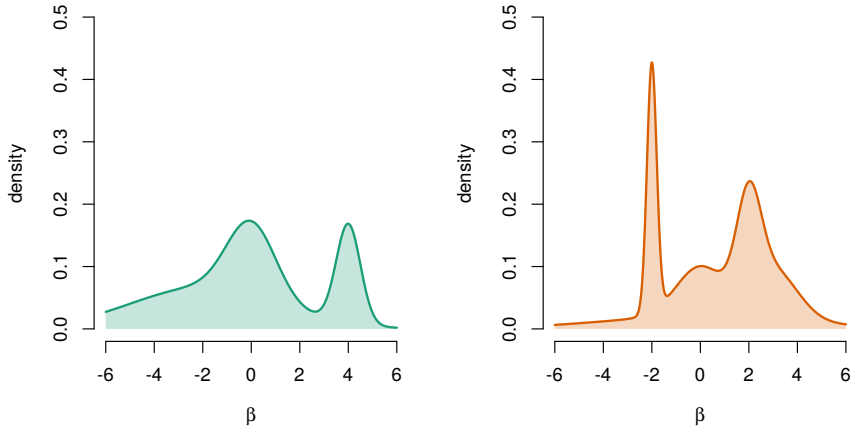


Figure 4.1. Illustration of two multi-modular distributions of the individual random effects β_i in a mixed effects model (2.9).

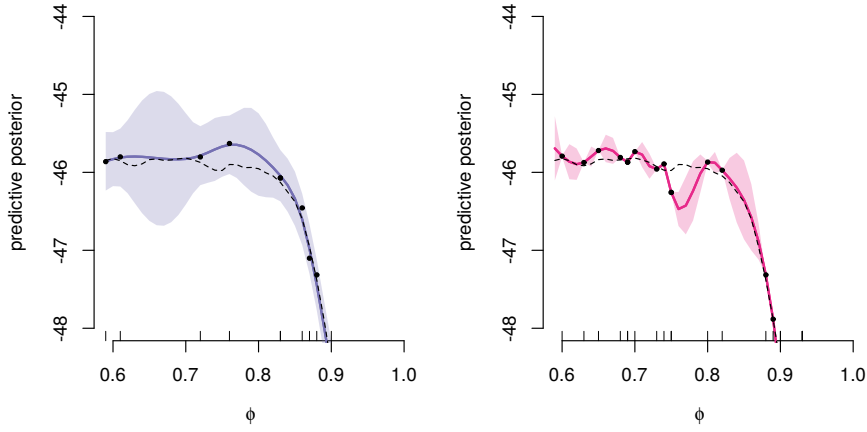


Figure 4.2. The predictive GP posterior for the log-likelihood estimates from the BPF in Example 3.6 using 10 (left) and (20) random values of ϕ. The dashed lines indicate the true log-likelihood and the shaded areas indicate the 95% credibility intervals.

increasing the number of particles also increases the computational cost of the particle method. When the noise variance is small, it is possible to make direct use of these estimates for computing the maximum likelihood estimate or the maximum a posteriori estimate of θ. For example, by using finite difference approaches, such as in the simultaneous perturbation and stochastic approximation (SPSA; Spall, 1987, 1998) algorithm. The gradient estimate can be utilised in a standard gradient ascent algorithm (Poyiadjis et al., 2011; Doucet et al., 2013) and in a Newton algorithm together with an estimate of Hessian (Kok et al., 2015).

Another approach is to make use of GP regression to build a surrogate function of the log-posterior. The surrogate should be smooth and cheap to evaluate, where both requirements are fulfilled by the GP predictive posterior. In Figure 4.2, we present an example of this by revisiting the log-likelihood estimates generated in Example 3.6. We randomly select 10 and 20 samples to create two surrogate functions using GP regression. We note that predictive mean differs slightly from the true log-likelihood (dashed line) around the parameter estimate. Hence, optimising the mean function will give a similar result to the mode of the posterior estimates in Figure 3.9.

After that the surrogate has been computed, it is easy to make use of a quasi-Newton algorithm to find the parameter estimates by optimising the predictive mean function. However, the predictive covariance function is also informative as it enables us to construct confidence intervals. This information can be used to decide in which parameters the log-posterior should be sampled in next. In Figure 4.2, it would be beneficial to decrease the predictive covariance around $\phi = 0.85$ as a peak in the log-likelihood could be hiding there. This is the idea behind the Bayesian optimisation (BO; Mockus et al., 1978) algorithm, which is especially designed for optimising noisy and expensive objective functions. In this thesis, we refer to the BO algorithm as the Gaussian process optimisation (GPO) algorithm as the surrogate function is a GP predictive posterior. General introductions to BO are given by Snoek et al. (2012), Lizotte (2008), Boyle (2007) and Osborne (2010).

We make use of GPO for parameter estimation in SSMs using maximum likelihood in Dahlin and Lindsten (2014) by combining it with the BPF algorithm for estimating the log-likelihood. The results are encouraging and the convergence is faster than compared with the SPSA algorithm in the number of log-likelihood estimates. In Paper E, we revisit the problem for SSMs with intractable likelihoods. In this setting, we can make use of SMC-ABC to compute estimates of the log-likelihood. However, the SMC-ABC algorithm often requires a larger value of N compared with the standard SMC algorithm to obtain reasonable accuracy in the log-likelihood estimates. This results in poorly mixing and computationally costly inference algorithms based on the PMH algorithm, which is discussed in Paper C.

Instead, we make use of the GPO algorithm in combination with SMC-ABC to estimate a Laplace approximation (Gelman et al., 2013) of the log-posterior. This can be seen as an approximate Bayesian inference algorithm or be applied as a method to find a suitable initialisation and parameter proposal for the PMH algorithm. This allows use to decrease the number of required estimates of the log-likelihood from about $10,000$ to 350 comparing with PMH in one example, which corresponds to decreasing the computational time from days to half-an-hour. Note that the use of a GP as the surrogate incurs a small computational overhead. However, the main computational cost in both the GPO and the PMH algorithms is incurred by the SMC algorithm applied for estimating the log-target.

Improving the inference algorithm

The third approach to accelerate inference is to make changes to the inference algorithms. Here, we discuss two alterations to the PMMH/PMH algorithm based on: (a) inducing correlation in the estimates of the target and (b) tailoring the parameter proposal to the target distribution.

Correlating auxiliary variables

Sometimes it is useful to extend the model with auxiliary parameters to make inference easier. One such approach is the pseudo-marginal algorithms presented in Section 3.3. In these algorithms, we can compute estimates of the target by introducing the auxiliary variables u. This makes it possible to use the MH algorithm to sample from target distributions, which we cannot evaluate point-wise but can estimate using e.g., importance sampling.

If we revisit Example 3.13, we made use of an independent proposal for u, which basically means that the particle filters are independent. From Remark 3.4, we know that the likelihood estimates from the particle filter are noisy with a variance that decreases proportional to $\sqrt{N}$. We can therefore increase N to decrease the variance in the likelihood and thus increasing the mixing in the PMH algorithm (up to the mixing of the optimal algorithm). However, this makes each iteration of the PMH algorithm slower as the computational cost increases. An interesting question is then if it is possible to correlate the errors in the likelihood estimates to increase the mixing without increasing N.

One approach for this is to realise that the particle filter is a deterministic algorithm given the random variables u, which are used for resampling and propagating particles. We can thus make the likelihood estimates positively correlated by inducing a correlation in u. A random walk proposal for u is not a good choice as millions of random variables typically are required in the particle filter. Therefore, a random walk would not explore the space $\mathcal{U}$ efficiently. Instead, we propose in Paper D to make use of a Crank-Nicolson (CN; Beskos et al., 2008; Cotter et al., 2013; Hairer et al., 2014) proposal for u. A CN proposal is a specific AR(1) process (2.6) given by

$$q(u' \mid u) = \mathcal{N}\left(u'; \sqrt{1-\sigma_u^2}u, \sigma_u^2 \mathbf{I}_{N_u}\right),$$

where N_u denotes the number of elements in u and $\sigma_u > 0$ denotes a parameter determined by the user. Note that for the BPF algorithm, we have that $N_u = (N+1)T$ when using N particles and systematic resampling in a scalar LGSS model.

In Figure 4.3, we present the correlation in the likelihood estimates in two consecutive iterations of the PMMH algorithm keeping θ fixed. Here, we use $T = 10$ samples from a LGSS model (2.13) with $\theta = \{0.0, 0.5, 1.0, 0.1\}$ while varying N in the particle filter and σ_u on the unit interval. We note that the correlation decreases from one to zero as we change the value of σ_u. In the CN proposal, we obtain the independent proposal for u by setting $\sigma_u = 1$. Hence, it is possible to control the correlation in the likelihood estimates by σ_u.

We apply the CN proposal for u in the PMMH algorithm in Paper D and investigate the impact on the IACT. We present numerical simulations that indicate a three-fold increase in

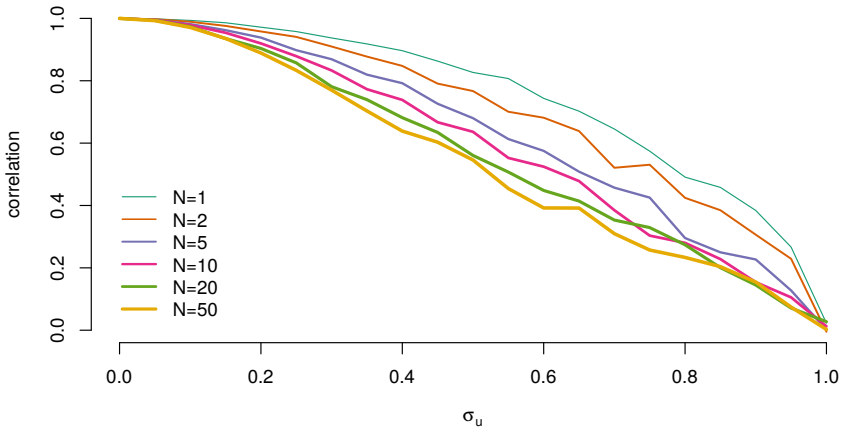


Figure 4.3. The correlation between two consecutive log-likelihood estimates from Example 3.1 with N particles and random variables sampled from the CN proposal when varying σ_u.

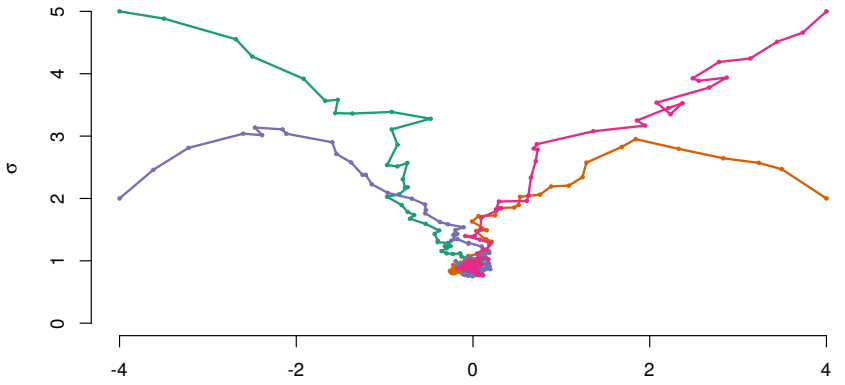


Figure 4.4. Four random walks on the Riemann manifold induced by the standard Gaussian IID model with $n = 100$ samples.

the IACT compared with when using an independent proposal for u. Note that only small changes in the implementation are required to switch from an independent proposal for u to the CN proposal. It is especially simple when all random variables can be generated using quantile transformations, see Section 3.2.1. This small change allows us to decrease N significantly, which results in an acceleration of the algorithm in terms of computational cost. However, the drawback is that u' and u are stored in memory at all times, which can be a limiting factor when N is large.

Furthermore, Deligiannidis et al. (2015) presents encouraging theoretical results indicating that it is possible to select $N \propto T^{\alpha}$ for $\alpha < 1$. They provide a numerical experiment illustrating that only a handful of particles is required to infer the parameters of a LGSS model given tens of thousands of observations. This is an impressive result and leads to a speed-up by 100 times. However, when T is that large the Bernstein-von-Mises theorem (see Section 2.3) kicks in and alternative methods are probably even faster. Also, they make use of a fully-adapted particle filter, which is a particle filtering using the optimal proposal for particles and the optimal weighting function, see Paper A.

We make use of a BPF for estimating the target in Paper D. In that case, we do not obtain such radical improvement in the efficiency of the algorithm. However, we still increase the mixing with a factor of about three and outperform the standard PMMH algorithm using an independent proposal for u.

Tailoring the proposal to the target geometry

In Section 3.2.3, we introduced the MH algorithm and presented two choices of proposals namely the independent and random walk. Both proposals are known as *blind* proposals as they do note take any information about the target into account when proposing the candidate state x'. The convergence to the sought target distribution is instead ensured by the accept/reject mechanism. An interesting question is therefore how to construct a proposal that takes e.g., the gradient of the log-target into account. Here, we discuss one such approach based on continuous time diffusion processes.

An interesting class of (continuous time) stochastic processes are the Itô diffusions governed by the stochastic differential equation (SDE) given by

$$\mathrm{d}X_t = b(X_t)\mathrm{d}t + \sigma(X_t)\mathrm{d}B_t, \quad X_0 = x_0,$$

where $b(\cdot)$ and $\sigma(\cdot)$ denote the drift and the volatility, respectively. A possible choice for these quantities is given by

$$b(X_t) = \frac{1}{2}\nabla \log \pi(X_t), \qquad \sigma(X_t) = 1,$$

then the resulting process is known as a *Langevin diffusion* or *Brownian dynamics*. This process has the interesting property that it has π as its stationary distribution and the resulting process is ergodic, see Livingstone and Girolami (2014) and Roberts and Tweedie (1996). Hence, for any initialisation a proposal based on this diffusion has the target as its stationary distribution and samples obtained after the burn-in are from the sought target.

In practice, it is common to make a discretisation of the Langevin diffusion using a first-order Euler approximation to obtain

$$q(x' \mid x) = \mathcal{N}\left(x'; x + \frac{\epsilon^2}{2}\nabla \log \pi(x),\ \epsilon^2 \mathbf{I}_p\right), \tag{4.1}$$

for some discretisation length $\epsilon > 0$. We see from the Langevin proposal (4.1) that the gradient of the log-target acts like a drift towards areas of high posterior probability. This could in theory be useful for exploring the target distribution more efficiently. Note that the gradient of the log-posterior is given by the gradient of the log-prior and the score function (2.15). The resulting algorithm from combining this proposal with the MH algorithm is known as the Metropolis adjusted Langevin algorithm (MALA; Roberts and Stramer, 2003). It is possible to show that the Markov kernel in the MALA algorithm is (geometrically) ergodic if the tails of the target are heavier than for a Gaussian distribution and lighter than for an exponential distribution, see Roberts and Tweedie (1996).

A possible extension of (4.1) is to include a matrix $\Sigma(x)$ to scale the gradient and determine the variance of the proposal. This results in the proposal given by

$$q(x' \mid x) = \mathcal{N}\left(x'; x + \frac{\epsilon^2}{2}\Sigma^{-1}(x)\nabla \log \pi(x),\ \epsilon^2 \Sigma^{-1}(x)\right), \tag{4.2}$$

where $\Sigma(x)$ can be selected as any positive semi-definite matrix or function. An interesting choice proposed by Girolami and Calderhead (2011) is to make use of

$$\Sigma(x) = -\nabla^2 \log \pi(z)\big|_{z=x},$$

which corresponds to a parameter dependent matrix. We refer to the MH algorithm with this proposal as the (simplified) manifold MALA (MMALA) algorithm. This choice of $\Sigma(x)$ corresponds to sum of the negative Hessian of the log-prior and the observed information matrix (2.16).

The main benefit of the MMALA algorithm is that the gradients are scaled by the curvature of the log-posterior. This is useful as the proposal takes larger steps when the Markov chain is far from the target mode and smaller steps as it gets closer. In Figure 4.4, we present four different Markov chains governed by (4.2). Here, we make use of a standard Gaussian IID model as the target and would like to infer the mean μ and the standard deviation σ from $n = 100$ samples. We see that the Markov chain behaves as expected with large step lengths far from the mode, which decreases as the chain approaches the true parameters $\{\mu, \sigma\} = \{0, 1\}$.

The Markov chain governed by (4.2) can be seen as a random walk on a Riemann manifold, see Girolami and Calderhead (2011) and Livingstone and Girolami (2014). They highlight that there are other choices for $\Sigma(x)$ that could be useful in practice. The Langevin proposal can also be seen as a Laplace approximation of the posterior as discussed by Robert and Casella (2004) and in Paper B. A third interpretation proposed in Paper C is that MALA and MMALA corresponds to noisy gradient ascent and noisy Newton algorithms, respectively.

The major potential benefit with using the MALA and the MMALA is that they can increase the mixing in the Markov chain. In Section 3.2.3, we showed that the mixing is connected with the asymptotic variance of the estimate of the target distribution. Increasing the mix-

ing can therefore have a positive impact on the computational time of the algorithm and accelerate inference. This as it is possible to decrease K and still obtain a similar accuracy in the estimate. Less iterations of the MH algorithm leads to a decreased computational cost. Another benefit for the MMALA algorithm is that it requires less tuning than a standard random walk proposal. This as the user only needs to tune the step length ϵ and not an entire covariance matrix as for the random walk proposal. This is a decrease from p^2 tuning parameters to 1 for the proposal, which potentially decreases the amount of user interaction.

In Papers B and C, we propose particle versions of the MALA and MMALA algorithms to increase the mixing compared to the standard PMH algorithm from Section 3.3. The main challenge is to obtain estimate of the intractable gradient and the Hessian with respect to the log-target distribution. We employ FL particle smoothing from Remark 3.2 together with the results from Remark 3.5 to estimate these quantities. The primary benefit is that this type of smoother does not increase the computational complexity of the algorithm and gives reasonable accuracy. The latter is important for the performance of the proposed algorithm as analysed by Nemeth et al. (2014).

Example 4.1: How does unemployment affect inflation? (cont. from p. 70)

We refer to the particle version of MMALA as the PMH algorithm of second-order (PMH2) since it makes use of gradient and Hessian information regarding the target. The basic algorithm for PMH2 is presented in Paper B. QPMH2 is presented in Paper C and makes use of quasi-Newton techniques (Nocedal and Wright, 2006) to estimate the Hessian.

We return to the problem of estimating the parameter β in the Phillips curve model to compare the PMH0 and the QPMH2 algorithms. Remember that the PMH0 algorithm makes use of a random walk proposal such as (3.32), which does not include information about the gradient and Hessian of the target. Furthermore, we make use of a post-processing of the Markov chain known as zero-variance (ZV; Mira et al., 2013; Papamarkou et al., 2014) to decrease the variance in the estimates. The ZV approach is based on the idea of control variates in vanilla Monte Carlo sampling, see Robert and Casella (2004). We can apply ZV to decrease the variance in the estimate of the posterior mean by

$$\widetilde{\beta}_k = \beta_k + \frac{\rho}{2}\mathcal{G}(\beta_k), \quad \rho = -\mathbb{V}[\mathcal{G}(\beta_{1:K})]\mathbb{C}[\beta_{1:K}, \mathcal{G}(\beta_{1:K})],$$

where $\mathcal{G}(\beta_k) = \nabla \log p(\theta \,|\, y)\big|_{\beta=\beta_k}$ denotes the gradient of the log-posterior evaluated at β_k. Hence, we can compute the mean of $\widetilde{\beta}_{1:K}$ to estimate the posterior mean, which has a smaller variance as making use of $\beta_{1:K}$ directly.

In Figure 4.5, we present the trace plot, ACF and the posterior estimates for: QPMH2, QPMH2 with ZV (QPMH2-ZV) and the PMH0 algorithm from Example 3.13. Note that the mixing is better for the two algorithms based on QPMH2, which can be seen in both the trace and the ACF. Note the spike in the ACF at lag 100, which is due to the memory of the quasi-Newton update for the Hessian, see Paper C. Moreover, note that the variance in the posterior estimate from QPMH2-ZV is smaller than the same estimate from QPMH2. Hence, the ZV post-processing of the Markov chain seems to have a beneficial effect.

We return to this model in Example 5.1 on page 87.

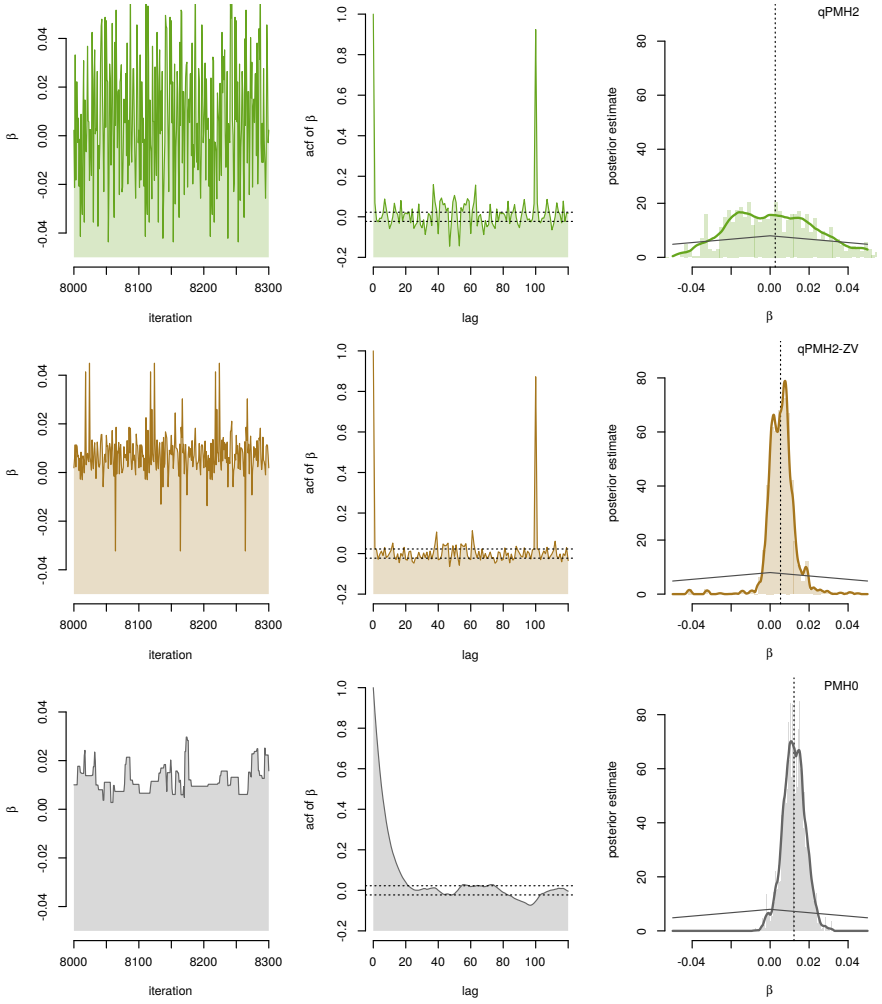


Figure 4.5. The trace, ACF and parameter posterior estimate for β in the Phillips curve model using the Swedish data. The results are presented for: QPMH2 (green), QPMH2 with ZV estimator (brown) and PMH0 (grey). Dotted lines indicate confidence intervals and estimates of posterior means. The grey curves indicate the prior distribution.

Outlook and extensions

There are many other potential approaches for accelerating inference in the algorithms. Some of them are discussed in Chapter 5. To conclude this chapter, we would like to take the opportunity to discuss some more specific ideas for accelerating the PMH algorithm.

For the PMH algorithm, we would like to highlight three different interesting approaches connected with the material in this chapter. The first is to make use of surrogate modelling of the target distribution, which allows for cheaper computations of the acceptance probability. This idea is proposed by Meeds and Welling (2014) and they make use of the predictive GP posterior similar to GPO for constructing the surrogate. An interesting direction for future work is therefore to investigate the combination of GPO and PMMH algorithms further for accelerating inference by decreasing the computational cost for each iteration.

The second approach is to alter the test function φ to decrease the variance in the resulting estimates. In vanilla Monte Carlo, this approach is known as control variates and is a useful method for variance reduction by using a known analytical approximation of the target. In the PMH algorithm, we can make use of ZV (Mira et al., 2013; Papamarkou et al., 2014) to achieve the same objective as in Example 4.1. In principle, it is straight-forward to directly apply ZV approaches for any the PMH algorithms proposed in Papers B-D to further accelerate the inference. This especially as ZV methods require estimates of the gradient of the log-posterior, which already are computed by the aforementioned algorithms. The third approach is to relax the requirement of detailed balance, see e.g. Diaconis et al. (2000) for a related reference.

5
Concluding remarks

We conclude the introductory part by discussing the two examples introduced in Chapter 2 one last time. We also summarise the contributions of the thesis in more technical terms connected to the concepts and issues discussed in the previous chapters. Furthermore, we give a summary of interesting trends and areas for future work in accelerating Bayesian inference. Finally, we discuss reproducible research and open source code in connection with this thesis and the papers included in it.

Example 5.1: How does unemployment affect inflation? (cont. from p. 70)

We are now ready to make predictions of the future inflation rate given the model and a change in the unemployment rate. In the left part of Figure 5.1, we present the future expected change in the unemployment rate as stated in this fictional scenario by the Swedish parliament. We assume that this forecast is correct and it acts as the input to our Philips curve model from Example 3.13.

To make the forecast, we draw random samples from the parameter posterior estimates and simulate the system forward in time using the SSM. The result is presented in the right part of Figure 5.1, where the purple line indicates the predictive mean. We see that the predicted mean of the inflation rate approaches the two percent target during this period but that no action needs to be taken at the moment. However, the uncertainty (purple area) is large and better models are required for long-term forecasting. In practice, the Riksbank makes use of elaborate SSMs known as dynamic stochastic general equilibrium (DSGE) models (Adolfson et al., 2007a, 2013, 2007b). to forecast the future inflation, unemployment, GDP and other macro-economic variables.

We conclude this example by noting that no strong negative correlation between the unemployment rate and the inflation rate in Sweden during the period between 1987 and 2015. We have also illustrated one simple approach for predicting future levels of inflation.

Example 5.2: Voting behaviour in the US Supreme court (cont. from p. 63)

We make use of the estimated model from Example 3.8 to investigate how the ideological leaning of the court would change if justice Scalia is replaced by a justice with: (i) the same ideological leaning or (ii) a slightly more liberal leaning. The latter choice is simulated by subtracting one from all the samples from $p(x_1|y)$ corresponding to the liberal/conservative score for justice Scalia. We make use of the model and the posteriors from Example 3.8 to simulate the number of liberal votes in $10,000$ cases. We sample the parameter for each case α_t from the aggregated posterior samples from the inference in Example 3.8.

In Figure 5.2 we present histograms of the number of liberal votes for the two scenarios. The shaded area approximates the probability of a liberal outcome from the ruling of the court. We conclude that the probability of a liberal majority changes from 0.42 to 0.47 when replacing Scalia with a more liberal justice. Hence, the court is almost balanced if the President choose alternative (b) or slightly conservative if the President choose to nominate according to alternative (a).

It is possible to repeat this procedure sequentially for each year to investigate how the ideological leaning changes over time for each justice and for the entire Court. This can be done by applying Kalman filtering from Remark 2.7 for the score $\{x_i\}_{i=1}^{9}$, see Martin and Quinn (2002) and Martin and Quinn (2007) for more information.

Summary of contributions

The contributions of this thesis strive to accelerate Bayesian inference in a number of different model classes. The acceleration could take the form of simpler implementation, being able to decrease the number of iterations or generated samples/particles while keeping the accuracy of the estimates or increasing information about the parameter in the data. Most of these improvements are the result of methodological contributions for a number of different algorithms.

In Paper A, we provide the reader with a gentle introduction to the PMH algorithm with plenty of references for further study. Source code is also provided to encourage the reader to use PMH for his/her own research problems. We propose particle versions of the MALA and MMALA in Paper B referred to as the PMH1 and the PMH2 algorithms, respectively. The main challenge in these algorithms is to obtain accurate estimates of the gradient and the Hessian of the log-posterior. We propose to make use of FL particle smoothing from Remark 3.2 together with regularisation approaches from the optimisation literature to solve this.

Furthermore, we propose a quasi-Newton scheme for estimating the Hessian in Paper C using only gradient information. This is useful in models where the Hessian estimates are corrupted by large amounts of noise. From numerical experiments in Papers B and C, we conclude that adding gradient and Hessian information can increase the mixing, shorten the burn-in phase and make the proposal scale-invariant. We also investigate the use of correlated target estimates in the PMMH algorithm in Paper D. A considerable increase in mixing can be obtained by changing the independent proposal for the random variables u

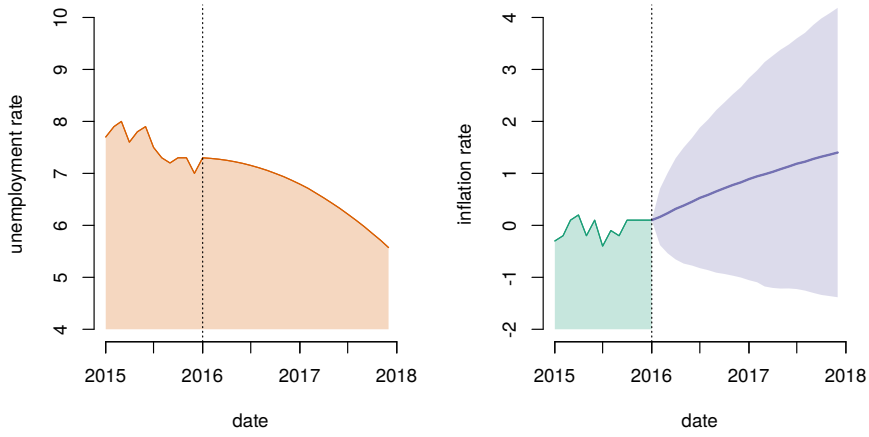


Figure 5.1. The fictional change in the unemployment rate during 24 months (left) and the resulting predicted inflation rate (right) from the Philips curve model. The predictive mean and 95% confidence intervals are indicated by the purple line and area, respectively. The dashed lines indicate the start of the forecasts.

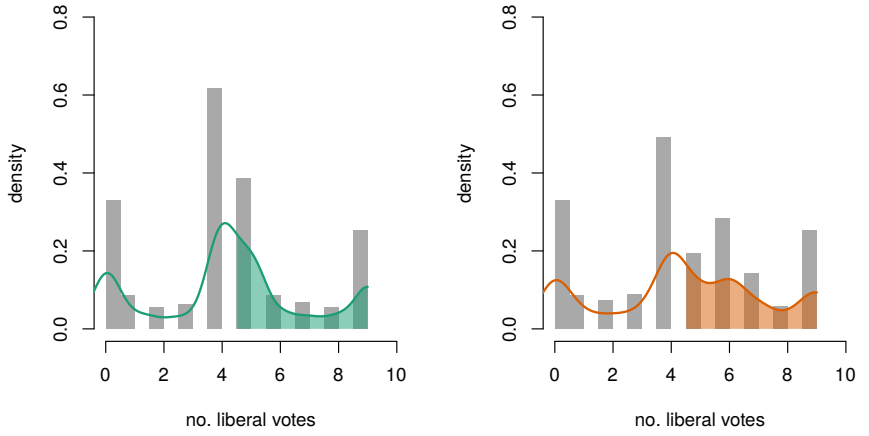


Figure 5.2. The predicted number of liberal votes when replacing justice Scalia with someone with same leaning (left) or slightly more liberal (right). The shaded areas approximate the probability of a liberal majority in an average case.

used to estimate the target to an autoregressive proposal, which sometimes only amounts to changing a few lines of code.

The main drawback with MCMC algorithms is their the computational cost can be quite large. This is especially a problem for PMH algorithms, which can take several days to execute. It is therefore interesting to investigate alternative approximate methods to carry out Bayesian parameter inference in SSMs. In Dahlin and Lindsten (2014) and Paper E, we propose a new algorithm based on the combination of particle filtering and GPO for maximum likelihood and approximate Bayesian inference in SSMs. We demonstrate that the resulting algorithm gives accurate results and can lead to a considerable speed-up compared with alternative optimisation methods and the PMH algorithm.

In Paper F, we investigate the use of ARX models with Student's t innovations to handle outliers and missing data. Furthermore, we make use of the RJMH algorithm for automatically selecting the model order. This approach is compared with making use of an over-parametrised ARX model with a sparseness prior on the AR parameters. Inference in the latter model can be carried out using Gibbs sampling. In the numerical experiments, the two approaches give similar results in terms of predictions. Good performance is also demonstrated for predicting real-world EEG data.

In Paper G, we make use of sparseness priors to simplify inference in DPM models applied for modelling random effects in panel data models. We investigate the findings by Rousseau and Mengersen (2011) and show that similar posterior estimates can be obtained by switching the DPM model to an over-parametrised finite mixture model with a sparseness prior of the mixture weights. This opens up for simpler implementation of inference algorithms in such models and can increase the mixing.

Finally in Paper H, we propose a novel approach for input design in non-linear SSMs. The optimal input is obtained as the solution to a convex optimisation problem of basis inputs computed by graph theory. The corresponding cost function depends on the Fisher information matrix for each basis input. Therefore, we propose a novel approach for estimating this quantity based on particle smoothing. We demonstrate that the input signal generated by the proposed method can increase the accuracy and convergence rate of the expectation maximisation algorithm applied for parameter inference.

Some trends and ideas for future work

In this section, we discuss some trends and ideas for future work to accelerate Bayesian inference. Further ideas and discussions are provided in the conclusions for each of the papers in Part II of this thesis.

Efficient implementations

Most of the contributions in this thesis are based on improving existing algorithms. Therefore, we have not discussed how to write efficient implementation of them to decrease the actual computational time. The main challenge with efficient implementation is typically that great care needs to be taken to optimise e.g., memory management. For instance, the

programming language Julia (Bezanson et al., 2014) has been designed with this in mind. The main advantages are the excellent computational performance together with a simple high-level syntax that resembles programming languages such as R (R Core Team, 2015) or Python. This could make efficient implementation of Monte Carlo algorithms easier in the future and lead to accelerated inference.

A related trend is to provide a general solver, which can be applied to many different types of problems. Two examples of this are STAN (Stan Development Team, 2015) and LIBBI (Murray, 2013), where the user can define a model and the program then takes care of solving the inference problem using HMC, MCMC and/or SMC. This makes inference easier for the user, who does not need to implement advanced algorithms and tune them on his/her own. Stan is already used extensively in many interesting applications, e.g., in the recent detection of gravitational waves by Abbott et al. (2016).

Similar software based on *probabilistic programming* is also becoming more popular as a general tool for carrying out inference, see Wood et al. (2014) and Mansinghka et al. (2014). Finally, GPU-implementations and *cloud computing* can also be useful for accelerating SMC and MCMC algorithms by carrying out the computations in a parallel manner, see e.g., Beam et al. (2014), Neiswanger et al. (2014), Henriksen et al. (2012) and Murray et al. (2015).

Better particle smoothing and Bayesian online methods

Many of the algorithms presented in this thesis are based on SMC methods for estimating the log-likelihood or its gradient and Hessian. The efficient implementation of these methods depends on good proposal distributions. More work is required in this area to improve SMC methods for high-dimensional targets, see e.g., Naesseth et al. (2015), Rebeschini and van Handel (2015) and Beskos et al. (2014).

Another approach is to improve the estimators for the gradient and the Hessian of the log-posterior. This is especially important for the Hessian as it should be PD (PSD). A problem when using the Louis identity as in Example 3.5 and Paper B is that the resulting estimate is often not PSD and therefore not a valid covariance matrix. Some alternative estimators are discussed in Papers C and G. However, more work is required to make PMH1/2 useful for a larger class of SSMs.

Finally, online algorithms for Bayesian inference are an important problem which currently lacks a satisfactory solution. Offline inference is provided by the MCMC algorithm, which can be applied to a wide range of problems. However, it is difficult to make use of MCMC algorithms when the target distribution changes between iterations. A natural solution is the use of particle filtering and SMC algorithms as proposed by Carvalho et al. (2010), Storvik (2002) and Fearnhead (2002). However, such algorithms can suffer from particle depletion/degeneracy, which results in that the estimators suffer from a large variance.

Scalable Bayesian inference

The amount of data generated by people, machines and sensors increases drastically for every year. Therefore, some like to refer to the current age as the era of big data. This represents new challenges for Bayesian inference algorithms to handle both the large amount of data

and the many different forms of it. A drawback with the MH algorithm and its pseudo-marginal version is that the acceptance probability depends on the likelihood of the data. If the number of observations is large, the estimation or computation of the likelihood can be computationally prohibitive, which limits the feasibility of inference.

A large number of alterations to the MH algorithm have recently been proposed to mitigate this problem. Two surveys of these recent efforts are given by Bardenet et al. (2015) and Angelino et al. (2016). One promising approach based on sub-sampling the data is proposed by Quiroz et al. (2016), which makes use of the correlated pseudo-marginal MH algorithm introduced in Paper D.

However, from the Bernstein-von-Mises theorem, we know that the posterior asymptotically concentrates to a Gaussian under some regularity conditions. It could therefore be more fruitful to make use of this in models for which the number of parameters are much smaller than the number of observations. We can then create a Laplace approximation of the mode based on the output from some optimisation algorithm targeting the log-posterior. Stochastic gradient methods have been proposed for solving this problem, where the stochasticity comes from computing the gradients using a sub-sample of the observations. This can be combined with MCMC approaches to estimate the posterior distribution as proposed by Welling and Teh (2011) and Ahn et al. (2012) or variational inference as discussed by Hoffman et al. (2012).

Probabilistic numerics

The GPO algorithm is an optimisation algorithm based on a surrogate function which often is the GP predictive posterior. It turns out that this type of surrogate modelling is useful for many other applications as well. This is the basis of the new field called *probabilistic numerics*, where similar approaches are used for solving differential equations, differentiation and integration.

The main benefits are that these methods make use of uncertainty and often require less samples from the function of interest. The former means that the approximate value of an integral is given by a probability distribution and not a point estimate. An example of the latter is the GPO algorithm, which requires less evaluations of the objective function than some other similar optimisation algorithms. More information about these methods are available from the homepage: `http://www.probabilistic-numerics.org/` as well as in Hennig (2013), Briol et al. (2015), Osborne et al. (2012), Osborne (2010) and Boyle (2007).

Reproducible research

Finally, we would like to take the opportunity to discuss the promise and necessity of reproducible research (Claerbout and Karrenbach, 1992). Before the advent of the computer, science was typically divided into theoretical and experimental fields as discussed by Vandewalle et al. (2009). In both set of fields, it is important to describe the method used to derive a proof or the experimental set-up used to study a certain phenomenon. A third branch of science based on computer experiments has developed rapidly during the last decades. However, the requirements on documenting the algorithms, their settings and data are not yet as strict as for the other two branches. This is a major problem since it often is difficult

or almost impossible to reproduce the results in peer-reviewed papers based on computer experiments. Hence, the fundamental principle of reproducibility in science is violated as new results cannot easily be verified by independent colleagues in the research community.

Naturally, it is not possible to provide implementations of the proposed algorithm for all different programming languages. However, just making the code and the data available is a big step forward in not only encouraging verification but also for encouraging further development of promising algorithms. Making the code available also makes it easier for your readers to understand your work and to suggest improvements on it. In our view, it should be as natural to provide the source code and data as a supplement to a paper as it is to provide a detailed proof of a theorem. It is only then that reviewers and colleagues really know what happens inside the *magical box*, which sometimes produce impressive plots and tables. Hence, not making the source code available is problematic and not really science as all details should be revealed.

There are also many other benefits in making the source code freely available to other researchers. One major benefit is that this practice puts pressure on commenting and documenting the code for your own sake. This is useful for decreasing the number of bugs and mistakes. Moreover, iPython notebooks (Pérez and Granger, 2007) and knitR (Xie, 2014) are excellent approaches to keep track of your own progress and write comments on how e.g., specific parameters of the algorithms were selected. This is helpful when a revision is due for a journal paper or when a new member joins the research group. Finally, if the code is not filed and documented, all the details of the algorithm are lost when e.g., a PHD student graduates and leaves the research group.

There is much more to read about reproducible research in e.g., Markowetz (2015), Fomel and Claerbout (2009), Vandewalle et al. (2009) and Donoho (2010).

Source code and data

Source code written in Python and R for recreating the examples in Part I are available from GitHub: `https://github.com/compops/phd-thesis`. Furthermore, source code for recreating some of the numerical illustrations from the papers included in the thesis are also available from `http://code.johandahlin.com`. See the `README.md` file in each repository for dependencies, instructions and implementation details. The source code and data are provided under various open source licenses with no guaranteed support and no responsibility for their use and function. Note that some data have to be download from other sites due to licensing issues.

Notation

Probability

$\xrightarrow{\text{a.s.}}$	Almost sure convergence.
$\xrightarrow{\text{d}}$	Convergence in distribution.
$\xrightarrow{\text{p}}$	Convergence in probability.
$\mathbb{P},\mathbb{E},\mathbb{C},\mathbb{V}$	Probability, expectation, variance and covariance operators.
$\sim$	Sampled from or distributed according to.
$\pi[\varphi]$	The expected value of φ under the distribution π.

Statistical distributions

$\delta_{x'}(\mathrm{d}x)$	Dirac distribution (measure) located at $x = x'$.
$\mathcal{A}(\alpha, \beta, \gamma, \eta)$	α-stable distribution with stability α, skewness β, scale γ and location η.
$\mathcal{B}(p)$	Bernoulli distribution with success probability p.
$\mathcal{D}(\alpha)$	Dirichlet distribution with concentration parameter α.
$\mathcal{DP}(\alpha, G_0)$	Dirichlet process with concentration parameter α and base measure G_0.
$\mathcal{N}(\mu, \sigma^2)$	Gaussian (normal) distribution with mean μ and variance σ^2
$\mathcal{GP}(\mu, \kappa)$	Gaussian process with mean function μ and covariance function(kernel) κ
$\mathcal{G}(\alpha, \beta)$	Gamma distribution with rate α and shape β.
$\mathcal{IG}(\alpha, \beta)$	Inverse Gamma distribution with rate α and shape β.
$\mathcal{M}(n, p)$	Multinomial distribution with n trials and probability p.
$\mathcal{P}(\lambda)$	Poisson distribution with mean λ.
$\mathcal{U}(a, b)$	Uniform distribution on the interval $[a, b]$.

Operators and other symbols

$\mathsf{0}_p$	zero p-vector.
$\mathbf{I}_d$	$d \times d$ 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.
$A^{-1}, A^\top$	Matrix inverse/transpose of A.
$\mathsf{tr}(A)$	Matrix trace 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

$\mathcal{G}(\theta)$	Gradient of log-posterior/target evaluated at θ.
$\mathcal{I}(\theta)$	Expected information matrix evaluated at θ.
$\mathcal{H}(\theta)$	Negative Hessian of log-posterior/target evaluated at θ.
$\mathcal{J}(\theta)$	Observed information matrix evaluated at θ.
$\widehat{\theta}$	Parameter estimate.
$p(\theta\|y)$	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

$a_t^{(i)}$	Ancestor of particle i at time t.
$R(x_{k-1}, x_k)$	Markov kernel.
$\mathcal{Z}$	Normalisation constant.
$x_t^{(i)}$	Particle i at time t.
$q_t(x_t\|x_{0:t-1})$	Particle proposal kernel.
$W_\theta(x_t, x_{t-1})$	Particle weighting function.
$q(\theta)$	Proposal distribution.
x_k	State of the Markov chain at iteration k.
$\pi(\theta)$	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

a.s.	Almost surely (with probability 1).
ABC	Approximate Bayesian computations.
ACF	Autocorrelation function.
AR(p)	Autoregressive process of order p.
ARD	Automatic relevance determination.
ARX(p)	Autoregressive exogenous process of order p.
BO	Bayesian optimisation.
BPF	Bootstrap particle filter.
CDF	Cumulative distribution function.
CLT	Central limit theorem.
DP(M)	Dirichlet process (mixture).
FAPF	Fully-adapted particle filter.
FFBSM	Forward-filtering backward-smoothing.
FFBSI	Forward-filtering backward-simulation.
FL	Fixed-lag (particle smoother).
GP(O)	Gaussian process (optimisation).
GPU	Graphical processing unit.
HMM	Hidden Markov model.
IACT	Integrated autocorrelation time.
IID	Independent and identically distributed.
(SN)IS	(self-normalised) Importance sampling.
KDE	Kernel density estimate/estimator.
LGSS	Linear Gaussian state space.
(P)MCMC	(particle) Markov chain Monte Carlo.
(RJ)MH	(reversible-jump) Metropolis-Hastings.
ML	Maximum likelihood.
MLE	Maximum likelihood estimator.
MSE	Mean square error.
PD	Positive definite.
PDF	Probability density function.
PMF	Probability mass function.
PF	Particle filter.
PMMH	Pseudo-marginal Metropolis-Hastings.
PMH	Particle Metropolis-Hastings.
PMH0	Marginal particle Metropolis-Hastings.
PMH1	PMH using first-order information
PMH2	PMH using first and second-order information
QPMH2	quasi-Newton Metropolis-Hastings

Abbreviations (cont.)

PS	Particle smoother.
RJMH	Reversible jump Metropolis-Hastings.
RLS	Regularised least squares.
RTS	Rauch-Tung-Stribel.
SBP	Stick-breaking process.
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.

Bibliography

B. P. Abbott, R. Abbott, T. D. Abbott, and Others. The rate of binary black hole mergers inferred from advanced LIGO observations surrounding GW150914. *Pre-print*, 2016. arXiv:1602.03842v1.

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. *Sveriges Riksbank Occasional Paper Series*, 12, 2013.

S. Ahn, A. Korattikara, and M. Welling. Bayesian posterior sampling via dtochastic gradient Fisher scoring. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1591–1598, Edinburgh, Scotland, July 2012.

J. H. Albert. Bayesian estimation of normal ogive item response curves using Gibbs sampling. *Journal of Educational and Behavioral Statistics*, 17(3):251–269, 1992.

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.

E. Angelino, M. J. Johnson, and A. P. Ryan. Patterns of scalable Bayesian inference. *Pre-print*, 2016. arXiv:1602.05221v1.

C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

B. H. Baltagi. *Econometric analysis of panel data*. John Wiley & Sons, 2008.

R. Bardenet, A. Doucet, and C. Holmes. On Markov chain Monte Carlo methods for tall data. *Pre-print*, 2015. arXiv:1505.02827v1.

T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1764.

A. L. Beam, S. K. Ghosh, and J. Doyle. Fast Hamiltonian Monte Carlo ssing 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 Verlag, 1985.

A. Beskos, G. Roberts, A. Stuart, and J. Voss. MCMC methods for diffusion bridges. *Stochastics and Dynamics*, 8(03):319–350, 2008.

A. Beskos, D. Crisan, A. Jasra, K. Kamatani, and Y. Zhou. A stable particle filter in high-dimensions. *Pre-print*, 2014. arXiv:1412.3501v1.

J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *Pre-print*, 2014. arXiv:1411.1607v4.

P. Billingsley. *Probability and measure*. Wiley series in probability and mathematical statistics. John Wiley & Sons, 3 edition, 2012.

H. J. B. Birks. Numerical tools in palaeolimnology–progress, potentialities, and problems. *Journal of Paleolimnology*, 20(4):307–332, 1998.

C. M. Bishop. *Pattern recognition and machine learning*. Springer Verlag, New York, USA, 2006.

D. Blackwell and J. B. MacQueen. Ferguson distributions via Pólya urn schemes. *The Annals of Statistics*, 1(2):353–355, 1973.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Pre-print*, 2016. arXiv:1601.00670v1.

A. Bouchard-Côté, S. Sankararaman, and M. I. Jordan. Phylogenetic inference via sequential Monte Carlo. *Systematic biology*, 2012.

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

F-X. Briol, C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic. Probabilistic integration. *Pre-print*, 2015. arXiv:1510.00933v1.

A. Brockwell, P. Del Moral, and A. Doucet. Sequentially interacting Markov chain Monte Carlo methods. *The Annals of Statistics*, 38(6):3387–3411, 2010.

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

M. Burda and M. Harding. Panel probit with flexible correlated effects: quantifying technology spillovers in the presence of latent heterogeneity. *Journal of Applied Econometrics*, 28(6):956–981, 2013.

W. S. Bush and J. H. Moore. Genome-wide association studies. *PLoS Computational Biology*, 8(12):e1002822, 2012.

O. Cappé, A. Guillin, J-M. Marin, and C. P. Robert. Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13(4), 2004.

O. Cappé, E. Moulines, and T. Rydén. *Inference in hidden Markov models*. Springer Verlag, 2005.

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

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

N. Chopin. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *The Annals of Statistics*, 32(6):2385–2411, 2004.

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.

J. Claerbout and M. Karrenbach. Electronic documents give reproducible research a new meaning. In *Proceedings of the 62nd Annual International Meeting of the Society of Exploration Geophysics*, pages 601–604, New Orleans, USA, October 1992.

M. K. Condliff, D. D. Lewis, D. Madigan, and C. Posse. Bayesian mixed-effects models for recommender systems. In *Proceedings of ACM SIGIR'99 Workshop on Recommender Systems*, Berkeley, USA, August 1999.

J-M. Cornuet, J-M. Marin, A. Mira, and C. P. Robert. Adaptive multiple importance sampling. *Pre-print*, 2011. arXiv:0907.1254v5.

S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. MCMC methods for functions: modifying old algorithms to make them faster. *Statistical Science*, 28(3):424–446, 2013.

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. *Sequential Monte Carlo for inference in nonlinear state space models*. Licentiate's thesis no. 1652, Linköping University, May 2014.

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.

J. Dahlin and T. B. Schön. Getting started with particle Metropolis-Hastings for inference in nonlinear models. *Pre-print*, 2015. arXiv:1511.01707v4.

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.

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 2014b.

J. Dahlin, F. Lindsten, J. Kronander, and T. B. Schön. Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables. *Pre-print*, 2015a. arXiv:1512.05483v1.

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015b.

J. Dahlin, F. Lindsten, and T. B. Schön. Quasi-Newton particle Metropolis-Hastings. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 981–986, Beijing, China, October 2015c.

J. Dahlin, M. Villani, and T. B. Schön. Efficient approximate Bayesian inference for models with intractable likelihoods. *Pre-print*, 2015d. arXiv:1506.06975v1.

J. Dahlin, R. Kohn, and T. B. Schön. Bayesian inference for mixed effects models with heterogeneity. Technical Report LiTH-ISY-R-3091, Department of Electrical Engineering, Linköping University, Linköping, Sweden, March 2016.

A.C. Davison and D.V. Hinkley. *Bootstrap methods and their application*. Cambridge University Press, 1997.

T. A. Dean and S. S. Singh. Asymptotic behaviour of approximate Bayesian estimators. *Pre-print*, 2011. arXiv:1105.3655v1.

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, USA, jul 1998.

P. Del Moral. *Feynman-Kac formulae - genealogical and interacting particle systems with applications*. Springer Verlag, 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.

G. Deligiannidis, A. Doucet, and M. K. Pitt. The correlated pseudo-marginal method. *Pre-print*, 2015. arXiv:1511.04992v2.

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

P. Diaconis, S. Holmes, and R. Neal. Analysis of a nonreversible Markov chain sampler. *The Annals of Applied Probability*, 10(3):685–1064, 2000.

D. L. Donoho. An invitation to reproducible computational research. *Biostatistics*, 11(3): 385–388, 2010.

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. 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.

A. Doucet, M. K. Pitt, G. Deligiannidis, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2): 295–313, 2015.

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

J. Durbin and S. J. Koopman. *Time series analysis by state space methods*. Oxford University Press, 2 edition, 2012.

R. Eckhardt. Stan Ulam, John von Neumann, and the Monte Carlo method. *Los Alamos Science*, 15:131–136, 1987.

B. Efron. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979.

P. Embrechts, C. Klüppelberg, and T. Mikosch. *Modelling extremal events*. Springer Verlag, 1997.

P. Fearnhead. Markov chain Monte Carlo, sufficient statistics, and particle filters. *Journal of Computational and Graphical Statistics*, 11(4):848–862, 2002.

P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21, 2004.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

T. S. Ferguson. Prior distributions on spaces of probability measures. *The Annals of Statistics*, 2(4):615–629, 1974.

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

A. Finke. *On extended state-space constructions for Monte Carlo methods*. PhD thesis, University of Warwick, 2015.

R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London Series A*, 222:309–368, 1922.

G. Fitzmaurice, M. Davidian, G. Verbeke, and G. Molenberghs. *Longitudinal data analysis*. CRC Press, 2008.

S. Fomel and J. F. Claerbout. Reproducible research. *Computing in Science & Engineering*, 11(1):5–40, 2009.

E. B. Fox. *Bayesian nonparametric learning of complex dynamical phenomena*. PhD thesis, Massachusetts Institute of Technology, 2009.

J-P. Fox. *Bayesian item response modeling: Theory and applications*. Springer Verlag, 2010.

A. Gelman. Bayesian model-building by pure thought: some principles and examples. *Statistica Sinica*, 6(1):215–232, 1996.

A. Gelman, X-L Meng, and H. Stern. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica sinica*, 6(4):733–760, 1996.

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.

M. Gerber and N. Chopin. Sequential quasi Monte Carlo. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(3):509–579, 2015.

Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553): 452–459, 2015.

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.

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*. Springer Verlag, 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.

P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrica*, 82(4):711–732, 1995.

W. H. Greene. Econometric analysis. *Prentice Hall*, 2008.

M. U. Gutmann and J. Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Pre-print*, 2015. arXiv:1501.03291v1.

M. Hairer, A. M. Stuart, and S. J. Vollmer. Spectral gaps for a Metropolis-Hastings algorithm in infinite dimensions. *The Annals of Applied Probability*, 24(6):2455–2490, 2014.

D. I. Hastie, S. Liverani, and S. Richardson. Sampling from Dirichlet process mixture models with unknown concentration parameter: mixing issues in large data implementations. *Statistics and Computing*, 25(5):1023–1037, 2015.

T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning. *Springer Verlag*, 2009.

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

P. Hennig. Fast probabilistic optimization from noisy gradients. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, Atlanta, USA, June 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. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*, Bellevue, USA, July 2013.

N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker. *Bayesian nonparametrics*. Cambridge University Press, 2010.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

M. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Pre-print*, 2012. arXiv:1206.7051v3.

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.

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, USA, June 2014.

H. Ishwaran and M. Zarepour. Dirichlet prior sieves in finite normal mixtures. *Statistica Sinica*, 12(3):941–963, 2002.

P. E. Jacob and A. H Thiery. On nonnegative unbiased estimators. *The Annals of Statistics*, 43(2):769–784, 2015.

A. Jasra, S. S. Singh, J. S. Martin, and E. McCoy. Filtering via approximate Bayesian computation. *Statistics and Computing*, 22(6):1223–1237, 2012.

G. L. Jones. On the Markov chain central limit theorem. *Probability surveys*, 1:299–320, 2004.

T. Kailath, A. H. Sayed, and B. Hassibi. *Linear estimation*. Prentice Hall, 2000.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

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 Verlag, 2001.

M. Kok, J. Dahlin, , T. B. Schön, and A. Wills. Newton-based maximum likelihood estimation in nonlinear state space models. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 398–403, Beijing, China, October 2015.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Y. Koren. The BellKor solution to the Netflix grand prize. Netflix prize documentation, 2009. URL `http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf`.

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.

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.

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.

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.

P-S. Laplace. Essai philosophique sur les probabilités. *Académie des Sciences, Oeuvres complétes de Laplace*, 7, 1886.

B. Larget and D. L. Simon. Markov chain Monte Carlo algorithms for the Bayesian analysis of phylogenetic trees. *Molecular Biology and Evolution*, 16(6):750–759, 1999.

Q. V. Le, A. J. Smola, and S. Canu. Heteroscedastic Gaussian process regression. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 489–496, Bonn, Germany, August 2005.

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

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.

S. Livingstone and M. Girolami. Information-geometric Markov chain Monte Carlo methods using diffusions. *Entropy*, 16(6):3074–3102, 2014.

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

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

V. K. Mansinghka, D. Selsam, and Y. N. Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *Pre-print*, 2014. arXiv:1404.0099.

J-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

F. Markowetz. Five selfish reasons to work reproducibly. *Genome biology*, 16(1):1–4, 2015.

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.

A. Martin, K. Quinn, and J. H. Park. MCMCpack: Markov chain Monte Carlo in R. *Journal of Statistical Software*, 42(1):1–21, 2011.

A. D. Martin and K. M. Quinn. Dynamic ideal point estimation via Markov chain Monte Carlo for the US Supreme Court, 1953–1999. *Political Analysis*, 10(2):134–153, 2002.

A. D. Martin and K. M. Quinn. Assessing preference change on the US Supreme Court. *Journal of Law, Economics, and Organization*, 23(2):365–385, 2007.

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

M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

P. McCullagh and J. A. Nelder. *Generalized linear models*. Chapman Hall/CRC, 1989.

G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley-Interscience, 2 edition, 2008.

A. J. McNeil, R. Frey, and P. Embrechts. *Quantitative risk management: concepts, techniques, and tools*. Princeton University Press, 2010.

E. Meeds and M. Welling. GPS-ABC: Gaussian process surrogate approximate Bayesian computation. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)*, Quebec City, Canada, July 2014.

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.

A. Mira, R. Solgi, and D. Imparato. Zero variance Markov chain Monte Carlo for Bayesian estimators. *Statistics and Computing*, 23(5):653–662, 2013.

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.

C. Monteleoni, G. A. Schmidt, S. Saroha, and E. Asplund. Tracking climate models. *Statistical Analysis and Data Mining*, 4(4):372–392, 2011.

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

I. Murray and M. M. Graham. Pseudo-marginal slice sampling. *Pre-print*, 2015. arXiv:1510.02958v1.

L. M. Murray. Bayesian state-space modelling on high-performance hardware using LibBi. *Pre-print*, 2013. arXiv:1306.3277.

L. M. Murray, A. Lee, and P. E. Jacob. Parallel resampling in the particle filter. *Journal of Computational and Graphical Statistics (accepted for publication)*, 2015.

C. A. Naesseth, F. Lindsten, and T. B. Schön. Sequential Monte Carlo for graphical models. In *Proceedings of the 2014 Conference on Neural Information Processing Systems (NIPS)*, Montreal, Canada, December 2014.

C. A. Naesseth, F. Lindsten, and T. B. Schön. Nested sequential Monte Carlo methods. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, July 2015.

R. M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–741, 2003.

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, 2010.

W. Neiswanger, C. Wang, and E Xing. Asymptotically exact, embarrassingly parallel MCMC. *Pre-print*, July 2014.

R. B. Nelsen. *An introduction to copulas*. Springer Verlag, 2007.

C. Nemeth, C. Sherlock, and P. Fearnhead. Particle Metropolis adjusted Langevin algorithms. *Pre-print*, 2014. arXiv:1412.7299v1.

E. Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. URL `https://CRAN.R-project.org/package=RColorBrewer`. R package version 1.1-2.

H. Niederreiter. Quasi-Monte Carlo methods. In *Encyclopedia of Quantitative Finance*. John Wiley & Sons, 2010.

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

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

P. Orbanz. Construction of nonparametric Bayesian models from parametric Bayes equations. In *Proceedings of the 2009 Conference on Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2009.

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 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 832–840, La Palma, Canary Islands, Spain, April 2012.

A. B. Owen. Monte Carlo theory, methods and examples. Book manuscript, 2013. URL `http://statweb.stanford.edu/~owen/mc/`.

T. Papamarkou, A. Mira, and M. Girolami. Zero variance differential geometric Markov chain Monte Carlo algorithms. *Bayesian Analysis*, 9(1):97–128, 2014.

F. Pérez and R. E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, 2007.

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.

A. W. Phillips. The relation between unemployment and the rate of change of money wage rates in the United Kingdom, 1861-1957. *Economica*, 25(100):283–299, 1958.

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.

M. Quiroz, M. Villani, and R. Kohn. Speeding up MCMC by efficient data subsampling. *Pre-print*, 2016. arXiv:1404.4178v3.

R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL `https://www.R-project.org/`.

W. Raghupathi and V. Raghupathi. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, 2(1):3, 2014.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

D. A. Rasmussen, O. Ratmann, and K. Koelle. Inference for nonlinear epidemiological models using genealogies and time series. *PLoS Computational Biology*, 7(8):1–11, 2011.

P. Rebeschini and R. van Handel. Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866, 2015.

C. P. Robert. *The Bayesian choice*. Springer Verlag, 2007.

C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Verlag, 2 edition, 2004.

C. P. Robert and G. Casella. *Introducing Monte Carlo methods with R*. Springer Verlag, 2009.

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 and R. L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

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.

J. Rousseau and K. Mengersen. Asymptotic behaviour of the posterior distribution in overfitted mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(5):689–710, 2011.

T. B. Schön, F. Lindsten, J. Dahlin, J. Wågberg, C. A. Naesseth, A. Svensson, and L. Dai. Sequential Monte Carlo methods for system identification. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 775–786, Beijing, China, October 2015.

M. Segal and E. Weinstein. A new method for evaluating the log-likelihood gradient, the Hessian, and the Fisher information matrix for linear dynamic systems. *IEEE Transactions on Information Theory*, 35(3):682–687, 1989.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

A. Shah, A. G. Wilson, and Z. Ghahramani. Student-t processes as alternatives to Gaussian processes. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 832–840, Reykjavik, Iceland, April 2014.

C. Sherlock, A. H. Thiery, G. O. Roberts, and J. S. Rosenthal. On the efficency of pseudo-marginal random walk Metropolis algorithms. *The Annals of Statistics*, 43(1):238–275, 2015.

R. H. Shumway and D. S. Stoffer. *Time series analysis and its applications*. Springer Verlag, 3 edition, 2011.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the 2012 Conference on Neural Information Processing Systems (NIPS)*, Lake Tahoe, USA, December 2012.

I. M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

H. J. Spaeth, L. Epstein, A. D. Martin, J. A. Segal, T. J. Ruget, and S. C. Benesh. Supreme court database. Version 2016 Release 01, 2016. URL `http://supremecourtdatabase.org`.

J. C. Spall. A stochastic approximation technique for generating maximum likelihood parameter estimates. In *Proceedings of the 6th American Control Conference (ACC)*, pages 1161–1167, Minneapolis, 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.

D. J. Spiegelhalter. Incorporating Bayesian ideas into health-care evaluation. *Statistical Science*, 19(1):156–174, 2004.

Stan Development Team. Stan: A C++ library for probability and sampling, version 2.8.0, 2015. URL `http://mc-stan.org/`.

D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online Bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*, pages 111–120, Madrid, Spain, April 2009.

L. Stewart and P. McCarty, Jr. Use of Bayesian belief networks to fuse continuous and discrete information for target recognition, tracking, and situation assessment. In *Proceedings of SPIE Signal Processing, Sensor Fusion and Target Recognition*, pages 177–185, Orlando, USA, April 1992.

J. Stoer and R. Bulirsch. *Introduction to numerical analysis*. Springer Verlag, 2 edition, 1993.

G. Storvik. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on Signal Processing*, 50(2):281–289, 2002.

A. Svensson, J. Dahlin, and T. B. Schön. Marginalizing Gaussian process hyperparameters using sequential Monte Carlo. In *Proceedings of the 6th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Cancun, Mexico, December 2015.

R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 58(1):267–288, 1996.

L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22 (4):1701–1728, 1994.

M. K. Titsias and N. D. Lawrence. Bayesian quadrature for ratios. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 844–851, Sardinia, Italy, May 2010.

R. S. Tsay. *Analysis of financial time series*. John Wiley & Sons, 2 edition, 2005.

V. Turri, B. Besselink, and K. H. Johansson. Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning. *Pre-print*, 2015. arXiv:1505.00447v1.

Y. Ulker, B. Gunsel, and T. A. Cemgil. Sequential Monte Carlo samplers for Dirichlet process mixtures. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 876–883, Sardinia, Italy, May 2010.

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, 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.

P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. On robust input design for nonlinear dynamical models. *Automatica*, 2016a. (provisionally accepted).

P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. Particle-based Gaussian process optimization for input design in nonlinear dynamical models. *Pre-print*, 2016b. arXiv:1603.05445v1.

A. W. Van der Vaart. *Asymptotic statistics*. Cambridge University Press, 2000.

P. Vandewalle, J. Kovačević, and M. Vetterli. Reproducible research in signal processing. *IEEE Signal Processing Magazine*, 26(3):37–47, 2009.

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, USA, August 1995.

J. Wågberg, F. Lindsten, and T. B. Schön. Bayesian nonparametric identification of piecewise affine ARX systems. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 709–714, Beijing, China, October 2015.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 681–688, Bellevue, USA, July 2011.

N. Whiteley. Stability properties of some particle filters. *The Annals of Applied Probability*, 23(6):2500–2537, 2013.

F. Wood, J. W. van de Meent, and V. Mansinghka. A new approach to probabilistic programming inference. In *Proceedings of the 17th International conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1024–1032, Reykjavik, Iceland, April 2014.

Y. Xie. knitr: a comprehensive tool for reproducible research in R. In V. Stodden, F. Leisch, and R. D. Peng, editors, *Implementing reproducible computational research*. Chapman and Hall/CRC, 2014.

E. P. Xing, M. I. Jordan, and R. Sharan. Bayesian haplotype inference via the Dirichlet process. *Journal of Computational Biology*, 14(3):267–284, 2007.

H. Xue, F. Gu, and X. Hu. Data assimilation using sequential Monte Carlo methods in wildfire spread simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 22(4):23, 2012.

J. Yang, N. A. Zaitlen, M. E. Goddard, P. M. Visscher, and A. L. Price. Advantages and pitfalls in the application of mixed-model association methods. *Nature genetics*, 46(2): 100–106, 2014.

Z. Zhang, E. Ersoz, C-Q. Lai, R. J. Todhunter, H. K. Tiwari, M. A. Gore, P. J. Bradbury, J. Yu, D. K. Arnett, J. M. Ordovas, and E.S Buckler. Mixed linear model approach adapted for genome-wide association studies. *Nature genetics*, 42(4):355–360, 2010.

H. S. Zhou. Modified backward sampling smoothing with EM algorithm - application to economics and finance. *Pre-print*, 2013. Unpublished report.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Part II

Papers

Paper A

Getting started with particle Metropolis-Hastings for inference in non-linear models

Authors: J. Dahlin and T. B. Schön

Edited version of the paper:

J. Dahlin and T. B. Schön. Getting started with particle Metropolis-Hastings for inference in nonlinear models. *Pre-print*, 2015. arXiv:1511.01707v4.

Getting started with particle Metropolis-Hastings for inference in non-linear models

J. Dahlin* and T. B. Schön†

*Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden.
johan.dahlin@liu.se

†Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
thomas.schon@it.uu.se

Abstract

We provide a gentle introduction to the particle Metropolis-Hastings (PMH) algorithm for parameter inference in non-linear state space models (SSMs) together with a software implementation in the statistical programming language R. Throughout this tutorial, we develop an implementation of the PMH algorithm (and the integrated particle filter), which is distributed as the package **pmhtutorial** available from the CRAN repository. Moreover, we provide the reader with some intuition for how the algorithm operates and discuss some solutions to numerical problems that might occur in practice. To illustrate the use of PMH, we consider parameter inference in a linear Gaussian SSM with synthetic data and a non-linear stochastic volatility model with real-world data. We conclude the tutorial by discussing important possible improvements to the algorithm and we also list suitable references for further study.

Keywords

Bayesian inference, state space models, particle filtering, particle Markov chain Monte Carlo.

Data and source code in R, MATLAB and Python

`https://github.com/compops/pmh-tutorial`

Financial support from

The projects *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) and CADICS, a Linnaeus Center, both funded by the Swedish Research Council.

Introductory overview

We are concerned with Bayesian parameter inference in non-linear and/or non-Gaussian state space models (SSMs). This is an important problem as SSMs are ubiquitous in e.g., automatic control (Ljung, 1999), econometrics (Durbin and Koopman, 2012), finance (Tsay, 2005) and many other fields. An overview of some concrete application of state space modelling is given by Langrock (2011). The major problem with Bayesian inference in SSMs is that it is an analytically intractable problem. That is, we cannot obtain closed-form prior-posterior updates by using conjugate priors, which is sometimes possible e.g., the data is independent and identically distributed. Instead, we make use of statistical simulation techniques based on Monte Carlo to create an *empirical approximation* of the posterior distribution. That is, a weighted sum of Dirac atoms that converge to the true posterior as the number of atoms tends to infinity.

In this tutorial, we make use of the particle Metropolis-Hastings (PMH; Andrieu et al., 2010) algorithm to sample from the sought posterior distribution and create the empirical approximation. A requirement for using the PMH algorithm is that we can evaluate the posterior point-wise by an unbiased non-negative estimator. In this tutorial, we employ a particle filter (Gordon et al., 1993; Doucet and Johansen, 2011) to create this unbiased estimator. Note that particle filtering is interesting on its own, which is discussed in the aforementioned references.

The main aim and contribution of this tutorial is to give a gentle introduction (both in terms of the methods and in terms of software) to the PMH algorithm and to the inherent particle filtering. We assume that the reader is familiar with traditional time series analysis and Kalman filtering at the level of Brockwell and Davis (2002). Some familiarity with Monte Carlo approximations, importance sampling and Markov chain Monte Carlo at the level of Ross (2012) is also beneficial. The reader is referred to the aforementioned books for the background material required for this tutorial.

We focus on developing implementable code for the algorithms in two different models. Furthermore, we provide the reader with some implementation tricks to avoid numerical problems that might occur in some models. We also discuss some properties of the algorithms and try to give the reader some intuition of the ideas behind them. The final implementation is available as a R (R Core Team, 2015) package with the name **pmhtutorial** distributed under the GPL-2 license via the CRAN repository. See Section 7.3 for alternative implementations in MATLAB and Python.

We continue this section by introducing the SSM and the parameter and the state inference problems that are solved using the PMH algorithm and the particle filter.

State space models

In Figure 1, we present a graphical model of the SSM with the latent states (upper), the observations (lower) and no input[1]. We note that the latent state denoted by x_t only depends on the previous state x_{t-1} of the process as it is a Markov process (of first order).

[1]It is straightforward to modify the algorithms presented for vector valued quantities and also to include a known exogenous input u_t.

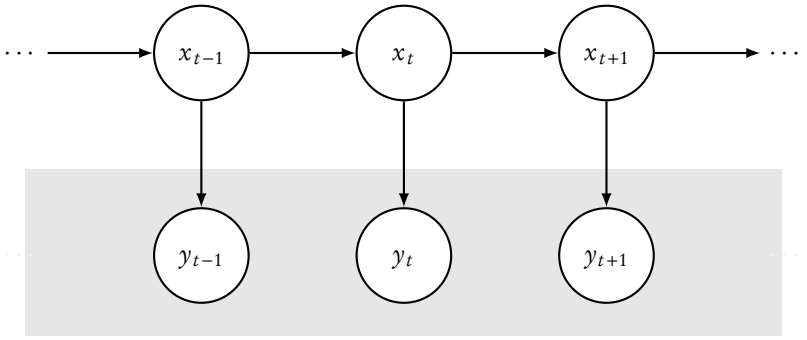


Figure 1. A cartoon of the structure of an SSM using a graphical model.

That is, all the information in the past states $x_{0:t-1}$ is summarised in the most recent state x_{t-1}. Furthermore, the observations are conditionally independent as they only relate to each other through the states.

More specially, we assume that the observations and the states are denoted by $y_{1:T} \triangleq \{y_t\}_{t=1}^T$ and $x_{0:T}$, respectively. Furthermore, we assume that the state and observations are real-valued scalars, i.e., $y_t \in \mathcal{Y} \subseteq \mathbb{R}$ and $x_t \in \mathcal{X} \subseteq \mathbb{R}$. Note that this assumption can be straightforwardly relaxed so that states and observations are vector-valued real numbers. The initial state is assumed to be distributed according to the density $\mu_\theta(x_0)$ parametrised by some unknown static parameters $\theta \in \Theta \subset \mathbb{R}^p$. The latent state and the observation processes are assumed to be governed by the densities[2] $f_\theta(x_t \,|\, x_{t-1})$ and $g_\theta(y_t \,|\, x_t)$, respectively. That is, the density describing the state processes gives the probability that the next state is x_t given the previous state x_{t-1}. The same interpretation holds for the observation process.

With these assumptions in place, we can express a general non-linear SSM by

$$x_0 \sim \mu_\theta(x_0), \qquad x_t \,|\, x_{t-1} \sim f_\theta(x_t \,|\, x_{t-1}), \qquad y_t \,|\, x_t \sim g_\theta(y_t \,|\, x_t), \tag{1}$$

where we make no notational distinction between the random variables and their realisations for brevity.

Bayesian inference

The main objective in the *parameter inference problem* is to obtain an estimate of the parameters θ given the measurements $y_{1:T}$. In this tutorial, we adopt a Bayesian approach to estimate θ by computing the parameter posterior distribution given by

$$\pi_\theta(\theta) \triangleq p(\theta \,|\, y_{1:T}) = \frac{p(\theta)p(y_{1:T} \,|\, \theta)}{\int_\Theta p(\theta')p(y_{1:T} \,|\, \theta')\mathrm{d}\theta'}, \tag{2}$$

[2] In this tutorial, we assume that $x_t \,|\, x_{t-1}$ and $y_t \,|\, x_t$ can be modelled as continuous random variables with a density function. However, the algorithms that we introduce can be applied to deterministic states and discrete states/observations as well.

where $p(\theta)$ and $p(y_{1:T}\,|\,\theta) = p_\theta(y_{1:T})$ denote the *parameter prior distribution* and the *likelihood* of the data, respectively. The denominator is usually referred to as the *marginal likelihood* or the *model evidence*.

A standard approach to estimate (2) is to employ Markov chain Monte Carlo (MCMC; Robert and Casella, 2004) methods. This family of methods constructs a Markov chain to sample from $\pi_\theta(\theta)$. An empirical approximation is then constructed of the parameter posterior by samples from the Markov chain. In this tutorial, we make use of an MCMC algorithm known as the particle Metropolis-Hastings (PMH; Andrieu et al., 2010) algorithm. However, this algorithm requires that we can estimate $p_\theta(y_{1:T})$ point-wise but this is difficult as we do not know the state sequence. The problem appears when we would like to compute the likelihood by the decomposition

$$p_\theta(y_{1:T}) = p_\theta(y_1)\prod_{t=2}^{T} p_\theta(y_t\,|\,y_{1:t-1}). \tag{3}$$

In this expression, the *predictive likelihood* can be computed by

$$p_\theta(y_t\,|\,y_{1:t-1}) = \int g_\theta(y_t\,|\,x_t) f_\theta(x_t\,|\,x_{t-1})\pi_{t-1}(x_{t-1})\mathrm{d}x_t\mathrm{d}x_{t-1}, \tag{4}$$

where $\pi_{t-1}(x_{t-1}) = p_\theta(x_{t-1}\,|\,y_{1:t-1})$ denotes the unknown *marginal filtering distribution*. Hence, we are required to solve a *state inference problem* to estimate the value of the state x_t to be able sample from $\pi_\theta(\theta)$ to form the empirical approximation.

For an SSM, we can compute the marginal filtering distribution using the *Bayesian filtering recursions* given the parameters. As the name suggests, these are recursions that update the estimate of the states as more observations arrive in a Bayesian prior-posterior update. Specifically, we obtain the marginal filtering distribution $\pi_t(x_t)$ as the posterior distribution of the current state given all the observations up until the current time step. However, for most models of interest the recursions cannot be solved in closed form. Instead, we approximate $\pi_t(x_t)$ using a Monte Carlo method known as the *particle filter* (Doucet and Johansen, 2011) or sequential importance sampling with resampling (SIR).

Related software

There are a number of different software packages related to the current tutorial. We focus on some minimal working examples for some toy problems to explain the workings of the algorithms. Other software projects are far more comprehensive and provide platforms for solving the inference problem in more general models. Hence, we view **pmhtutorial** as a gentle introduction to the PMH algorithm after which the reader can move on to more general software to solve larger application specific problems.

The software LibBi (Murray, 2013) provides a platform for Bayesian inference in SSMs using both serial and parallel hardware. It consists of a C++ template library, a parser and a compiler. This allow the user to in a simple manner define new models and solve the parameter inference problem using e.g., PMCMC and SMC^2 (Chopin et al., 2013). Furthermore, interfaces for R and MATLAB are currently under development. Another alternative with the same set-up is Biips (Todeschini et al., 2014).

Two interesting probabilistic programming languages for more multi-purpose statistical inference are Venture (Mansinghka et al., 2014) and Anglican (Wood et al., 2014). Both packages provides functionality for implementing particle filtering and PMCMC for general SSMs. However, Venture is currently under the alpha stage of development and the documentation is therefore currently a bit lacking.

There are also a number of additional software packages that implement particle filtering in SSMs. The software pyParicleEst (Nordh and Berntorp, 2013) is written in Python and provides functionality for state estimate using different types of particle filters. It also includes parameter estimation using Maximum Likelihood via the Expectation Maximisation (EM; Dempster et al. (1977); McLachlan and Krishnan (2008)) algorithm. A C++ template for particle filtering and more general sequential algorithms based on the Feynman-Kac formalism is provided by SMCTC (Johansen, 2009).

Overview of the PMH algorithm

As previously discussed, we approximate the intractable parameter posterior for a general SSM (1) using an empirical approximation based on a Monte Carlo method known as PMH. This algorithm generates samples from $\pi_\theta(\theta)$ by creating a Markov chain with some necessary properties. More specifically, we require the stationary distribution of the Markov chain to be the parameter posterior. This means that we can simulate samples from the posterior by using the Markov chain when it has reached stationarity. We refer to the initial transient phase before stationarity has been reached as the *burn-in*.

Constructing the Markov chain

The construction of the Markov chain consists of two steps. The first step is to propose a so-called *candidate parameter* θ' from a *proposal distribution* $q(\theta' \mid \theta_{k-1})$ given the previous state of the Markov chain denoted θ_{k-1}. The proposal distribution is determined by the user with the requirement that it should at least have a support covering the target distribution.

The second step is to determine if we should change the state to θ' or remain in the current state θ_{k-1}. This is done to facilitate exploration of (in theory) the entire posterior distribution. Also, this mechanism is the necessary condition for the Markov chain to actually have the sought posterior as its stationary distribution. The so-called *acceptance probability* is computed as

$$\alpha(\theta', \theta_{k-1}) = 1 \wedge \frac{\pi_\theta(\theta')}{\pi_\theta(\theta_{k-1})} \frac{q(\theta_{k-1} \mid \theta')}{q(\theta' \mid \theta_{k-1})}, \tag{5}$$

which determines the probability that we assign the candidate parameter as the next state of the Markov chain, i.e., $\{\theta_k \leftarrow \theta'\}$. Here, we introduce the notation $a \wedge b \triangleq \min\{a, b\}$.

The intuition for the acceptance probability (5) (disregarding the influence of the proposal q) is that we always accept a candidate parameter θ' if $\pi_\theta(\theta') > \pi_\theta(\theta_{k-1})$. That is if θ' increases the value of the target compared with the previous state θ_{k-1}. This results in a mode seeking behaviour which is similar to an optimisation algorithm, where we would like to estimate the maximum of the posterior distribution. However from (5), we also

note that we can accept a small decrease in the posterior value to facilitate exploration of the entire posterior. This is the main difference between PMH and that of an optimisation algorithm, where we would like to explore the full posterior in the former and only the mode in the latter.

The resulting K correlated samples $\theta_{1:K} \triangleq \{\theta_k\}_{k=1}^K$ can be used to construct a Monte Carlo approximation of $\pi_\theta(\theta)$. In this case, we can write the *empirical approximation* of the parameter posterior distribution as

$$\widehat{\pi}_\theta^K(\mathrm{d}\theta) = \frac{1}{K}\sum_{k=1}^K \delta_{\theta_k}(\mathrm{d}\theta), \tag{6}$$

which corresponds to a collection of Dirac atoms $\delta_{\theta'}(\mathrm{d}\theta)$ located at $\theta = \theta'$ with equal weights. In practice, we make use of histograms or kernel density estimators to visualise the estimate of the parameter posterior obtained from (6).

In Bayesian parameter inference, we are often interested in computing the expectation of a so-called *test function*, which is a well-behaved (integrable) function $\varphi : \Theta \to \mathbb{R}$, which maps the parameters to a value on the real line. The expectation with respect to parameter posterior is given by

$$\pi_\theta[\varphi] \triangleq \mathbb{E}\Big[\varphi(\theta)\,|\,y_{1:T}\Big] = \int_\Theta \varphi(\theta)\pi(\theta)\,\mathrm{d}\theta, \tag{7}$$

where $\varphi(\theta) = \theta$ corresponds to computing the mean of the parameter posterior. Unfortunately, we have that $\pi_\theta[\varphi]$ is intractable as we do not have access to a closed-form expression for $\pi_\theta(\theta)$. Instead, we can replace it with the empirical approximation in (6) to obtain

$$\widehat{\pi}_\theta^K[\varphi] \triangleq \int_\Theta \varphi(\theta)\widehat{\pi}^K(\mathrm{d}\theta) = \frac{1}{K}\sum_{k=1}^K \int_\Theta \varphi(\theta)\delta_{\theta_k}(\mathrm{d}\theta) = \frac{1}{K}\sum_{k=1}^K \varphi(\theta_k), \tag{8}$$

which follows from the properties of the Dirac delta function (measure). This estimator is well-behaved and it is possible to establish a law of large numbers (LLN) and a central limit theorem (CLT), see Robert and Casella (2004) or Meyn and Tweedie (2009) for more information. From the LLN, we know that the estimator is *consistent* (and asymptotically unbiased as $K \to \infty$).

Moreover from the CLT, we know that the error is approximately Gaussian with a variance that decreases with $1/K$, which is the usual Monte Carlo rate. Note that the LLN usually assumes independent samples but a similar result known as the *ergodic theorem* gives a similar result (under some assumptions) even when $\theta_{1:K}$ are correlated. However, the asymptotic variance is larger than if the samples would be uncorrelated.

Approximating the acceptance probability

The main problem when implementing the PMH algorithm for many SSMs is that we cannot compute the acceptance probability (5). This is the result of the intractability of the posterior as the likelihood cannot be evaluated point-wise. Instead, we make use of a point

estimate of the likelihood given by the particle filter. The resulting estimator is unbiased and non-negative, which turns out to be the necessary condition for this approach to be valid. That is, for the PMH algorithm to still generate samples from and generate a valid empirical approximation of $\pi_\theta(\theta)$. This approach is known as an *exact approximation*, as we approximate the likelihood but still obtain an *exact* algorithm. We return to discussing this family of methods in Section 7.

To estimate the likelihood, we are required to estimate the states as outlined in the decomposition of the likelihood in (3). These states can (in theory) be estimated using the *Bayesian filtering recursions* (Anderson and Moore, 2005) (for fixed parameters θ) by

$$\pi_t(x_t) = \frac{g_\theta(y_t \mid x_t)}{p_\theta(y_t \mid y_{1:t-1})} \int_{\mathcal{X}} f_\theta(x_t \mid x_{t-1}) \pi_{t-1}(x_{t-1}) \, \mathrm{d}x_{t-1}, \qquad \text{for } 0 < t \leq T, \tag{9}$$

with $\pi_0(x_0) = \mu_\theta(x_0)$ as the initialisation. In theory, we can construct a sequence of filtering distributions by iteratively applying (9). However, we cannot carry out this procedure using closed-form expressions for many models of interest.

Instead, we again make use of an empirical approximation of $\pi_t(x_t)$ for each t. To generate samples from the marginal filtering distribution, we apply importance sampling sequentially to estimate the state trajectories by approximating (9). The resulting algorithm is known as the particle filter or SIR. Using the samples generated by the particle filter, we can construct the approximation by

$$\widehat{\pi}_t^N(\mathrm{d}x_t) \triangleq \widehat{p}_\theta^N(\mathrm{d}x_t \mid y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{x_t^{(i)}}(\mathrm{d}x_t), \tag{10}$$

where the *particles* $x_t^{(i)}$ and their corresponding weights $w_t^{(i)}$ constitutes the so-called *particle system* generated during a run of the particle filter.

In Figure 2, we present a cartoon of (10) with the true marginal filtering distribution (green) and the Dirac point masses (orange) that aims to approximate it. The location of the point masses corresponds to the location of the particle $x_t^{(i)}$ and the height is given by the weight of the particle $w_t^{(i)}$. It is possible to prove that the empirical approximation converges to the true distribution when $N \to \infty$ under some regularity assumptions.

As for the parameter inference problem, we are often required to compute an expectation of a well-behaved test function $\varphi : \mathcal{X} \to \mathbb{R}$ with respect to $\pi_t(x_t)$ given by

$$\pi_t[\varphi] \triangleq \mathbb{E}\left[\varphi(x_t) \mid y_{1:t}\right] = \int_{\mathcal{X}} \varphi(x_t) \pi_t(x_t) \, \mathrm{d}x_t,$$

where $\varphi(x_t) = x_t$ corresponds to computing the mean of the marginal filtering distribution. The computation of $\pi_t[\varphi]$ is intractable as we cannot compute π_t in closed form for most SSMs. However, we can approximate this expectation by inserting the empirical

approximation (10). This results in

$$\widehat{\pi}_t^N[\varphi] \triangleq \int_{\mathcal{X}} \varphi(x_t)\widehat{\pi}_t^N(\mathrm{d}x_t) = \sum_{i=1}^{N} w_t^{(i)} \int_{\mathcal{X}} \varphi(x_t)\delta_{x_t^{(i)}}(\mathrm{d}x_t) = \sum_{i=1}^{N} w_t^{(i)} \varphi\left(x_t^{(i)}\right), \tag{11}$$

for some $0 \leq t \leq T$ by the properties of the Dirac delta function. Under some assumptions, the properties of $\widehat{\pi}_t^N[\varphi]$ are similar as for the estimator in the PMH algorithm, see Crisan and Doucet (2002) or Doucet and Johansen (2011) for more information. Hence, we have that the estimator is consistent (and asymptotically unbiased when $N \to \infty$) and the error is Gaussian with a variance that decreases as $1/N$.

We conclude this section by noting that solving the parameter inference problem in an SSM also requires us to solve the state inference problem. As a consequence, we begin the subsequent section by presenting how to implement the particle filter for constructing (10). We then proceed with implementing the PMH algorithm using the output from the particle filter to approximate the acceptance probability (5).

We also discuss how to avoid some issues with numerical precision and to decrease the computational complexity by constructing optimal proposal distributions. In Section 5, we consider a real-world application of the PMH algorithm from finance and highlight some practical problems and more sophisticated improvements in Section 6. Finally, we conclude in Section 7 with an overview of further reading for extending the material covered here.

Estimating the parameters in a linear Gaussian SSM

In this section, we discuss how to estimate the parameter posterior $\pi_\theta(\theta)$ for a linear Gaussian state space (LGSS) model. We consider this model as it is possible to solve the state inference problem exactly using the Kalman filter. That is, we can solve (9) in closed-form using the properties of the Gaussian distribution. Hence, we can make use of the Kalman filter to validate and benchmark the particle filter to learn more about its properties.

The particular LGSS model that we consider is given by

$$x_0 \sim \delta_{x_0}(x), \qquad x_t \,|\, x_{t-1} \sim \mathcal{N}\left(x_t; \phi x_{t-1}, \sigma_v^2\right), \qquad y_t \,|\, x_t \sim \mathcal{N}\left(y_t; x_t, \sigma_e^2\right), \tag{12}$$

where parameters are denoted by $\theta = \{\phi, \sigma_v, \sigma_e\}$. The parameter $\phi \in (-1, 1)$ determines the persistence of the state. The parameters $\sigma_v, \sigma_e \in \mathbb{R}_+$ denote the standard deviations of the state noise and the observation noise, respectively. Here, $\mathcal{N}(x; \mu, \sigma^2)$ denotes the Gaussian distribution of a random variable x with mean μ and standard deviation $\sigma > 0$ and $\delta_{x_0}(x)$ denotes a Dirac mass located at $x = x_0 \in \mathbb{R}$. In Figure 3, we present $T = 250$ simulated data points from this model using $\theta = \{0.95, 0.10, 1.00\}$. Note that, the autocorrelation function (ACF) for $y_{1:T}$ falls off slowly as the persistence parameter ϕ is close to one and σ_v is rather small.

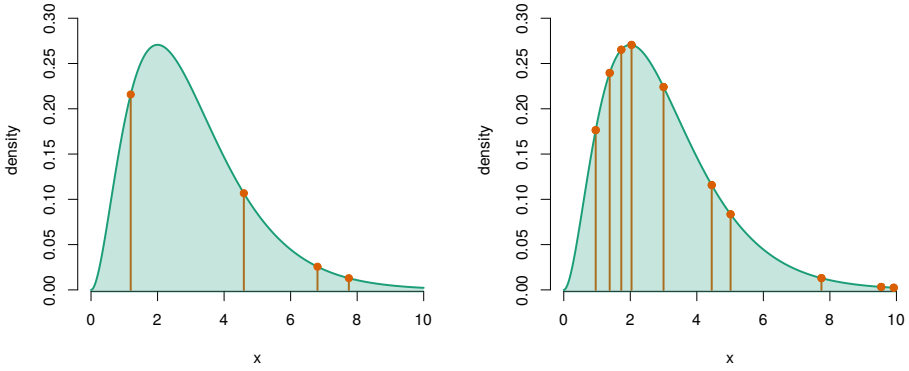


Figure 2. A cartoon of the particle approximation of some distribution (green) given by four (left) and nine (right) particles (orange), respectively.

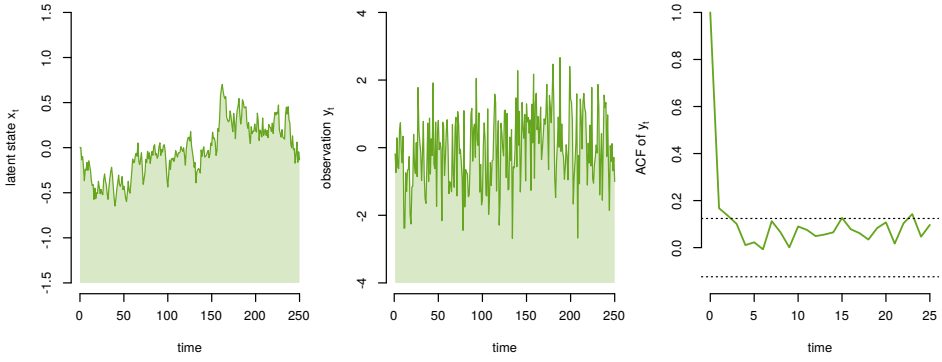


Figure 3. Simulated data from the LGSS model with latent state (left), observations (center) and ACF of the observations (right).

Implementing the particle filter

To implement the PMH algorithm, we are required to also implement a particle filter to estimate the likelihood in the parameter posterior (2). Here, we start out by presenting the particle filter and then make use of it in the PMH algorithm in Section 4.3.

Both algorithms are implemented in R (R Core Team, 2015). For this end, we make use of *literate programming*[3] and the complete code is available in the function `sm` included in the package **pmhtutorial**. The corresponding skeleton for the code is given by:

```
sm <- function(y, par, nPart, T, x0) {
  < initialisation >

  for (tt in 2:T) {
    < resampleParticles >
    < propagateParticles >
    < weightParticles >
    < estimateFilteredStateAndLogLikelihood >
  }
  < returnEstimates >
}
```

The function `sm` has inputs: `y` (vector with T observations), `par` (parameters $\{\phi, \sigma_v, \sigma_e\}$), `nPart` (no. particles), `T` (no. data points) and `x0` (the initial state). Note that, the function updates the particle system and compute the estimates iteratively over the index `tt` starting at 2 and ending with T (corresponding to $t = 1$ to $T - 1$ in (12)).

`< initialisation >` We allocate the variables `p`, `u`, `w`, `xhatf` and `ll` for the particles, their corresponding unnormalised and normalised weights, the filtered state estimates and the log-likelihood estimate, respectively. For the LGSS model (12), we have $\mu_\theta(x_0) = \delta_0(x_0)$ so all the particles are initially set to $x_0 = 0$ and all weights to $1/N$ (as every particle is identical). This operation is carried out by:

```
< initialisation > =
xhatf <- matrix(x0,        nrow=T,     ncol=1)
p     <- matrix(x0,        nrow=nPart, ncol=T+1)
v     <- matrix(1,         nrow=nPart, ncol=T+1)
w     <- matrix(1 / nPart, nrow=nPart, ncol=T+1)
ll    <- 0;
```

`< resampleParticles >` The intuition behind this step is that we focus the computational effort of the algorithm to the relevant part of the state space. That is, we focus on states that are likely to have generated the obtained observation under the model. More specifically, we compute a weight for each particle corresponding to the probability that the specific particle generated the observation under the model. In the resampling, we then randomly duplicate particles with large weights and discard particles with small weights. Note that, the resampling step is unbiased in the sense that the expected proportions of the resampled

[3] In this approach, we start by outlining the skeleton of the code with some variables marked by `< variable >`. In the following paragraphs, we assign code to each of these variables using a C++ like syntax. That is, the assignment operator is denoted by = and += denotes the operation that we add the code after the current code stored in the variable. For more information, see e.g., Section 1.1 in Pharr and Humphreys (2010).

particles are given by the particle weights. This step is important as we otherwise would end up with a single unique particle and a large variance in (11) after a number of iterations.

In our implementation, we make use of *multinomial* resampling, which is also known as a weighted bootstrap with replacement. The output from this procedure are the so-called *ancestor indices* $a_t^{(i)}$ for each particle i, which can be interpreted as the parent index of particle i at time t. For each $i = 1, \ldots, N$, we sample the ancestor index from the multinomial distribution with

$$\mathbb{P}\left[a_t^{(i)} = j\right] = w_{t-1}^{(j)}, \qquad j = 1, \ldots, N.$$

The resampling step is carried out by a call to the function `sample` by:

```
< resampleParticles > =
nIdx <- sample(nPart, nPart, replace = TRUE, prob = w[, tt-1])
```

where the resulting ancestor indices $a_t^{(1:N)}$ are stored in `nIdx`.

`< propagateParticles >` In this step, we simulate the particle system one step forward in time. This is done to generate a number of hypotheses of the current state of the system, which we then can compare to the observation. The simulation step amounts to sampling from some *particle proposal distribution* where we make use of the previous state $x_{t-1}^{a_t^{(i)}}$ and the current measurement y_t to propose new states $x_t^{(i)}$ at time t by

$$x_t^{(i)} \,|\, x_{t-1}^{a_t^{(i)}} \sim p_\theta\left(x_t^{(i)} \,|\, x_{t-1}^{a_t^{(i)}}, y_t\right). \tag{13}$$

In our implementation, we make use of *optimal proposal* for the LGSS model given by

$$\begin{aligned} p_\theta^{\text{opt}}\left(x_t^{(i)} \,|\, x_{t-1}^{a_t^{(i)}}, y_t\right) &\propto g_\theta\left(y_t \,|\, x_t\right) f_\theta\left(x_t \,|\, x_{t-1}^{a_t^{(i)}}\right) \\ &\propto \mathcal{N}\left(x_t^{(i)}; \sigma^2\left[\sigma_e^{-2} y_t + \sigma_v^{-2} \phi x_{t-1}^{a_t^{(i)}}\right], \sigma^2\right), \end{aligned} \tag{14}$$

with $\sigma^{-2} = \sigma_v^{-2} + \sigma_e^{-2}$ which minimises the variance of the incremental particle weights at the current time step[4]. The expression follows from the model and the properties of the Gaussian distribution. The propagation step is carried out by:

```
< propagateParticles > =
Delta   <- ( par[2]^(-2) + par[3]^(-2) )^(-1)
mup     <- par[3]^(-2) * y[tt] + par[2]^(-2) * par[1] * p[nIdx, tt-1]
p[, tt] <- Delta * mup + rnorm(nPart, 0, sqrt(Delta))
```

From (14), we have that the ratio between the noise variances determine the shape of the proposal. Essentially, we have two different extreme situations (a) $\sigma_e^2/\sigma_v^2 \ll 1$ and (b) $\sigma_e^2/\sigma_v^2 \gg 1$. In the first situation (a), the location of the proposal is essentially governed by y_t and the scale is mainly determined by σ_e. This corresponds to essentially simulating particles from $g_\theta(y_t \,|\, x_t)$. When σ_e is small this usually allows for running the particle filter using only a small number of particles. In the second situation (b), we essentially simulate from $f_\theta(x_t \,|\, x_{t-1})$ and do not take the observation into account. In summary, the

[4] However, it is unclear exactly how this influences the entire particle system, i.e., if this is the globally optimal choice.

performance and characteristics of the optimal proposal is therefore related to the noise variances of the model and their relative sizes.

< weightParticles > In this step, we compute the weights required for the resampling step. We make use of the *optimal weighting* function for the LGSS model given by

$$v_t^{(i)} \triangleq p(y_{t+1}|x_t) = \int g_\theta\left(y_{t+1}|x_{t+1}\right) f_\theta\left(x_{t+1}|x_t^{(i)}\right) \mathrm{d}x_{t+1} = \mathcal{N}\left(y_{t+1}; \phi x_t^{(i)}, \sigma_v^2 + \sigma_e^2\right),$$

which follows from the properties of the Gaussian distribution. Hence in this implementation, we make use of the new observation y_{t+1} between the propagation and the weighting steps for the first time. In many situations, the resulting weights are small and it is therefore beneficial to work with shifted log-weights to avoid problems with numerical precision. This is done by applying the transformation

$$\widetilde{v}_t^{(i)} = \log v_t^{(i)} - v_{\max}, \qquad \text{for } i = 1, \ldots, N,$$

where $v_{\max}$ denotes the largest element in $\log v_t^{(1:N)}$. Then we normalise the weights (so that they sum to one) by

$$w_t^{(i)} = \frac{\exp(-v_{\max}) \exp\left(\log v_t^{(i)}\right)}{\exp(-v_{\max}) \sum_{j=1}^N \exp\left(\log v_t^{(j)}\right)} = \frac{\exp\left(\widetilde{v}_t^{(i)}\right)}{\sum_{j=1}^N \exp\left(\widetilde{v}_t^{(j)}\right)}, \tag{15}$$

where the shifts $-v_{\max}$ cancel and does not affect the relative size of the weight. The computation of the weights is carried out by:

```
< weightParticles > =
v[, tt] <- dnorm(y[tt+1], par[0] * p[, tt], sqrt(par[2]^2 + par[3]^2), log = TRUE)
vmax    <- max(v[, tt])
v[, tt] <- exp(v[, tt] - wmax)
w[, tt] <- v[, tt] / sum(v[, tt])
```

We remind the reader that we compare `y[tt+1]` and `p[, tt]` due to the convention for indexing arrays in R, which corresponds to y_t and $x_{t-1}^{(1:N)}$ in the model, respectively. Note also that the weights depends on the next observation and this is the reason for why the for-loop runs over `tt`$= 1, \ldots, T-1$.

< estimateFilteredStateAndLogLikelihood > In this step, we estimate the mean of the filtered state distribution and the log-likelihood at time t. The former can be expressed as $\widehat{\pi}_t^N[x_t]$, i.e., by using $\varphi(x_t) = x_t$ in (11). This results in the estimator[5]

$$\widehat{x}_{t|t}^N = \frac{1}{N} \sum_{i=1}^N x_t^{(i)}, \tag{16}$$

[5]Note that this corresponds to that all weights $w_t^{(i)}$ are equal with value $1/N$. However, this is not the case as we can see from (15). The reason for the slight reformulation of the general estimator (11) is that we make use of the optimal propagation and the optimal weighting steps. This results in a specific implementation of the particle filter known as the fully-adapted particle filter (FAPF; Pitt and Shephard, 1999). In a FAPF, we make use of separate weights for resampling and for constructing the empirical approximation of $\pi_t(x_t)$. However, we refrain from using this more general formulation to keep the presentation simple.

of the filtered state at time t. For the latter, we make use of the logarithm of the decomposition in (3) to obtain

$$\log \widehat{p}^N(y_{1:T}) = \log \widehat{p}^N(y_1) + \sum_{t=2}^{T} \log \widehat{p}^N(y_t \,|\, y_{1:t-1}) = \sum_{t=1}^{T-1} \left\{ \left[\log \sum_{i=1}^{N} v_t^{(i)} \right] - \log N \right\}, \quad (17)$$

where the second equality follows from making use of the particle system to approximate the predictive likelihood (4). That is, making use of the empirical approximation as

$$\widehat{p}^N(y_t \,|\, y_{1:t-1}) = \int_{\mathcal{X}^2} g_\theta(y_t \,|\, x_t) f_\theta(x_t \,|\, x_{t-1}) \widehat{\pi}_{t-1}^N(\mathrm{d}x_{t-1})\, \mathrm{d}x_t,$$

in a similar manner to (11), see Dahlin (2014) for more information. We carry out (16) and (17) by:

```
< estimateFilteredStateAndLogLikelihood > =
xhatf[tt] <- mean(p[, tt])
ll        <- ll + vmax + log(sum(v[, tt])) - log(nPart)
```

`< returnEstimates >` The outputs from `sm` are `xh` (filtered state estimates) and `ll` (estimate of the log-likelihood). This operation is carried by:

```
< returnEstimates > =
output <- list(xh = xhatf, ll = ll)
```

Numerical illustration of state inference

In this section, we make use of the particle filter to estimate the filtered state and to investigate the properties of this estimate for finite N. The complete implementation and code is available in the function `example1_lgss`. We simulate a single realisation from (12) with $T = 250$ observations using the parameters $\theta = \{\phi, \sigma_v, \sigma_e\} = \{0.75, 1.0, 0.1\}$. We present the resulting observations $y_{1:T}$ in upper part of Figure 4.

We generate data using the function `generateData` with the same inputs as for the particle filter. Running this function returns `x` (the latent states $x_{0:T}$) and `y` (the observations $y_{1:T}$). The particle filter can then be executed by a call to the function `sm` with the generated data as an input. This entire operation is carried out by:

```
set.seed(10)
library("pmhtutorial")
d     <- generateData(par=c(0.75, 1.0, 0.10), T=250, x0=0.0)
outPF <- sm(d$y, par=c(0.75, 1.0, 0.10), nPart=100, T=250, x0=0.0)
```

In the middle of Figure 4, we present the difference between the optimal state estimate from the Kalman filter and the estimate from the particle filter using $N = 20$ particles. Two alternative measures of accuracy are the bias (absolute error) and the mean square error (MSE) of the state estimate. These are computed according to

$$\mathsf{Bias}(\widehat{x}_{t|t}^N) = \frac{1}{T} \sum_{t=1}^{T} |\widehat{x}_{t|t}^N - \widehat{x}_{t|t}|, \qquad \mathsf{MSE}(\widehat{x}_{t|t}^N) = \frac{1}{T} \sum_{t=1}^{T} (\widehat{x}_{t|t}^N - \widehat{x}_{t|t})^2,$$

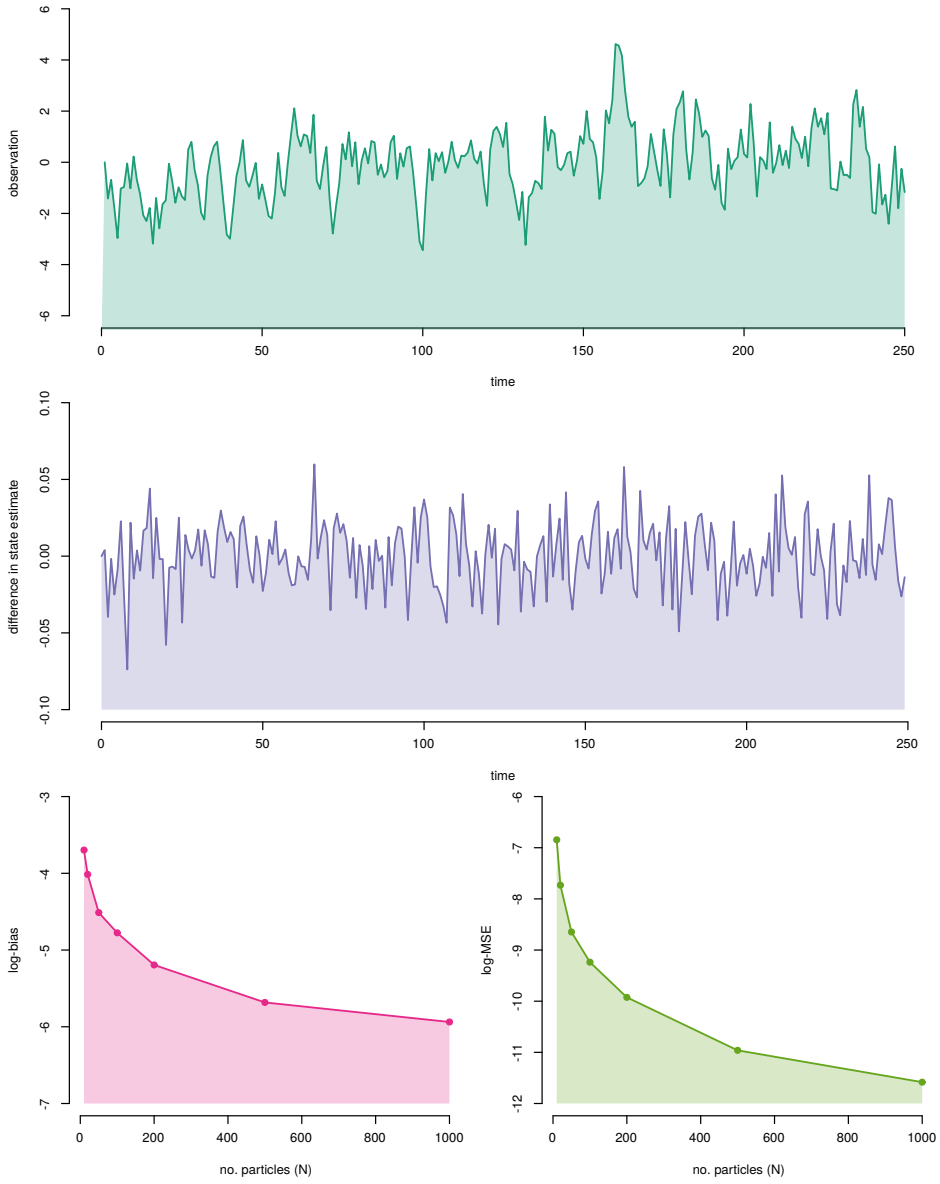


Figure 4. Upper and middle: A simulated set of observations (green) from the LGSS model and the error in the latent state estimate (purple) using a particle filter with $N = 20$. Lower: the estimated log-bias and log-MSE for the particle filter when varying the number of particles N.

N	10	20	50	100	200	500	1000
log-bias	-3.70	-4.01	-4.51	-4.78	-5.19	-5.68	-5.94
log-MSE	-6.84	-7.73	-8.65	-9.24	-9.93	-10.96	-11.58

Table 1. The logarithm of the bias and the MSE of the filtered states while varying N.

where $\widehat{x}_{t|t}$ denotes the optimal state estimate obtained by the Kalman filter. In Table 1 and in the lower part of Figure 4, we present the logarithm of the bias and the MSE for different values of N. We note that the bias and the MSE decrease rapidly when increasing N. Hence, we conclude that for this model N does not need to be large for the estimate to be accurate.

Implementing particle Metropolis-Hastings

We proceed by implementing PMH for sampling from $\pi_\theta(\theta)$ with $\theta = \phi$, which we refer to as the *target distribution* of the Markov chain. We fix the parameters $\{\sigma_v, \sigma_e\}$ to their true values and only aim to infer the posterior of ϕ given the data $y_{1:T}$ denoted by $\pi_\theta(\phi)$. The complete source code is available in the function `pmh` and its skeleton is given by:

```
pmh <- function(y, initPar, par, nPart, T, x0, nIter, stepSize)
{
  < initialisation >
  for (kk in 2:nIter) {
    < proposeParameters >
    < computeAcceptanceProbability >
    < acceptRejectStep >
  }
  output <- th
}
```

The function `pmh` has inputs: `y` (vector with T observations), `initPar` (ϕ_0 the initial value for ϕ), `sigmav,sigmae` (parameters $\{\sigma_v, \sigma_e\}$), `nPart` (no. particles), `T` (no. observations), `x0` (initial state), `nIter` (no. PMH iterations K) and `stepSize` (step length in the proposal). The output from `pmh` is $\theta_{1:K}$, i.e., the correlated samples approximately distributed according to the parameter posterior π_θ.

`< initialisation >` We allocate some variables to store the current state of the Markov chain `th`, the proposed state `thp`, the current log-likelihood `ll` and the proposed likelihood `llp`. Furthermore, we allocate the binary variable `accept` that assumes the value 1 if the proposed parameter is accepted and `0` otherwise. Finally, we run the particle filter with the parameters $\{\phi_0, \sigma_v, \sigma_e\}$ to estimate the initial likelihood. The initialisation is done by:

```
< initialisation > =
th     <- matrix(0, nrow=nIter, ncol=1)
thp    <- matrix(0, nrow=nIter, ncol=1)
ll     <- matrix(0, nrow=nIter, ncol=1)
llp    <- matrix(0, nrow=nIter, ncol=1)
accept <- matrix(0, nrow=nIter, ncol=1)
th[1]  <- initPar
ll[1]  <- sm(y, c(th[1], par[2:3]), nPart, T, xo)$ll
```

< proposeParameters > As discussed in Section 3, we are required to select a parameter proposal distribution to implement the PMH algorithm. A standard choice is a *Gaussian random walk* given by

$$q(\theta' \mid \theta_{k-1}) = \mathcal{N}(\theta'; \theta_{k-1}, \epsilon^2), \tag{18}$$

where $\epsilon > 0$ denotes the step length of the random walk, i.e., the standard deviation of the increment. The proposal step is carried out by:

```
< proposeParameters > =
thp[kk] <- th[kk-1] + stepSize * rnorm(1)
```

< computeAcceptanceProbability > The acceptance probability (5) can be simplified as (18) is *symmetric*, i.e.,

$$q(\theta' \mid \theta_{k-1}) = q(\theta_{k-1} \mid \theta').$$

Finally, we have to make an assumption of the prior distribution of ϕ, which we select as a (unnormalised) truncated Gaussian prior defined by

$$\mathcal{TN}_{[a,b]}(z; \mu, \sigma^2) = \mathbb{I}[a < z < b]\mathcal{N}(z; \mu, \sigma^2),$$

where $\mathbb{I}(s)$ denotes the indicator function that assumes the value one if s is true and zero otherwise. Hence, $\mathcal{TN}_{[a,b]}(z; \mu, \sigma^2)$ denotes a distribution that is Gaussian with mean μ, standard deviation σ in the interval $z \in [a, b]$ and zero otherwise.

For the LGSS model, we make use of $p(\phi) = \mathcal{TN}_{(-1,1)}(\phi; 0, 0.5)$ to ensure that the system is stable (i.e., that the value of x_t is bounded for every t). This is the reason for why we need to check that $|\phi'| < 1$ before running the particle filter to avoid numerical problems. We set the acceptance probability to zero if this stability requirement is not fulfilled. From these choices of prior and proposal, we can rewrite (5) to obtain

$$\alpha(\theta', \theta_{k-1}) = 1 \wedge \log\left[\frac{p(\theta')}{p(\theta_{k-1})}\right] + \log\left[\frac{\widehat{p}^N_{\theta'}(y_{1:T})}{\widehat{p}^N_{\theta_{k-1}}(y_{1:T})}\right],$$

where we make use of the log-likelihood estimates to avoid numerical problems with small values of the likelihood. The computation of the acceptance probability is carried out by:

```
< computeAcceptanceProbability > =
if (abs(thp[kk]) < 1.0) {
  llp[kk]      <- sm(y, c(thp[kk], par[2:3]), nPart, T, x0)$ll
  aprob_prior  <- dnorm(thp[kk], log = TRUE) - dnorm(th[kk-1], log = TRUE)
  aprob_lldiff <- llp[kk] - ll[kk-1]
  aprob        <- exp(aprob_prior + aprob_lldiff)
} else {
  aprob        <- 0
}
```

< acceptRejectStep > Finally, we need to take a decision for accepting or rejecting the proposed parameter. This is done by simulating a uniform random variable ω over $[0, 1]$ by the built-in R command `runif`. We accept θ' if $\omega < \alpha(\theta', \theta_{k-1})$ by storing it and its corresponding log-likelihood as the current state of the Markov chain. Otherwise, we keep

the current values for the state and the log-likelihood from the previous iteration. The accept/reject step is carried out in R by:

```
< acceptRejectStep > =
u <- runif(1)

if (u < aprob) {
  th[kk]     <- thp[kk]
  ll[kk]     <- llp[kk]
  accept[kk] <- 1.0
} else {
  th[kk]     <- th[kk-1]
  ll[kk]     <- ll[kk-1]
  accept[kk] <- 0.0
}
```

Numerical illustration of parameter inference

In this section, we make use of the PMH algorithm to estimate the parameter posterior of ϕ given the data. The complete implementation and code is available in the function `example2_lgss`. We consider the same data and settings for the particle filter as in Section 4.2. Futhermore, we select the proposal step length $\epsilon = 0.10$ and initialise the Markov chain in $\theta_0 = 0.5$. The algorithm is executed for $K = 5,000$ iterations and the first $K_b = 1,000$ iterations are discarded as burn-in. That is, we only make use of the last $4,000$ samples to construct the empirical approximation of the parameter posterior distribution. We can setup and run the algorithm by:

```
set.seed(10)
library("pmhtutorial")

d   <- generateData(par=c(0.75, 1.0, 0.10), T=250, x0=0.0)
res <- pmh(d$y, initPar=0.50, par, nPart=100, T=250, x0 = 0.0,
           nIter=5000, stepSize=0.10)
```

In Figure 5, we present three runs of the PMH algorithm using different step lengths $\epsilon = 0.01$ (left), 0.10 (center) and 0.50 (right). The resulting posterior estimate (upper) are presented as a histogram and a kernel density estimate (solid line). We also plot the state of the Markov chain at each iteration (middle) and the resulting estimate of the ACF (lower). The dotted lines indicate the estimate of the posterior mean $\widehat{\phi} = 0.66$ computed by:

```
mean(res[nBurnIn:nIter])
```

From the ACF, we see that the choice of ϵ influences the correlation in the Markov chain and thereby the variance in the estimates. The tuning of the proposal step length is therefore important to obtain an efficient exploration of the parameter posterior. We return to this problem in Section 6.3.

We note that the parameter estimate differs slightly from the true value 0.75 and that the uncertainty is rather large in the estimate of the parameter posterior. This is due to the relatively small sample size T (and a finite K). From the asymptotic theory of the Bayesian estimator, we know that the posterior mass tends to concentrate around the true parameter as T (and K) increase.

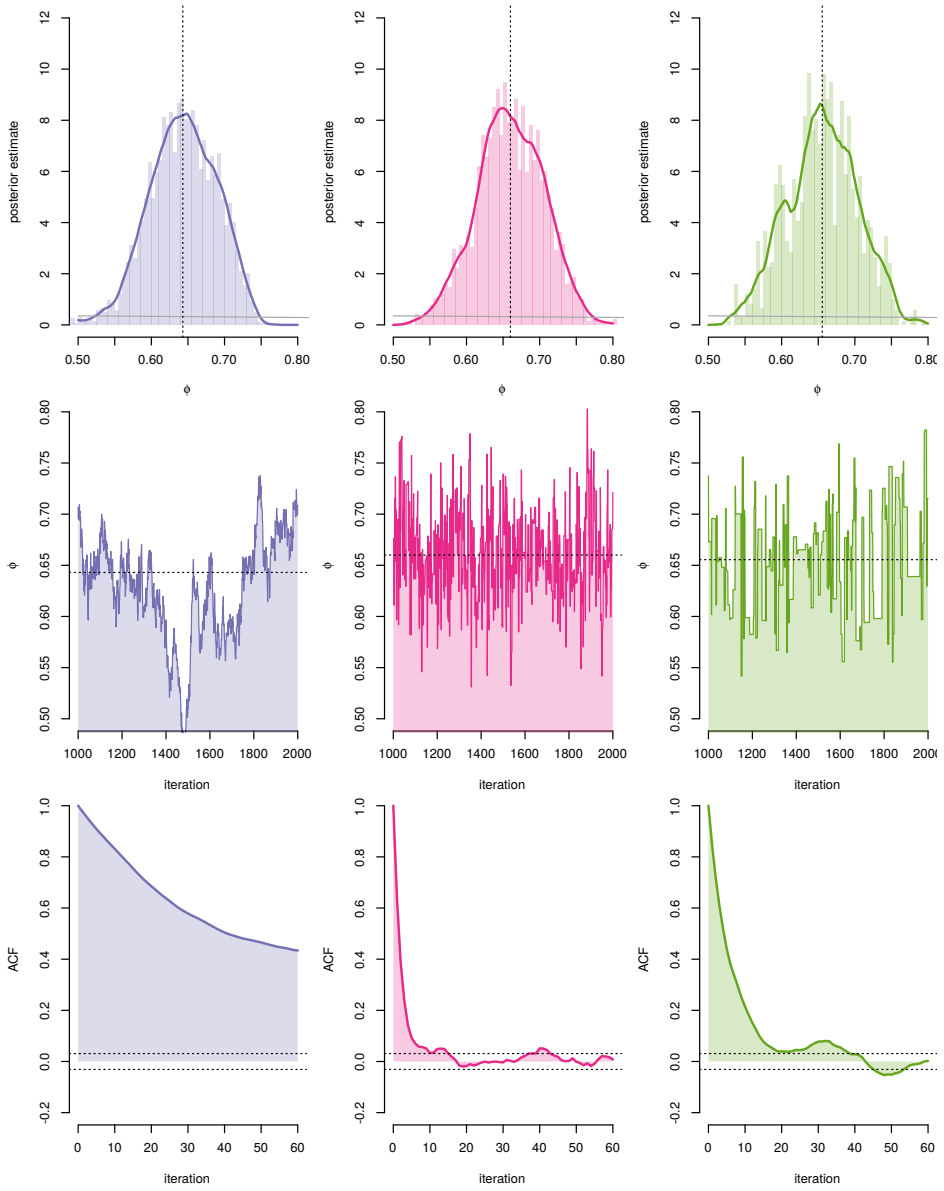


Figure 5. The estimate of $\pi_\theta(\phi)$ in the LGSS model using the PMH algorithm using three different step lengths: $\epsilon = 0.01$ (left), 0.10 (center) and 0.50 (right). Upper: the estimate of π_θ presented as a histogram and kernel density estimate (solid line). Middle: the state of the Markov chain at 1,000 iterations after the burn-in. Lower: the estimated ACF for the Markov chain. Dotted lines in the upper and middle plots indicate the estimate of the posterior mean. The dotted lines in the lower plot indicate the 95% confidence intervals of the ACF coefficients.

Number of observations T	10	20	50	100	200	500
Estimated posterior mean	0.596	0.551	0.581	0.685	0.723	0.737
Estimated posterior variance	0.040	0.025	0.011	0.005	0.005	0.001

Table 2. The estimated posterior mean and variance when varying T.

We exemplify this in Table 2 by estimating the posterior mean and variance using the same setup when T increases. This small study supports that the true parameter is indeed recovered by the posterior mean estimate in the limit. However, the rate of this convergence is determined by the model and therefore it is not possible to give any general guidelines for how large T needs to be to achieve a certain accuracy.

Application example: volatility estimation OMXS30

We continue with a concrete application of the PMH algorithm to infer the parameters of a stochastic volatility (SV; Hull and White, 1987) model. This is a non-linear SSM with Gaussian noise and inference in this type of model is an important problem as the log-volatility (the latent state in this model) is useful for risk management and to price various financial contracts. See e.g., Tsay (2005) and Hull (2009) for more information.

A particular parametrisation of the SV model is given by

$$x_0 \sim \mathcal{N}\Big(x_0; \mu, \frac{\sigma_v^2}{1-\phi^2}\Big), \tag{19a}$$

$$x_{t+1} \,|\, x_t \sim \mathcal{N}\big(x_{t+1}; \mu + \phi(x_t - \mu), \sigma_v^2\big), \tag{19b}$$

$$y_t \,|\, x_t \sim \mathcal{N}\big(y_t; 0, \exp(x_t)\big), \tag{19c}$$

where the parameters are denoted by $\theta = \{\mu, \phi, \sigma_v\}$. Here, $\mu \in \mathbb{R}$, $\phi \in [-1, 1]$ and $\sigma_v \in \mathbb{R}_+$ denote the mean value, the persistence and standard deviation of the state process, respectively. Note that this model is quite similar to the LGSS model, but here the state x_t scales the variance of the observation noise. Hence, we have Gaussian observations with zero mean and a state dependent standard deviation given by $\exp(x_t/2)$.

In econometrics, volatility is another word for standard deviation and therefore we refer to x_t as the *log-volatility*. The measurements in this model y_t are so-called *log-returns*,

$$y_t = 100 \log\left[\frac{s_t}{s_{t-1}}\right] = 100\big[\log(s_t) - \log(s_{t-1})\big],$$

where s_t denotes the price of some financial asset (e.g., an index, stock or commodity) at time t. Here, we consider $\{s_t\}_{t=1}^T$ to be daily closing prices of the NASDAQ OMXS30 index, i.e., a weighted average of the 30 most traded stocks at the Stockholm stock exchange. We extract the data from Quandl[6] for the period between January 2, 2012 and January 2, 2014. The resulting log-returns are presented in the upper part of Figure 6. Note the varying persistent volatility in the log-returns, i.e., periods of small and large variations. This is

[6]The data is available for download from: `https://www.quandl.com/data/NASDAQOMX/OMXS30`.

known as the *volatility clustering* effect and is one of the features of real-world data that SV models aim to capture. From (19) and the example in Figure 3, we note that this can be achieved when $|\phi|$ is close to one and when σ_v is small.

The objectives in this application are to estimate the parameters θ and to estimate the log-volatility $x_{0:T}$ from the observed data $y_{1:T}$. We can estimate both quantities using the PMH algorithm as we obtain both samples from the posterior of the parameter and the state at each iteration of the algorithm. To complete the SV model, we assume some priors for the parameters based on domain knowledge of usual ranges of the parameters, i.e.,

$$p(\mu) = \mathcal{N}(\mu; 0, 1), \quad p(\phi) = \mathcal{TN}_{[-1,1]}(\phi; 0.95, 0.05^2), \quad p(\sigma_v) = \mathcal{G}(\sigma_v; 2, 10).$$

Here, $\mathcal{G}(a, b)$ denotes a Gamma distribution with shape a and scale b, i.e., with expected value a/b.

We need to adapt the code for the particle filter and for the PMH algorithm to this new model. We outline the necessary modifications by replacing parts of the code in the skeleton for the two algorithms. The resulting implementations and source codes are found in the functions `sm_sv` and `pmh_sv`, respectively. In the particle filter, we need to modify all steps except the resampling. In the initialisation, we need to simulate the initial particle system from $\mu_\theta(x_0)$ by adding

```
< initialisation > +=
p[, 1]     <- rnorm(nPart, mu, par[3] / sqrt(1 - par[2]^2))
xhatf[, 1] <- mean(p[, 1])
```

To implement the particle filter for the SV model, we need to choose a (particle) proposal distribution and the weighting function. However, we cannot find the optimal choices as for the LGSS model, so instead the state dynamics is used as the proposal (13) given by

$$x_t^{(i)} \,|\, x_{t-1}^{a_t^{(i)}} \sim f_\theta\left(x_t \,|\, x_{t-1}^{a_t^{(i)}}\right) = \mathcal{N}\left(x_t; \mu + \phi\left(x_{t-1}^{a_t^{(i)}} - \mu\right), \sigma_v^2\right),$$

and the observation model as a weighing function by

$$W_t^{(i)} = g_\theta\left(y_t \,|\, x_t^{(i)}\right) = \mathcal{N}\left(y_t; 0, \exp\left(x_t^{(i)}\right)\right).$$

These two choices result in that we must change the estimator for $x_{t|t} = \pi_t[x_t]$ to

$$\widehat{x}_{t|t}^N = \sum_{i=1}^N w_t^{(i)} x_t^{(i)},$$

where the weights are normalised according to (15). We refer the interested reader to Doucet and Johansen (2011) or to Dahlin (2014) for details. These three changes to the particle filter are carried out by replacing:

```
< propagateParticles > =
p[, tt] <- par[1] + par[2] * (p[nIdx, tt-1] - par[1]) + par[3] * rnorm(nPart)

< weightParticles  > =
v[, tt]  <- dnorm(y[tt-1], 0, exp(0.5 * p[, tt]), log = TRUE)
vmax     <- max(v[, tt])
```

```
v[, tt]  <- exp(v[, tt] - wmax)
w[, tt]  <- v[,tt] / sum(v[,tt])

< estimateFilteredStateAndLogLikelihood > =
zhatf[tt] <- sum(w[, tt] * p[, tt])
ll        <- ll + wmax + log(sum(v[, tt])) - log(nPart)
```

We also need to generalise the PMH code to have more than one parameter. This is straightforward and we refer the reader to the source code for the necessary changes. The major change is to turn `th` and `thp` into matrices instead of vectors and to modify the parameter proposal. In this model, we make use of a multivariate Gaussian proposal centred around the previous parameters θ_{k-1} with covariance matrix $\Sigma = \mathsf{diag}(\epsilon)$, where ϵ now denotes a vector with three elements. The complete implementation and code is available in the function `example3_sv`.

We can load the data and the settings as well as run the PMH algorithm by executing:

```
library("pmhtutorial")

y   <- as.numeric(read.table("omxs30data.csv")[,1 ])
res <- pmh(y, initPar=c(0, 0.9, 0.2), nPart=500, T=500, nIter=7500,
           stepSize=diag(c(0.10, 0.01, 0.05)^2))
```

The estimate of the log-volatility is obtained by a marginalisation approach, see Andrieu et al. (2010) for the details. In this approach, we average the state estimate over the parameter posterior. This is done by modifying the code for the particle filter. After a run of the algorithm, we sample a single trajectory by sampling a particle at time T with probability given by $w_T^{(1:N)}$. We then follow the ancestor lineage back to $t = 0$ and extract the corresponding path in the state space.

To implement this, we introduce the ancestor index variable explicitly in the code and resample the ancestry during each iteration. This operation is carried out by changing the code for the particle filter in the function `sm_sv` by adding:

```
< initialisation > +=
a <- matrix(0, nrow=nPart, ncol=T+1);

< resampleParticles  > +=
a[, 1:tt-1]  <- a[idx, 1:tt-1]
a[, tt]      <- idx
```

We can then sample the state trajectory by replacing:

```
< returnEstimates > =
nIdx   <- sample(nPart, 1, prob=w[, T])
xhatf  <- p[ cbind(a[nIdx,], 1:(T+1)) ]
output <- list(xh = xhatf, ll = ll)
```

in the function `sm_sv`.

The sampled state trajectory in `xhatf` is then treated in the same manner as the candidate parameter in the PMH algorithm. Hence, we can compute the posterior mean of the parameters and the log-volatility and its corresponding standard deviation by:

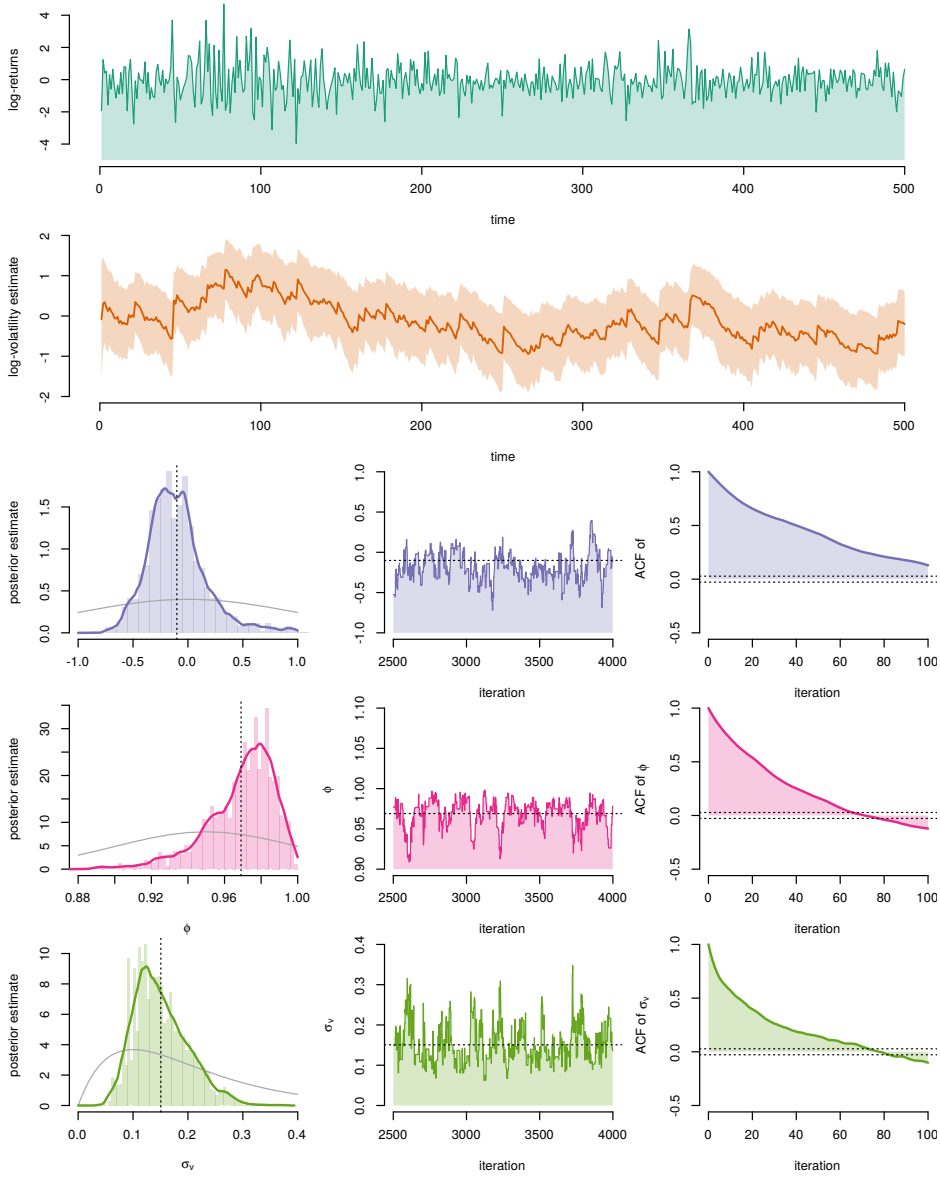


Figure 6. Upper: the daily log-returns (dark green) and estimated log-volatility (orange) with 95% confidence intervals of the NASDAQ OMXS30 index for the period January 2, 2012 to January 2, 2014. Lower: the posterior estimate (left), the trace of the Markov chain (middle) and the corresponding ACF (right) for μ (purple), ϕ (magenta) and σ_v (green) obtained from PMH. Dotted and gray lines in the left and middle plots indicate the parameter posterior mean and the parameter priors, respectively.

```
thhat   <- colMeans(res$thhat[ 2500:7500,])
thhatSD <- apply(res$thhat[2500:7500,], 2, sd)

xhat    <- colMeans(res$xhat[ 2500:7500,])
xhatSD  <- apply(res$xhat[2500:7500,], 2, sd)
```

The resulting posterior estimates, traces and ACFs are presented in Figure 6. We see that the chain and posterior are clearly concentrated around the posterior mean estimate $\widehat{\theta} = \{-0.10, 0.97, 0.15\}$ with standard deviations $\{0.27, 0.02, 0.05\}$. This confirms our belief that the log-volatility is a slowly varying process as ϕ is close to one and σ_v is small.

We also compute the estimate of the log-volatility given this parameter and present it in the second row of Figure 6. We note that the volatility is larger around $t = 100$ and $t = 370$, which corresponds well to what is seen in the log-returns.

Improving the PMH algorithm

In this section, we outline some possible improvements and *best practices* for the PMH algorithm applied to estimate the SV model in Section 5. The material in this section is more advanced than in the previous and can be omitted during a first reading. We discuss initialisation, convergence diagnostics and how to improve the so-called mixing of the constructed Markov chain. For the latter, we consider two different approaches: tuning the random walk proposal and reparameterising the model.

Initialisation

It is important to initialise the Markov chain in areas of high posterior probability mass to obtain an efficient algorithm. In theory, we can initialise the chain anywhere in the parameter space and still obtain a convergent Markov chain. However, in practice we can experience numerical difficulties when the parameter posterior assumes small values or is relatively insensitive to changes in θ. Therefore it is advisable to try to obtain an approximate estimate of the parameters and initialise closer to the posterior mode.

In the LGSS model, we can make use of a quasi-Newton optimisation algorithm (Nocedal and Wright, 2006) in combination with a Kalman filter to estimate the mode of the proposal and extract a Hessian estimate of the log-posterior around that mode. This information can be used directly to tailor a Gaussian random walk proposal. We initialise the PMH algorithm at the mode and set $\Sigma = -\widehat{\mathcal{P}}^{-1}$, where $\widehat{\mathcal{P}}$ denotes the estimate of the Hessian obtained from the quasi-Newton optimisation.

However, this is not possible for general SSMs as we can only obtain noisy estimates of the posterior distribution by the particle filter. Therefore, quasi-Newton algorithms can get stuck in local minima or fail to converge. This is due to the fact that the gradient estimates are noisy and line search algorithms often fail. Good approaches for initialisations of the PMH algorithm for non-linear SSMs is an open research question and we provide references to some attempts to solve this in Section 7.

Diagnosing convergence

It is in general difficult to prove that the Markov chain has reached its stationary regime and that the samples obtained are actually samples from $\pi_\theta(\theta)$. A simple solution is to initialise the algorithm at different points in the parameter space and compare the resulting posterior estimates. If they are approximately the same (after discarding the burn-in), we conclude that the Markov chains have reach the stationary phase.

Another alternative is to make use of the Kolmogorov-Smirnov (KS; Massey, 1951) test to establish that the approximation of the posterior distribution does not change after the burn-in. This is done by dividing the samples obtained from the PMH algorithm $\{\theta_k\}_{k=1}^K$ into three partitions: the burn-in and two sets of equal number of samples from the stationary phase. We thin the Markov chain to obtain uncorrelated samples by extracting every lth sample from the two sets of samples from the stationary phase. We then conduct a standard KS test to compare the corresponding posterior estimates. The Markov chain is in the stationary regime, if the test does not reject the null hypothesis that the data is identically distributed. There also exist other methods for diagnosing convergence, see Robert and Casella (2009) and Gelman et al. (2013).

Improving mixing

Mixing is an important concept for the PMH algorithm and is associated with the variance of the estimates obtained from the algorithm. We can see this from the central limit theorem (CLT) for the estimator $\widehat{\pi}_\theta^K[\varphi]$ in (8) given by

$$\sqrt{K}\left(\pi_\theta[\varphi] - \widehat{\pi}_\theta^K[\varphi]\right) \xrightarrow{d} \mathcal{N}\left(0, \mathbb{V}_\pi[\varphi] \cdot \mathsf{IACT}(\theta_{1:K})\right), \qquad K \to \infty,$$

when $\mathbb{V}_\pi[\varphi] = \pi_\theta[(\varphi - \pi_\theta[\varphi])^2] < \infty$. Here, $\mathsf{IACT}(\theta_{1:K})$ denotes the integrated autocorrelation time (IACT) of the Markov chain, which is computed as the area under the ACF. Note that the CLT is established under some assumptions discussed by e.g., Meyn and Tweedie (2009) and Robert and Casella (2004).

Intuitively, we can view the IACT as the *price that we have to pay* in variance for using correlated samples to estimate the expectation in (8). An interpretation of the IACT is that it represents the number of iterations of the PMH algorithm between two uncorrelated samples. Hence, an IACT of one indicates that the samples are completely uncorrelated as is the case in an importance sampler. Minimising the IACT is therefore an important objective when implementing a computationally efficient algorithm. We refer to that the chain is *mixing well* or *mixing bad* depending on the IACT.

Tuning the proposal

In the middle of Figure 6, we present the ACF for the three parameters in the SV model. We can make use of this information to compute the corresponding IACTs by

$$\mathsf{IACT}(\theta_{1:K}) = 1 + 2\sum_{\tau=1}^{\infty} \rho_\tau(\theta_{1:K}),$$

where $\rho_\tau = \mathbb{E}[(\theta_k - \pi_\theta[\theta])(\theta_{k+\tau} - \pi_\theta[\theta])]/\pi_\theta[(\theta - \pi_\theta[\theta])^2$ denotes the autocorrelation coefficient at lag τ for $\theta_{1:K}$. In practice, we cannot compute the IACT exactly as we do not know the autocorrelation coefficients for all possible lags. Also, it is difficult to estimate ρ_τ when τ is large and K is rather small. A possible solution to this problem is to cut off the ACF estimate at some lag and only consider lags smaller than this limit L. In this tutorial, we make use of $L = 100$ lags to estimate the IACT by

$$\widehat{\mathsf{IACT}}(\theta_{1:K}) = 1 + 2\sum_{\tau=1}^{100} \widehat{\rho}_\tau^K(\theta_{1:K}),$$

where $\widehat{\rho}_\tau^K = \mathsf{corr}(\varphi(\theta_k), \varphi(\theta_{k+\tau}))$ denotes the estimate of the lag-τ autocorrelation of φ. When we apply this estimator for the SV model in Section 5, we obtain the IACT $\{91, 50, 42\}$ for each of the three parameters.

To obtain a better proposal, we can tune the proposal by the unknown covariance of the posterior distribution as discussed in connection with the initialisation problem. An estimate of the so-called *pre-conditioning* matrix $\mathcal{P}$ can be computed using the sample from a pilot run of the PMH algorithm by:

```
resTh  <- res$thhat[nBurnIn:nIter,]
estCov <- var(resTh)
```

If we make use of the trace from the run in Section 5, we obtain the covariance estimate

$$\widehat{\mathcal{P}} = 10^{-4}\begin{bmatrix} 726 & 10 & -11 \\ 10 & 3 & -7 \\ -11 & -7 & 24 \end{bmatrix}.$$

Using this we can form a new improved proposal by

$$\theta' \sim \mathcal{N}\left(\theta'; \theta_{k-1}, \epsilon^2\widehat{\mathcal{P}}\right), \tag{20}$$

where $\epsilon^2 = 0.8$ is selected using trial-and-error (pilot runs) to obtain an acceptance rate of about 35%.

In Figure 7, we present the marginal posteriors (in color) for each pair of parameters in (19). These posteriors are estimated using a 2-dimensional kernel density estimate (KDE) and indicated by contour lines at a number of posterior levels. We also compare the original and the new proposal (20) distributions. Both proposals are centred around $\theta_{k-1} = \widehat{\theta}$, i.e., the estimated posterior mean. We note that the new proposal fits the posterior better as there shapes are more similar than for the original proposal. This improvement in the matching of the proposal to the posterior is expected to improve the mixing of the Markov chain.

In Figure 8, we present the results of the same implementation as in the previous section with the new proposal. The complete implementation and code is available in the function `example4_sv`. The resulting IACT estimates (computed in analogue with the above) are $\{28, 22, 25\}$. That is, we obtain clear improvements in the mixing of the Markov chain for μ and σ_v. The consequence is that we can cut the number of iterations (and therefore computational cost) by $91/28 = 3.25$ and still obtain the same variance in the estimates.

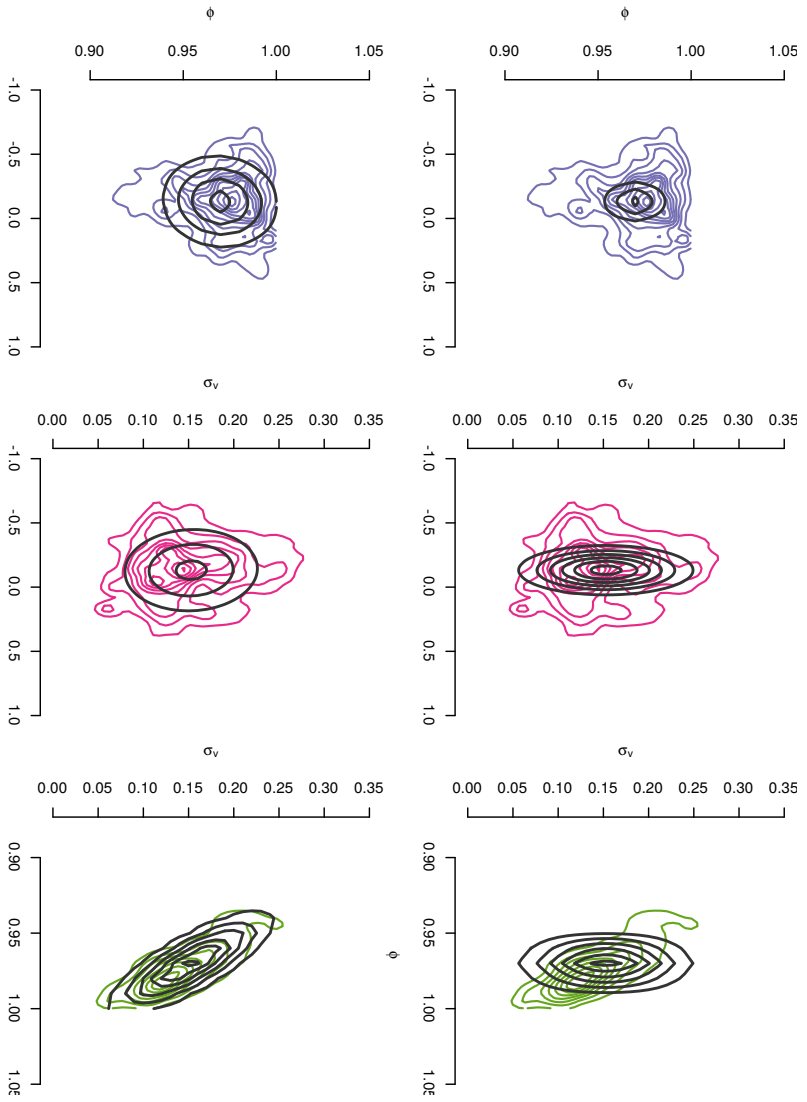


Figure 7. The estimated marginal posterior for μ and ϕ (purple), μ and σ_v (magenta) and ϕ and σ_v (green) obtained from PMH. The dark contours (upper/lower) indicate the original proposal from Section 5 and the new (improved) proposal estimated from the pilot run, respectively. Both proposals are centred at the posterior mean estimate.

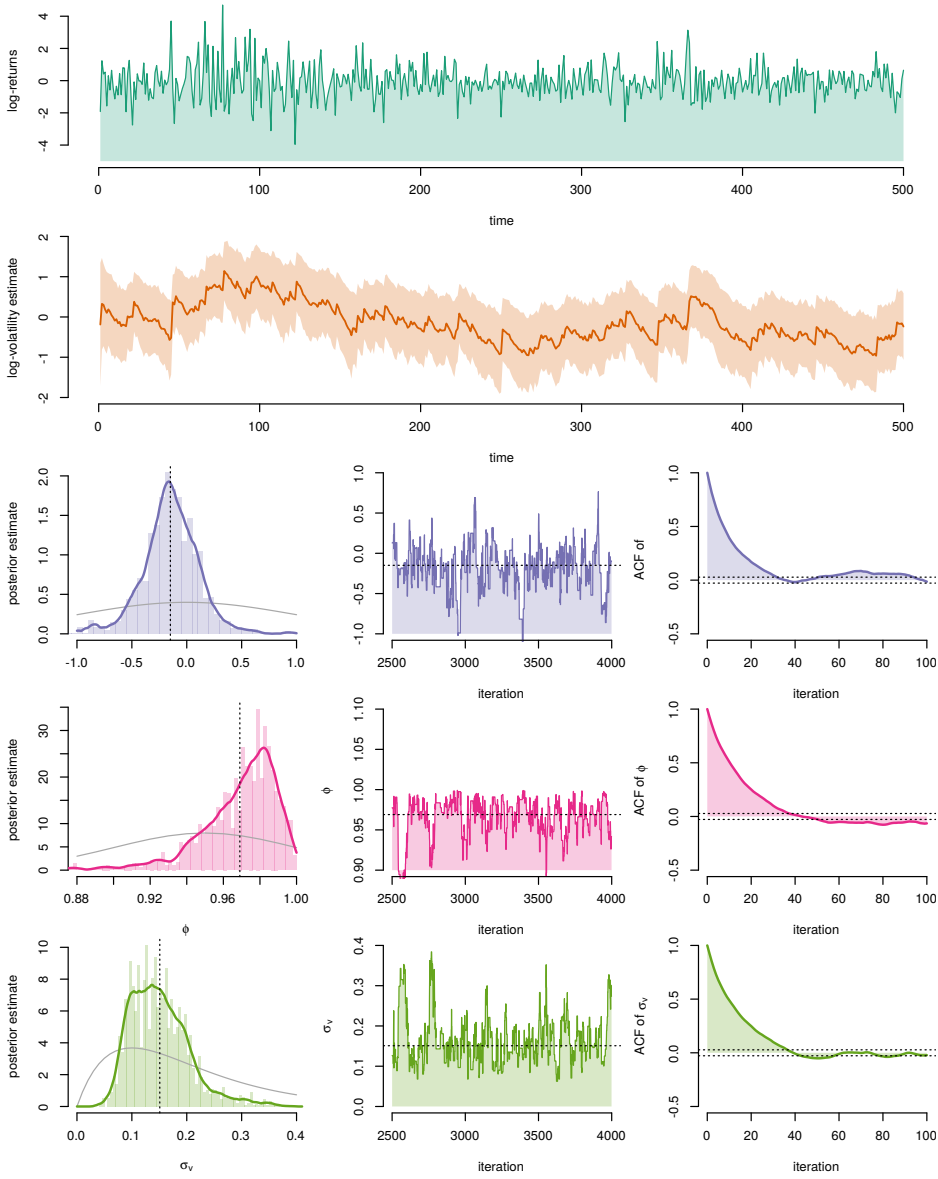


Figure 8. Upper: the daily log-returns (dark green) and estimated log-volatility (orange) with 95% confidence intervals of the NASDAQ OMXS30 index for the period January 2, 2012 and January 2, 2014. Lower: the posterior estimate (left), the trace of the Markov chain (middle) and the corresponding ACF (right) for μ (purple), ϕ (magenta) and σ_v (green) obtained from PMH. Dotted and gray lines in the left and middle plots indicate the parameter posterior mean and the parameter priors, respectively.

Reparametrisating the model

Another approach to improve mixing is to reparameterise the model to obtain unconstrained parameters, which can assume any value on the real line. In our model, a problem is the parameter ϕ, which is constrained to the region $|\phi| < 1$ to obtain a stable SSM. This results in poor mixing if we propose many candidate parameters such that $|\phi'| > 1$. Because any such candidate parameter leads to a reject decision in the PMH algorithm and this increases the autocorrelation. Also, for the same reason it is beneficial to constrain the standard deviation σ_v of the process noise to be positive.

To mitigate the problem with constrained parameters, we consider a reparameterisation of the model given by

$$\phi = \tanh(\psi), \qquad \sigma_v = \exp(\varsigma), \tag{21}$$

such that $\psi, \varsigma \in \mathbb{R}$ are unconstrained parameters. Hence, we change the target of the PMH algorithm to the parameter posterior of $\vartheta = \{\mu, \psi, \varsigma\}$. To retain a valid algorithm that still targets the original parameter posterior, we need to compensate for this transformation in the acceptance probability. This is done by taking the Jacobians of (21) into account, which are given by

$$\frac{\partial}{\partial \psi} \tanh^{-1}(\psi) = \frac{1}{1-\psi^2}, \qquad \frac{\partial}{\partial \varsigma} \log(\varsigma) = \varsigma^{-1}. \tag{22}$$

The resulting acceptance probability is calculated according to

$$\alpha(\vartheta', \vartheta_{k-1}) = 1 \wedge \frac{p(\theta')}{p(\theta_{k-1})} \frac{\widehat{p}^N_{\vartheta'}(y_{1:T} \mid u')}{\widehat{p}^N_{\vartheta_{k-1}}(y_{1:T} \mid u_{k-1})} \left| \frac{1-(\phi')^2}{1-\phi^2_{k-1}} \right| \left| \frac{\sigma'_v}{\sigma_{v,k-1}} \right|, \tag{23}$$

where we make use of the original parameters to compute the prior and to estimate the likelihood. Hence, the proposal operates on ϑ to propose a new candidate parameter ϑ'. We then use the transformation (22) to obtain θ', which we make use of to estimate the likelihood and to compute the acceptance probability (23). After a run of this implementation, we recover the samples from the original posterior by transforming the trace of the Markov chain by (22).

To implement this, we need change the call to the particle filter and the computation of the acceptance probability. The complete implementation and code is available in the function `example5_sv`. The resulting mean estimate of the parameter posterior is $\{-0.12, 0.96, 0.17\}$ with standard deviation $\{0.28, 0.02, 0.05\}$ and IACT $\{18, 28, 26\}$.

Outlook and conclusions

We have introduced the PMH algorithm for Bayesian parameter inference in non-linear SSMs. The particle filter plays an important role in the PMH algorithm and provides an non-negative unbiased estimator of the likelihood. Furthermore, we have applied the PMH algorithm for inference in an LGSS model and a SV model using both synthetic and real-world data. We have identified that initialisation and tuning of the parameter proposal are important problems for the user. The complete code developed in this tutorial is available in the R-package **pmhtutorial** and can be seen as a compilation of minimal working examples

of how to implement the particle filter and the PMH algorithm. Hopefully, these code snippets can be of use for the interested reader as a starting point to develop his/her own implementations of the algorithms.

We devote this section to give some references for important improvements and further studies of the PMH algorithm and the particle filter. A good start for learning more about the particle filter is the surveys by Doucet and Johansen (2011) and Cappé et al. (2007). They discuss a more general form of the particle filter and related algorithms for solving the filtering problem in SSMs. The parameter inference problem is surveyed by Kantas et al. (2015) and Schön et al. (2015). They discuss both Bayesian and maximum likelihood inference methods in SSMs based on the particle filter and related algorithms.

Improving the particle filter

In this tutorial, we introduced two different types of particle filters: the bootstrap particle filter (BPF) in Section 5 and FAPF in Section 4.1. For small and relatively simple models, the BPF performs well but often performs poorly (or is computationally costly) in many real-world problems with noisy observations, non-linear models or a large dimension of the state vector. Some common improvements are to: (i) consider better proposals and weighting functions, (ii) use other resampling schemes and (iii) combine the particle and Kalman filters.

A possible improvement to BPF is the FAPF introduced by Pitt and Shephard (1999) discussed in Section 4.1. As previously stated, this algorithm makes use of the observations in the resampling and propagation steps. In the cases when FAPF can be implemented, it can result in a significant decrease in the MSE of the estimates and therefore decrease the number of particles required. However, we need to be able to sample from $p(x_{t+1} \mid x_t, y_{t+1})$ and evaluate $p(y_{t+1} \mid x_t)$ to be able to implement FAPF. This is not possible in many cases although Gaussian approximations can be used instead, see Doucet et al. (2001) and Pitt et al. (2012). However, these methods rely on quasi-Newton optimisation that can be computationally prohibitive if N is large.

Another approach is to use another particle filter to approximate the fully adapted proposal, resulting in a nested construction where one particle filter is used within another particle filter to construct a proposal distribution for that particle filter. The resulting construction is referred to as nested sequential Monte Carlo (SMC), see Naesseth et al. (2015) for details. The nested SMC construction makes it possible to consider state spaces of significantly higher state dimension compared to what the BPF can handle.

It is also possible to make use of a mixture of proposals and weighting functions in the particle filter as discussed by Kronander and Schön (2014). This type of filters are based on multiple importance sampling, which is commonly used in e.g., computer graphics. Alternative resampling schemes can also be useful in decreasing the MSE, see Hol et al. (2006) and Douc and Cappé (2005) for some comparisons.

The resampling step can also be modified to give smooth (or continuous) estimates of the likelihood as presented by Malik and Pitt (2011) and Pitt et al. (2014). This results in that standard optimisation approaches can be used to obtain the maximum a-posteriori (MAP)

estimate. However, fixing the random numbers (which is required in this type of filters) can introduce a bias in the parameter estimates.

Another possible improvement is the combination of Kalman and particle filtering, which is possible if the model is conditionally linear and Gaussian in some of its states. The idea is then to make use of Kalman filtering to estimate these states and particle filtering for the remaining states while keeping the linear ones fixed to their Kalman estimates. These types of models are common in engineering and Rao-Blackwellisation schemes like this can lead to a substantial decrease in variance. For more information see e.g., Doucet et al. (2000), Chen and Liu (2000) and Schön et al. (2005).

Finally, note that particle filtering is an instance of SMC methods (Del Moral et al., 2006). SMC is a large class of importance sampling based algorithms that can be applied for many interesting problems in statistics. In principle, SMC can be applied to any problem that can be solved using MCMC methods. In particular, SMC can be used in a sequential or online setting, which makes it an attractive alternative or complement to MCMC.

Improving particle Metropolis-Hastings

As previously discussed, the PMH algorithm is a member of the family of exact approximation or pseudo-marginal algorithms introduced by Andrieu and Roberts (2009). Furthermore, this framework can be used to introduce particle MCMC (PMCMC) algorithms as presented by Andrieu et al. (2010). These papers are useful resources for the reader that is interested in the proofs of the validity of the PMCMC algorithms and their properties.

The particle filter plays an important role in the PMH algorithm and greatly influences the mixing of the Markov chain. If the log-likelihood estimates are noisy (too small N in the particle filter), the chain tends to get stuck for several iterations and this leads to bad mixing. This is the result of that sometimes $p_{\theta'}(y_{1:T}) \ll \widehat{p}^N_{\theta'}(y_{1:T})$, which is due to the noise in the estimator. Thus, balancing N and K to obtain good mixing at a reasonable computational cost is an important problem in the PHM algorithm. A small N might result in that we need to take K large and vice verse. This problem is investigated by Pitt et al. (2012) and Doucet et al. (2015). A simple rule-of-thumb is to select N such that the standard deviation of the log-likelihood estimates is around one.

Another aspect of the algorithm that influences the mixing is the choice of proposal. In this tutorial, we considered a marginal proposal, which we usually refer to as PMH0. This proposal is studied by Sherlock et al. (2015) and the authors offer a rule-of-thumb for tuning the step length of the proposal to take into account the (unknown) posterior covariance.

However, we typically encounter problems when the number of parameters p grows over about 5 or when the chain is initialised far from the areas of high posterior probability. In these cases, we often experience poor mixing of the Markov chain, which requires a large number of iterations K to produce accurate estimates.

To mitigate this problem, it can be useful to take geometrical information about the posterior into account. Girolami and Calderhead (2011) show how to make use of the gradient and the Hessian of the log-posterior to guide the Markov chain to areas of high posterior probability. In the paper, this is motivated by diffusion processes on Riemann manifolds.

However, a perhaps simpler analogy is found in optimisation, where we can make use of noisy gradient ascent or Newton updates in the proposal. Gradient information is useful to guide the Markov chain towards the area of interest during the burn-in and also to keep it in this area after the burn-in phase. The Hessian information can be used to rescale the parameter posterior to make it more isotropic, which greatly simplifies sampling.

In Dahlin et al. (2015a), the authors show how to make use of this type of proposals in the PMH algorithm. We refer to the proposal that makes use of only gradient information as PMH1 (for first-order PMH). The proposal that makes use of both gradient and Hessian information is referred to as PMH2 (for second-order). The challenge here is to obtain good estimates of the gradient and Hessian, which both are analytically intractable for a non-linear SSM. Furthermore, the computation of these quantities usually requires the use of a particle smoother with a high computational cost, which is prohibitive inside the PMH algorithm.

To mitigate this problem, they propose to make use of the faster but more inaccurate fixed-lag particle smoother and instead regularise the Hessian estimate when it is non-positive definite. In Dahlin et al. (2015b), the authors describe a quasi-Newton PMH2 proposal (QPMH2) based on a noisy quasi-Newton update that does not require any Hessian information but constructs a local approximation of the Hessian based on gradient information. PMH1 and similar algorithms are theoretically studied by Nemeth et al. (2014), which offers a rule-of-thumb to tune the step lengths based on an estimate of the posterior covariance.

Another approach to decrease the variance of the estimates from PMH is to modify the test function. Some preliminary work by Mira et al. (2013) and Papamarkou et al. (2014) make use of this in the case when the log-likelihood is available in closed-form. In these papers, the authors make use of a so-called *zero-variance* post-processing step, which removes some of the variance in the estimates. It is straightforward to make use of the same approach for PMH1 and PMH2 as it only requires information about the gradient of the log-posterior. The required information can be collected during the run of the PMH algorithm and applied analogously to Mira et al. (2013).

Finally, we return to the problem of initialising the PMH algorithm. The main problem is that many optimisation methods encounter problems with noisy evaluations of the posterior and its gradients. To mitigate this problem, Dahlin et al. (2015c) propose a method based on Gaussian process optimisation (GPO; Boyle, 2007, Lizotte, 2008), where we optimise a smooth surrogate of the posterior. The predictive distribution of a Gaussian process is used as the surrogate, which is constructed using the noisy estimates of the posterior distribution obtained from a particle filter. Another alternative approach is the simultaneous perturbation and stochastic approximation (SPSA; Spall, 1987) algorithm, which can be applied in combination with the particle filter to estimate the posterior mode. A drawback with SPSA compared with GPO is that it does not provide an estimate of the Hessian to help tuning the proposal.

Additional software implementations

The code developed in this tutorial is available for R via the package **pmhtutorial** from CRAN. However, similar code for MATLAB and Python is available from GitHub at: `https:`

`//github.com/compops/pmh-tutorial`. Make sure to read the `README.md` file at GitHub for more information and to fix the dependencies required for running the code.

Acknowledgements

The authors would like to thank Christian Andersson Naesseth, Wilfried Bonou, Manon Kok, Joel Kronander, Fredrik Lindsten, Andreas Svensson and Patricio Valuenzula for comments and suggestions that greatly improved this tutorial.

Bibliography

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, 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.

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

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

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.

R. Chen and J. S. Liu. Mixture Kalman filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(3):493–508, 2000.

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.

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. *Sequential Monte Carlo for inference in nonlinear state space models*. Licentiate's thesis no. 1652, Linköping University, May 2014.

J. Dahlin and T. B. Schön. Getting started with particle Metropolis-Hastings for inference in nonlinear models. *Pre-print*, 2015. arXiv:1511.01707v4.

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015a.

J. Dahlin, F. Lindsten, and T. B. Schön. Quasi-Newton particle Metropolis-Hastings. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 981–986, Beijing, China, October 2015b.

J. Dahlin, M. Villani, and T. B. Schön. Efficient approximate Bayesian inference for models with intractable likelihoods. *Pre-print*, 2015c. arXiv:1506.06975v1.

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

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.

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, N. de Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 176–183, Stanford, USA, July 2000.

A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer Verlag, 2001.

A. Doucet, M. K. Pitt, G. Deligiannidis, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2): 295–313, 2015.

J. Durbin and S. J. Koopman. *Time series analysis by state space methods*. Oxford University Press, 2 edition, 2012.

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.

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.

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. 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.

A. Johansen. SMCTC: Sequential Monte Carlo in C++. *Journal of Statistical Software*, 30 (1):1–41, 2009.

N. Kantas, A. Doucet, S.S. Singh, J.M. Maciejowski, and N. Chopin. On particle methods for parameter estimation in general state-space models. *Statistical Science*, 30(3):328–351, 2015.

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.

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.

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

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

S. Malik and M. K. Pitt. Particle filters for continuous likelihood evaluation and maximisation. *Journal of Econometrics*, 165(2):190–209, 2011.

V. K. Mansinghka, D. Selsam, and Y. N. Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *Pre-print*, 2014. arXiv:1404.0099.

F. J. Massey, Jr. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley-Interscience, 2 edition, 2008.

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

A. Mira, R. Solgi, and D. Imparato. Zero variance Markov chain Monte Carlo for Bayesian estimators. *Statistics and Computing*, 23(5):653–662, 2013.

L. M. Murray. Bayesian state-space modelling on high-performance hardware using LibBi. *Pre-print*, 2013. arXiv:1306.3277.

C. A. Naesseth, F. Lindsten, and T. B. Schön. Nested sequential Monte Carlo methods. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, July 2015.

C. Nemeth, C. Sherlock, and P. Fearnhead. Particle Metropolis adjusted Langevin algorithms. *Pre-print*, 2014. arXiv:1412.7299v1.

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

J. Nordh and K. Berntorp. pyParticleEst - a Python framework for particle based estimation. Technical Report LUTFD2/TFRT–7628–SE, Department of Automatic Control, Lund, Sweden, March 2013.

T. Papamarkou, A. Mira, and M. Girolami. Zero variance differential geometric Markov chain Monte Carlo algorithms. *Bayesian Analysis*, 9(1):97–128, 2014.

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

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.

M. K. Pitt, S. Malik, and A. Doucet. Simulated likelihood inference for stochastic volatility models using continuous particle filtering. *Journal of Econometrics*, 66(3):527–552, 2014.

R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL `https://www.R-project.org/`.

C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Verlag, 2 edition, 2004.

C. P. Robert and G. Casella. *Introducing Monte Carlo methods with R*. Springer Verlag, 2009.

S. M. Ross. *Simulation*. Academic Press, 5 edition, 2012.

T. Schön, F. Gustafsson, and P.-J. Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7):2279–2289, July 2005.

T. B. Schön, F. Lindsten, J. Dahlin, J. Wågberg, C. A. Naesseth, A. Svensson, and L. Dai. Sequential Monte Carlo methods for system identification. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 775–786, Beijing, China, October 2015.

C. Sherlock, A. H. Thiery, G. O. Roberts, and J. S. Rosenthal. On the efficency of pseudo-marginal random walk Metropolis algorithms. *The Annals of Statistics*, 43(1):238–275, 2015.

J. C. Spall. A stochastic approximation technique for generating maximum likelihood parameter estimates. In *Proceedings of the 6th American Control Conference (ACC)*, pages 1161–1167, Minneapolis, USA, June 1987.

A. Todeschini, F. Caron, M. Fuentes, P. Legrand, and P. Del Moral. Biips: software for Bayesian inference with interacting particle systems. *Pre-print*, 2014. arXiv:1412.3779.

R. S. Tsay. *Analysis of financial time series*. John Wiley & Sons, 2 edition, 2005.

F. Wood, J. W. van de Meent, and V. Mansinghka. A new approach to probabilistic programming inference. In *Proceedings of the 17th International conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1024–1032, Reykjavik, Iceland, April 2014.

Paper B

Particle Metropolis-Hastings using gradient and Hessian information

Authors: J. Dahlin, F. Lindsten and T. B. Schön

This paper is published by Springer:
http://dx.doi.org/10.1007/s11222-014-9510-0.

Edited version of the paper:

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015b.

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.

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@liu.se

†Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
fredrik.lindsten@it.uu.se
thomas.schon@it.uu.se

Abstract

Particle Metropolis-Hastings (PMH) allows for Bayesian parameter inference in non-linear state space models by combining Markov chain Monte Carlo (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 a number of 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 by-product 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.

Keywords

Sequential Monte Carlo, Particle Markov chain Monte Carlo, Manifold MALA, Fixed-lag particle smoothing, Parameter Inference.

Data and source code in Python

`https://github.com/compops/pmh-stco2015`

Financial support from

The projects *Learning of complex dynamical systems* (Contract number: 637-2014-466), *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) and CADICS, a Linnaeus Center, all funded by the Swedish Research Council.

Introduction

We are interested in Bayesian parameter inference in non-linear state space models (SSMs) 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 $x_{0:T} = x_{0:T} \triangleq \{x_t\}_{t=0}^T$ and $y_{1:T} = y_{1:T}$, 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 some distribution $\mu(x_0)$ which, for notational simplicity, is assumed to be independent of θ.

The aim of Bayesian parameter inference is to compute the *parameter posterior distribution*

$$p(\theta \,|\, y_{1:T}) = \frac{p_\theta(y_{1:T})p(\theta)}{p(y_{1:T})}, \tag{2}$$

where $p(\theta)$ denotes the prior of θ and $p_\theta(y_{1:T})$ denotes the likelihood, which for an SSM can be expressed as

$$p_\theta(y_{1:T}) = p_\theta(y_1) \prod_{t=2}^{T} p_\theta(y_t \,|\, y_{1:t-1}). \tag{3}$$

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) (Doucet and Johansen, 2011; Del Moral, 2004) and these can be used as *plug-in estimators*. This is especially useful in the Metropolis-Hastings (MH) algorithm that can be used for estimating the parameter posterior in (2).

This combination of MH and SMC is known as the particle Metropolis-Hastings (PMH; Andrieu et al., 2010) algorithm. The MH acceptance probability depends on the intractable likelihood, which in PMH is estimated using SMC (see Section 2). Despite the apparent approximation, this results in an algorithm that targets the correct posterior distribution (Andrieu et al., 2010). 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. The theoretical properties of the marginal PMH algorithm have been analysed in Andrieu and Vihola (2015); Pitt et al. (2012); Doucet et al. (2015) and it has been applied for a number of interesting applications in, e.g., economics, social network analysis and ecology (Flury and Shephard, 2011; Everitt, 2012; Golightly and Wilkinson, 2011).

In this paper, we show that information such as the gradient and the Hessian about the posterior can be included in the construction of the PMH proposal. This idea is first suggested by Doucet et al. (2011) in the discussions following Girolami and Calderhead (2011). In two previous proceedings, we have applied and extended this idea with gradient information (Dahlin et al., 2013) and also using Hessian information (Dahlin et al., 2014). The present article builds upon and extends this preliminary work. A PMH method using gradient information similar to Dahlin et al. (2013) has recently been proposed by Nemeth and Fearnhead (2014) and Nemeth et al. (2014).

In the context of MH sampling, it has been recognised that the gradient and Hessian can be 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, which intuitively guides the Markov chain to regions of high posterior probability. In the manifold MALA (MMALA; Girolami and Calderhead, 2011), the Hessian (or some other appropriate metric tensor) is also included to scale the proposal to 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, scaling the proposal with the Hessian can considerably simplify the tedious tuning of the method since it removes the need for running pilot runs, which are commonly used to tune the covariance matrices of the random walk MH and the MALA.

In our problem, i.e., for inference in a non-linear SSM (1), the gradient and Hessian 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); Doucet et al. (2013). This provides us with the tools necessary 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 (FL) particle smoother (Kitagawa and Sato, 2001) to estimate the gradient and Hessian. The motivation for this is that this smoother only makes use of the weighted particles computed by the particle filter. Consequently, we obtain this information as a *by-product* of the likelihood computation in the PMH algorithm. This results in only a small computational overhead for the proposed methods when compared to the marginal method.

Finally, we provide numerical experiments to illustrate the benefits of using the gradient and Hessian and the accuracy of the FL 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 distribution is non-isotropic, i.e., when the variances differs between dimensions.

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 also outline how this can be used to construct the proposed PMH1 and PMH2 algorithms.

MH sampling with unbiased likelihoods

The MH algorithm (see, e.g., Robert and Casella (2004)) 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 as its unique stationary distribution.

The algorithm consists of two steps during iteration k: (i) a new parameter θ' is sampled from a proposal distribution $q(\theta' \mid \theta_{k-1})$ given the current state θ_{k-1} and (ii) the current parameter is changed to θ' with probability $\alpha(\theta', \theta_{k-1})$, otherwise the chain remains at the current state. The acceptance probability is given by

$$\alpha(\theta', \theta_{k-1}) = 1 \wedge \frac{\pi(\theta')}{\pi(\theta_{k-1})} \frac{q(\theta_{k-1} \mid \theta')}{q(\theta' \mid \theta_{k-1})}, \tag{4}$$

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., $\pi(\theta) = p(\theta \mid y_{1:T})$. This implies that the acceptance probability (4) will depend explicitly on the intractable likelihood $p_\theta(y_{1:T})$, preventing direct application of the MH algorithm to this problem. However, this difficulty can be circumvented by using a *pseudo-marginal* approach (Beaumont, 2003; Andrieu and Roberts, 2009).

Assume that there exists an unbiased, non-negative estimator of the likelihood $\widehat{p}_\theta(y_{1:T} \mid u)$. We introduce explicitly the random variable $u \in \mathcal{U}$ used to construct this estimator, and we let $m_\theta(u)$ denote the probability density of u on $\mathcal{U}$. The pseudo-marginal method is then a standard MH algorithm operating in a non-standard extended space $\Theta \times \mathcal{U}$, with the *extended target*

$$\begin{aligned} \pi(\theta, u \mid y_{1:T}) &= \frac{\widehat{p}_\theta(y_{1:T} \mid u) m_\theta(u) p(\theta)}{p(y_{1:T})} \\ &= \frac{\widehat{p}_\theta(y_{1:T} \mid u) m_\theta(u) p(\theta \mid y_{1:T})}{p_\theta(y_{1:T})}, \end{aligned}$$

and proposal distribution $m_{\theta'}(u')q(\theta' \mid \theta_{k-1})$. Since the likelihood estimator is unbiased, $\mathbb{E}_{u|\theta}[\widehat{p}_\theta(y_{1:T} \mid u)] = p_\theta(y_{1:T})$, it follows that the extended target admits $p(\theta \mid y_{1:T})$ as a marginal. Hence, by simulating from the extended target $\pi(\theta, u \mid y_{1:T})$ we also obtain samples from the original target distribution $p(\theta \mid y_{1:T})$.

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 algorithm. However, these random variables carry useful information, not only about the likelihood, but also about the geometry of the posterior distribution. We suggest to incorporate this information into the proposal construction. With (θ_{k-1}, u_{k-1}) being the current state of the Markov chain we simulate $\theta' \sim q(\cdot \mid \theta_{k-1}, u_{k-1})$ and $u' \sim m_{\theta'}(\cdot)$, using some proposal q (see Section 2.2).

It follows that the MH acceptance probability for the extended target is given by

$$\begin{aligned} \alpha(\theta', u', \theta_{k-1}, u_{k-1}) &= 1 \wedge \frac{\widehat{p}_{\theta'}(y_{1:T} \mid u') m_{\theta'}(u') p(\theta')}{\widehat{p}_{\theta_{k-1}}(y_{1:T} \mid u_{k-1}) m_{\theta_{k-1}}(u_{k-1}) p(\theta_{k-1})} \frac{m_{\theta_{k-1}}(u_{k-1}) q(\theta_{k-1} \mid \theta', u')}{m_{\theta'}(u') q(\theta' \mid \theta_{k-1}, u_{k-1})} \\ &= 1 \wedge \frac{\widehat{p}_{\theta'}(y_{1:T} \mid u') p(\theta')}{\widehat{p}_{\theta_{k-1}}(y_{1:T} \mid u_{k-1}) p(\theta_{k-1})} \frac{q(\theta_{k-1} \mid \theta', u')}{q(\theta' \mid \theta_{k-1}, u_{k-1})}. \end{aligned} \tag{5}$$

Note that $q(\theta' \mid \theta_{k-1}, u_{k-1})$ may depend on the auxiliary variable u_{k-1} in a (formally) arbitrary way. In particular, in Section 3 we propose a construction making use of *biased* estimates of the gradient and Hessian of the log-posterior. Nevertheless, expression (5) still

defines a correct MH acceptance probability for the extended target, ensuring the validity of our approach. Note also that the aforementioned proposal construction opens up for a wide range of adapted proposals, possibly different from the ones considered here.

Constructing PMH1 and PMH2

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_{k-1}. Hence, consider a second-order Taylor expansion of $\log p(\theta' \mid y_{1:T})$ at θ_{k-1}:

$$\begin{aligned}\log p(\theta' \mid y_{1:T}) \approx{}& \log p(\theta_{k-1} \mid y_{1:T}) \\ &+ (\theta' - \theta_{k-1})^\top \left[\nabla \log p(\theta \mid y_{1:T})\right]_{\theta=\theta_{k-1}} \\ &+ \frac{1}{2}(\theta' - \theta_{k-1})^\top \left[\nabla^2 \log p(\theta \mid y_{1:T})\right]_{\theta=\theta_{k-1}} (\theta' - \theta_{k-1}).\end{aligned}$$

Taking the exponential of both sides and completing the square, we obtain

$$p(\theta' \mid y_{1:T}) \approx \mathcal{N}\Big(\theta'; \theta_{k-1} + \mathcal{H}_T^{-1}(\theta_{k-1})\mathcal{G}_T(\theta_{k-1}), \mathcal{H}_T^{-1}(\theta_{k-1})\Big),$$

where we have introduced the notation $\mathcal{G}_T(\theta_{k-1}) = \nabla \log p(\theta \mid y_{1:T})|_{\theta=\theta_{k-1}}$ for the gradient of the log-posterior and $\mathcal{H}_T(\theta_{k-1}) = -\nabla^2 \log p(\theta \mid y_{1:T})|_{\theta=\theta_{k-1}}$ for the negative Hessian of the log-posterior. Here, we assume for now that the negative Hessian is positive definite; see Section 3.5 for further discussion on this matter.

As pointed out above, these quantities cannot be computed in closed form, but they can be estimated from the random variable u_{k-1} (see Section 3). This suggests three different versions of the PMH algorithm, each resulting from a specific choice of the proposal:

$$q(\theta' \mid \theta_{k-1}) = \begin{cases} \mathcal{N}(\theta_{k-1}, \Sigma), & \text{[PMH0]} \\ \mathcal{N}\left(\theta_{k-1} + \frac{1}{2}\Sigma\widehat{\mathcal{G}}(\theta_{k-1}), \Sigma\right), & \text{[PMH1]} \\ \mathcal{N}\left(\theta_{k-1} + \frac{1}{2}\Sigma\widehat{\mathcal{H}}^{-1}(\theta')\widehat{\mathcal{G}}(\theta_{k-1}), \frac{1}{2}\Sigma\widehat{\mathcal{H}}^{-1}(\theta')\right), & \text{[PMH2]} \end{cases} \tag{6}$$

where we omit the dependence on u for brevity. Here, Σ denotes a scaling matrix that controls the step lengths of the proposal. For PMH0 and PMH1, Σ can be chosen as the inverse of 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 final PMH2 algorithm is presented in Algorithm 1. It makes use of Algorithm 2, described in Section 3, 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. Note that, while the algorithm make explicit reference to the auxiliary variable u, it only depends on this variable through the estimates $\widehat{p}_{\theta_{k-1}}(y_{1:T})$, $\widehat{\mathcal{G}}(\theta_{k-1})$ and $\widehat{\mathcal{H}}(\theta_{k-1})$.

Algorithm 1 Second-order particle Metropolis-Hastings

Inputs: Algorithm 2. $K > 0$ (no. MCMC steps), θ_0 (initial parameters), ϵ (step length).
Output: $\theta = \{\theta_1, \ldots, \theta_K\}$ (samples from the posterior).

1: Run Algorithm 2 to obtain $\widehat{p}_{\theta_0}(y_{1:T})$, $\widehat{\mathcal{G}}(\theta_0)$ and $\widehat{\mathcal{H}}(\theta_0)$.
2: **for** $k = 1$ to K **do**
3: Sample $\theta' \sim q(\theta' | \theta_{k-1}, u_{k-1})$ by (6), $\widehat{\mathcal{G}}(\theta_{k-1})$ and $\widehat{\mathcal{H}}(\theta_{k-1})$.
4: Run Algorithm 2 to obtain $\widehat{p}_{\theta'}(y_{1:T})$, $\widehat{\mathcal{G}}(\theta')$ and $\widehat{\mathcal{H}}(\theta')$.
5: Sample ω_k uniformly over $[0, 1]$.
6: **if** $\omega_k < \alpha(\theta', u', \theta_{k-1}, u_{k-1})$ given by (5) **then**
7: $\theta_k \leftarrow \theta'$. {Accept the parameter}
8: $\{\widehat{p}_{\theta_k}(y_{1:T}), \widehat{\mathcal{G}}(\theta_k), \widehat{\mathcal{H}}(\theta_k)\} \leftarrow \{\widehat{p}_{\theta'}(y_{1:T}), \widehat{\mathcal{G}}(\theta'), \widehat{\mathcal{H}}(\theta')\}$.
9: **else**
10: $\theta_k \leftarrow \theta_{k-1}$. {Reject the parameter}
11: $\{\widehat{p}_{\theta_k}(y_{1:T}), \widehat{\mathcal{G}}(\theta_k), \widehat{\mathcal{H}}(\theta_k)\} \leftarrow \{\widehat{p}_{\theta_{k-1}}(y_{1:T}), \widehat{\mathcal{G}}(\theta_{k-1}), \widehat{\mathcal{H}}(\theta_{k-1})\}$.
12: **end if**
13: **end for**

Properties of the PMH1 and PMH2 proposals

In the sequel, we use a single step size $\Sigma = \epsilon^2 \mathbf{I}_d$ for all the parameters in the (standard) 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 (Nocedal and Wright, 2006, Chapter 3). That is, the proposal is invariant to affine transformations of the parameters. 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., 2015), 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 taking the local curvature into account.

Estimation of the likelihood, gradient, and Hessian

In this section, we show how to estimate the likelihood together with the gradient and Hessian using SMC methods.

Auxiliary particle filter

An auxiliary particle filter (APF) (Pitt and Shephard, 1999) can be used to approximate the sequence of joint smoothing distributions (JSDs) $p_\theta(x_{1:t} \mid y_{1:t})$ 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 JSD at time t by

$$\widehat{p}_\theta(\mathrm{d}x_{1:t} \mid y_{1:t}) \triangleq \sum_{i=1}^{N} \frac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}} \delta_{x_{1:t}^{(i)}}(\mathrm{d}x_{1:t}). \tag{7}$$

Here, $\delta_z(\mathrm{d}x_{1:t})$ denotes the Dirac measure placed at z. The particle system is propagated from $t-1$ to t by first sampling an *ancestor index* $a_t^{(i)}$, with

$$\mathbb{P}(a_t^{(i)} = j) = \nu_{t-1}^{(j)} \left[\sum_{k=1}^{N} \nu_{t-1}^{(k)}\right]^{-1}, \quad i, j = 1, \ldots, N, \tag{8}$$

where $\nu_{t-1}^{(i)}$ denotes the resampling weights. Given the ancestor index, a new particle is sampled according to

$$x_t^{(i)} \sim R_\theta\left(x_t \mid x_{1:t-1}^{a_t^{(i)}}, y_t\right), \quad i = 1, \ldots, N. \tag{9}$$

Finally, we append the obtained sample to the trajectory by $x_{1:t}^{(i)} = \{x_{1:t-1}^{a_t^{(i)}}, x_t^{(i)}\}$ and compute a new importance weight by

$$w_t^{(i)} \triangleq \frac{w_{t-1}^{a_t^{(i)}}}{\nu_{t-1}^{a_t^{(i)}}} \frac{g_\theta\left(y_t \mid x_t^{(i)}\right) f_\theta\left(x_t^{(i)} \mid x_{t-1}^{a_t^{(i)}}\right)}{R_\theta\left(x_t^{(i)} \mid x_{1:t-1}^{a_t^{(i)}}, y_t\right)}, \quad i = 1, \ldots, N. \tag{10}$$

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,

$$u = \left(\left\{x_t^{(i)}, a_t^{(i)}\right\}_{i=1}^N, t = 1, \ldots, T\right).$$

In this article, we make use of two important special cases of the APF: 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 kernel $R_\theta(x_t \mid x_{1:t-1}, y_t) = f_\theta(x_t \mid x_{t-1})$ and the auxiliary weights $\nu_t = w_t = g_\theta(y_t \mid x_t)$. The FAPF is obtained by $R_\theta(x_t \mid x_{1:t-1}, y_t) = p_\theta(x_t \mid y_t, x_{t-1})$ and $\nu_t = p_\theta(y_{t+1} \mid x_t)$, resulting in the weights $w_t \equiv 1$. Note, that the FAPF can only be used in models for which these quantities are available in closed-form.

Estimation of the likelihood

The likelihood for the SSM in (1) can be estimated using (3) by inserting estimated one-step predictors $p_\theta(y_t \mid y_{1:t-1})$ obtained from the APF. The resulting estimator is given by

$$\widehat{p}_\theta(y_{1:T} \mid u) = \frac{1}{N^T} \sum_{i=1}^{N} w_T^{(i)} \left\{ \prod_{t=1}^{T-1} \sum_{i=1}^{N} v_t^{(i)} \right\}. \tag{11}$$

It is known that this likelihood estimator is 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 Section 2.1, this is exactly the property that is needed in order to obtain $p(\theta \mid y_{1:T})$ 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 PMH0 in Pitt et al. (2012) and Doucet et al. (2015).

Estimation of the gradient

As we shall see below, the gradient of the log-posterior can be estimated 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 JSD at time T according to (7) (see also Poyiadjis et al. (2011)). However, this method can give estimates with high variance due to *particle degeneracy*.

Instead, we make use of the FL smoother (Kitagawa and Sato, 2001) which has the same linear computational cost, but smaller problems with *particle degeneracy* than the APF. Alternative algorithms for estimating this information are also available (Del Moral et al., 2010; Poyiadjis et al., 2011).

The gradient of the parameter log-posterior is given by

$$\mathcal{G}_T(\theta) = \nabla \log p(\theta) + \nabla \log p_\theta(y_{1:T}), \tag{12}$$

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 $\nabla \log p_\theta(y_{1:T})$ can, using *Fisher's identity* (Cappé et al., 2005), be expressed as

$$\nabla \log p_\theta(y_{1:T}) = \mathbb{E}_\theta \left[\nabla \log p_\theta(x_{0:T}, y_{1:T}) \middle| y_{1:T} \right], \tag{13}$$

where for an SSM (1) we can write the gradient of the complete data log-likelihood as

$$\nabla \log p_\theta(x_{0:T}, y_{1:T}) = \sum_{t=1}^{T} \xi_\theta(x_t, x_{t-1}), \text{ where} \tag{14}$$

$$\xi_\theta(x_t, x_{t-1}) = \nabla \log f_\theta(x_t \mid x_{t-1}) + \nabla \log g_\theta(y_t \mid x_t).$$

Combining (14) with Fisher's identity (13) yields

$$\nabla \log p_\theta(y_{1:T}) = \sum_{t=1}^{T} \int \xi_\theta(x_t, x_{t-1}) p_\theta(x_{t-1:t} \,|\, y_{1:T}) \, \mathrm{d}x_{t-1:t},$$

which depends on the (intractable) two-step smoothing distribution $p_\theta(x_{t-1:t} \,|\, y_{1:T})$. To approximate this quantity 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

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

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 approximation

$$\widehat{p}^{\,\Delta}_\theta(\mathrm{d}x_{t-1:t} \,|\, y_{1:T}) \triangleq \sum_{i=1}^{N} w^{(i)}_{\kappa_t} \delta_{\tilde{x}^{(i)}_{\kappa_t, t-1:t}}(\mathrm{d}x_{t-1:t}). \tag{15}$$

Here, we use the notation $\tilde{x}^{(i)}_{\kappa_t, t}$ to denote the ancestor at time t of particle $x^{(i)}_{\kappa_t}$ and

$$\tilde{x}^{(i)}_{\kappa_t, t-1:t} = \{\tilde{x}^{(i)}_{\kappa_t, t-1}, \tilde{x}^{(i)}_{\kappa_t, t}\}.$$

Inserting (14)–(15) into (13) provides an estimator of (12),

$$\widehat{\mathcal{G}}(\theta \,|\, u) = \nabla \log p(\theta) + \sum_{t=1}^{T} \sum_{i=1}^{N} w^{(i)}_{\kappa_t} \xi_\theta\left(\tilde{x}^{(i)}_{\kappa_t, t}, \tilde{x}^{(i)}_{\kappa_t, t-1}\right), \tag{16}$$

which is used in the proposal distributions in (6).

Estimation of the Hessian

The negative Hessian of the parameter log-posterior can be written as

$$\mathcal{H}_T(\theta) = -\nabla^2 \log p(\theta) - \nabla^2 \log p_\theta(y_{1:T}), \tag{17}$$

where 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* (Cappé et al., 2005) be expressed as

$$\begin{aligned} -\nabla^2 \log p_\theta(y_{1:T}) = {} & \nabla \log p_\theta(y_{1:T})^2 \\ & - \mathbb{E}_\theta\left[\nabla^2 \log p_\theta(x_{0:T}, y_{1:T}) \,|\, y_{1:T}\right] \\ & - \mathbb{E}_\theta\left[\nabla \log p_\theta(x_{0:T}, y_{1:T})^2 \,|\, y_{1:T}\right]. \end{aligned} \tag{18}$$

Here, we have introduced the notation $v^2 = vv^\top$ for a vector v.

From this, we can construct an estimator of the negative Hessian in (17) using the estimator of the gradient in (16), of the form

$$\widehat{\mathcal{H}}(\theta \,|\, u) = -\nabla^2 \log p(\theta) + \widehat{\mathcal{G}}(\theta \,|\, u)^2 - \widehat{\mathcal{H}}^{(1)}(\theta \,|\, u) - \widehat{\mathcal{H}}^{(2)}(\theta \,|\, u), \tag{19}$$

where we introduce the auxiliary quantities given by

$$\mathcal{H}^{(1)}(\theta) = \mathbb{E}_\theta \left[\nabla^2 \log p_\theta(x_{0:T}, y_{1:T}) \,|\, y_{1:T}\right],$$
$$\mathcal{H}^{(2)}(\theta) = \mathbb{E}_\theta \left[\nabla \log p_\theta(x_{0:T}, y_{1:T})^2 \,|\, y_{1:T}\right].$$

We obtain the estimator of the first term analogously to (16) as

$$\widehat{\mathcal{H}}^{(1)}(\theta \,|\, u) = \sum_{t=1}^{T} \sum_{i=1}^{N} w_{\kappa_t}^{(i)} \zeta_\theta\left(\tilde{x}_{\kappa_t,t}^{(i)}, \tilde{x}_{\kappa_t,t-1}^{(i)}\right), \tag{20}$$

where we introduce the additive functional given by

$$\zeta_\theta(x_t, x_{t-1}) = \nabla^2 \log f_\theta(x_t \,|\, x_{t-1}) + \nabla^2 \log g_\theta(y_t \,|\, x_t).$$

The estimator of the second term needs a bit more work and we start by rewriting the last term in (18) as

$$\sum_{t=1}^{T} \sum_{s=1}^{T} \mathbb{E}_\theta \left[\xi_\theta(x_t, x_{t-1}) \xi_\theta(x_s, x_{s-1})^\top \,|\, y_{1:T}\right] = \sum_{t=1}^{T} \Big\{ \mathbb{E}_\theta \left[\xi_\theta(x_t, x_{t-1})^2 \,|\, y_{1:T}\right] + \sum_{s=1}^{t-1} \mathbb{E}_\theta \left[\left(\xi_\theta(x_t, x_{t-1}), \xi_\theta(x_s, x_{s-1})\right)^\dagger \,|\, y_{1:T}\right] \Big\}, \tag{21}$$

where we have introduced the operator $(a, b)^\dagger = ab^\top + ba^\top$ for brevity. Consider the last term appearing in this expression, we can rewrite it as

$$\sum_{s=1}^{t-1} \mathbb{E}_\theta \left[\xi_\theta(x_t, x_{t-1}) \xi_\theta(x_s, x_{s-1})^\top \,|\, y_{1:T}\right]$$
$$= \mathbb{E}_\theta \Bigg[\xi_\theta(x_t, x_{t-1}) \underbrace{\left\{ \sum_{s=1}^{t-1} \mathbb{E}_\theta \left[\xi_\theta(x_s, x_{s-1}) \,|\, x_{t-1}, y_{1:t-1}\right] \right\}^\top}_{\triangleq \alpha_\theta(x_{t-1})^\top} \,|\, y_{1:T} \Bigg].$$

From this, we see that (21) can be written as an additive functional of the form

$$\sum_{t=1}^{T} \mathbb{E}_\theta \left[(\xi_\theta(x_t, x_{t-1}))^2 + \left((\xi_\theta(x_t, x_{t-1}), \alpha_\theta(x_{t-1})\right)^\dagger \,|\, y_{1:T}\right],$$

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

$$\widehat{\alpha}_\theta\left(x_t^{(i)}\right) = \widehat{\alpha}_\theta\left(x_{t-1}^{a_t^{(i)}}\right) + \xi_\theta\left(x_t^{(i)}, x_{t-1}^{a_t^{(i)}}\right), \tag{22}$$

for $i = 1, \ldots, N$. Note that this approach 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 reasonable variance. The resulting estimator using (21) is

$$\widehat{\mathcal{H}}^{(2)}(\theta \mid u) = \sum_{t=1}^{T} \sum_{i=1}^{N} w_{\kappa_t}^{(i)} \eta_\theta\left(\tilde{x}_{\kappa_t,t}^{(i)}, \tilde{x}_{\kappa_t,t-1}^{(i)}\right), \text{ where} \tag{23}$$

$$\eta_\theta(x_t, x_{t-1}) = \xi_\theta(x_t, x_{t-1})^2 + \left(\xi_\theta(x_t, x_{t-1}), \widehat{\alpha}_\theta(x_{t-1})\right)^\dagger.$$

Hence, the Hessian can be estimated using (19) by inserting the estimators from (20), (22) and (23).

Regularisation of the estimate of the Hessian

The PMH2 proposal (6) relies on the assumption that the observed information matrix is positive definite (PD). 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, one alternative is to regularize the Hessian by adding a diagonal matrix to shift the eigenvalues to be positive. The diagonal matrix can e.g., be selected such that

$$\Delta\widehat{\mathcal{H}} = \max\left\{0, -2\lambda_{\min}\left(\widehat{\mathcal{H}}\right)\right\} I_d, \tag{24}$$

where $\lambda_{\min}(\widehat{\mathcal{H}})$ denotes the smallest eigenvalue of $\widehat{\mathcal{H}}(\theta \mid u)$. In this article, we make use of this method for handling non–PD estimates of the negative Hessian for the PMH2 algorithm. This heuristic is common for Newton-type optimisation algorithms (Nocedal and Wright, 2006, Chapter 3.4).

Note, that there are other solutions available for ensuring positive definiteness that only shifts the negative eigenvalues, see (Nocedal and Wright, 2006, Chapter 3). We emphasise that this type of regularization keeps the Markov chain invariant, i.e., still targets the correct posterior distribution (recall Section 2.1).

Another alternative is to replace the estimate of the negative Hessian with the inverse sample covariance matrix calculated using the trace of Markov chain when the estimate is not PD. This can be seen as a hybrid between the PMH2 algorithm and a *pre-conditioned PMH1 algorithm*. This resembles some other adaptive MH algorithms (Andrieu and Thoms, 2008) in which the same procedure is used to adapt the covariance matrix of a random walk proposal. For this, we can make use of the last L iterations of the MH algorithm after that the Markov chain has reached stationarity. During the burn-in phase, non–PD estimates can be handled using a regularization approach or by rejecting the proposed parameter. In this article, we refer to this method for handling non–PD estimates of the negative Hessian as the *hybrid* PMH2 *algorithm*, where we use the latter alternative during the burn-in phase. Note that this pre-conditioning can also be applied to the PMH0 and PMH1 algorithm, we return to this in Section 4.4.

Algorithm 2 Estimation of the likelihood, the gradient and the Hessian of the log-posterior

Inputs: $y_{1:T}$ (data), $R(\cdot)$ (propagation kernel), $v(\cdot)$ (weight function), $N > 0$ (no. particles), $0 < \Delta \leq T$ (lag).

Outputs: $\widehat{p}_\theta(\mathbf{y})$ (est. of the likelihood), $\widehat{\mathcal{G}}(\theta)$ (est. of the gradient), $\widehat{\mathcal{H}}(\theta)$ (est. of the negative Hessian).

1: Initialise each particle $x_0^{(i)}$.
2: **for** $t = 1$ to T **do**
3: Resample and propagate each particle using (9).
4: Calculate the weights for each particle using (10).
5: **end for**
6: Compute $\widehat{p}_\theta(y_{1:T})$ by (11).
7: Compute $\widehat{\mathcal{G}}(\theta)$ and $\widehat{\mathcal{H}}(\theta)$ by (16) and (19), respectively.
8: **if** $\widehat{\mathcal{H}}(\theta) \leq 0$ **then**
9: **[standard]** Regularize $\widehat{\mathcal{H}}(\theta)$ by adding $\Delta\widehat{\mathcal{H}}$ computed by (24)
10: **[hybrid]** Replace $\widehat{\mathcal{H}}(\theta)$ by the inverse covariance matrix computed using the L final samples of the Markov chain during the burn-in.
11: **end if**

Resulting SMC algorithm

In Algorithm 2, we present the complete procedure that combines the APF with the FL smoother to compute the estimates needed for the PMH2 proposal (6). Note that the two different methods to handle non–PD estimates of the negative Hessian matrix results in the *standard* and *hybrid* PMH2 algorithm, respectively.

We end this section by briefly discussing the statistical properties of the estimates of the gradient and Hessian obtained from the FL smoother. From Olsson et al. (2008), we know that the FL smoother gives biased estimates of the gradient and Hessian for any number of particles. Remember that this does not effect the invariance of the Markov chain (recall Section 2.1). The main advantage of the FL smoother over the APF (which gives a consistent estimate) is that the former enjoys a smaller variance than the APF, i.e., we obtain a favourable bias-variance trade-off for a certain choice of lag Δ. 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.

Numerical illustrations

In this section, we provide illustrations of the properties of the FL smoother and the different proposed algorithms. The source code in Python and the data used for some of the numerical illustrations are available for download from GitHub at: `https://github.com/compops/pmh-stco2015`.

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. Here, we consider a linear Gaussian state space (LGSS) model given by

$$x_{t+1} \mid x_t \sim \mathcal{N}\left(x_{t+1}; \phi x_t, \sigma_v^2\right), \qquad y_t \mid x_t \sim \mathcal{N}\left(y_t; x_t, \sigma_e^2\right). \tag{25}$$

We generate two data realisations of length $T = 100$ 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 zero state. We use the lag $\Delta = 5$ and run the PFs with systematic resampling (Carpenter et al., 1999).

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 $\sigma_e = 0.1$, we observe that the FAPF has a large advantage over the BPF for all choices of N. When $\sigma_e = 1.0$, 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.

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 obtained by 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. The results are largely robust to the lag, as long as this is chosen large enough 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.

Burn-in and scale-invariance

Consider the problem of inferring $\{\theta_1, \theta_2\} = \{\phi, \sigma_v\}$ in the LGSS model (25). We simulate a single data set with parameters $\theta^{(1)}$ (as defined in the previous section) of length $T = 250$. We use an uniform parameter prior over $|\phi| < 1, \sigma_v > 0$ and initialise in $\theta_0 = \{0.1, 2\}$. We use FAPF with systematic resampling, $N = 100$ and $\Delta = 12$. Here, we use the standard version of Algorithm 2 to adjust the estimate of the Hessian in the cases when it is not PD, resulting in the PMH2 algorithm.

We adjust the step lengths ϵ to give an acceptance rate during a pilot run of between 0.7 and 0.8 in the stationary phase. We obtain $\epsilon = \{0.04, 0.065, 1.0\}$ 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 using the approach discussed in Section 2.2. 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 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

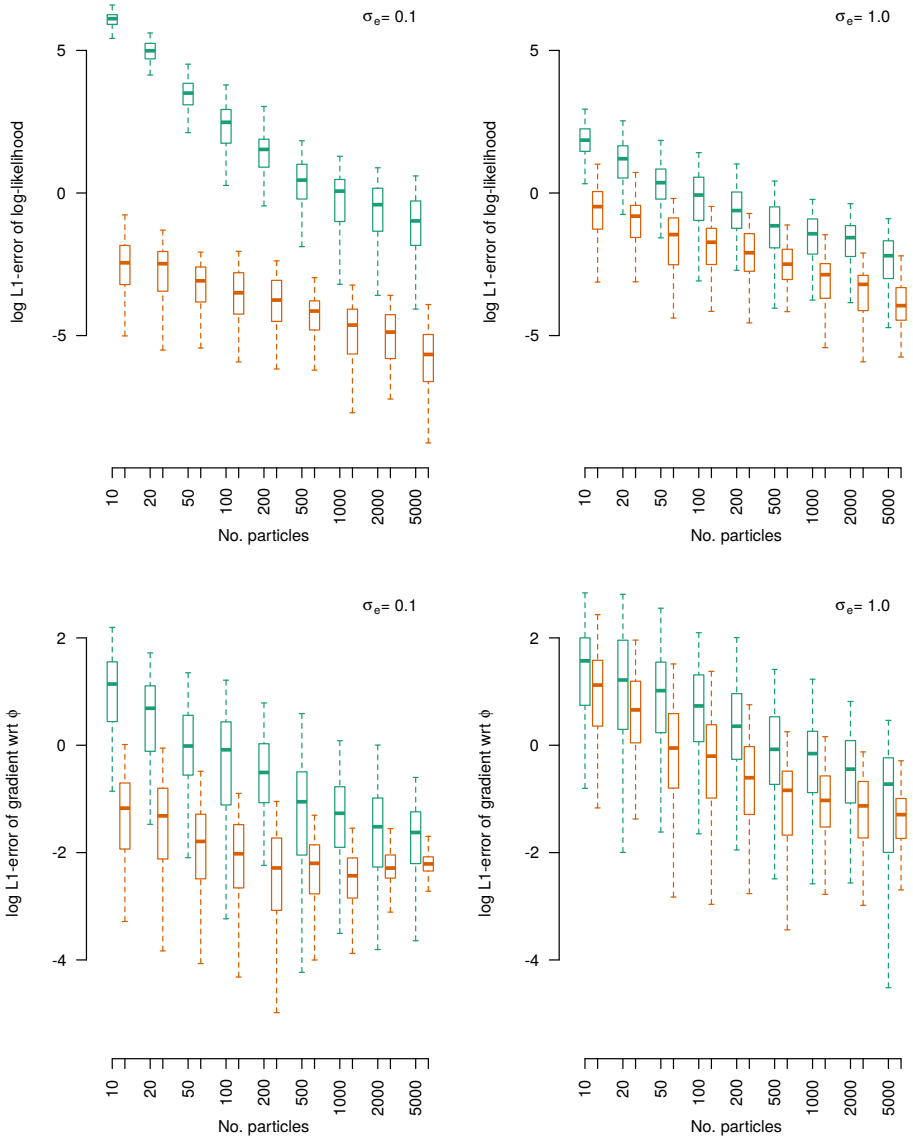


Figure 1. The log L_1-error in the log-likelihood estimates and the estimates of the gradient with respect to ϕ in the LGSS model with $\sigma_e = 0.1$ (left) and $\sigma_e = 1$ (right). The BPF (green) and FAPF (orange) are evaluated by 1,000 MC iterations using a fixed data set with $T = 100$.

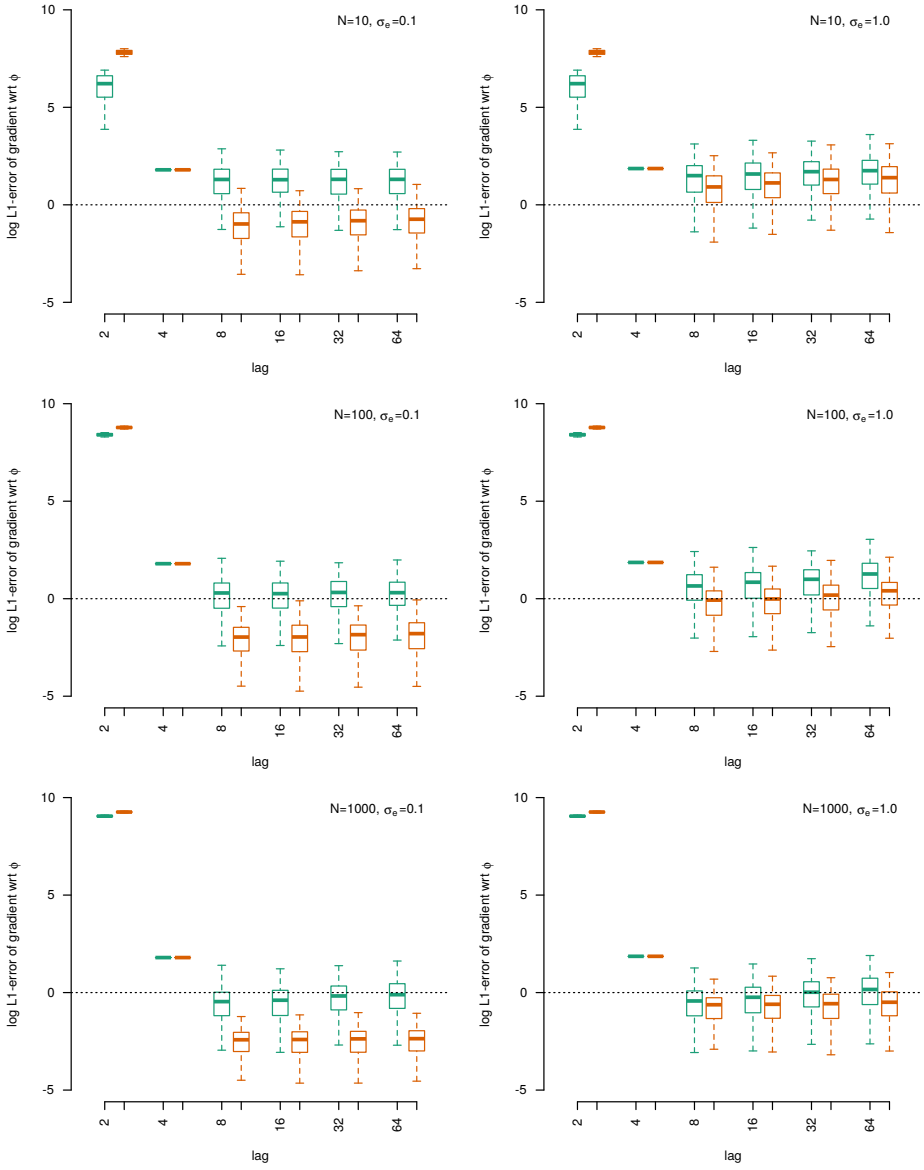


Figure 2. The log L_1-error in the estimates of the gradient with respect to ϕ in the LGSS model with $\sigma_e = 0.1$ (left) and $\sigma_e = 1$ (right). The BPF (green) and FAPF (orange) are evaluated by 1,000 Monte Carlo iterations using a fixed data set with $T = 100$.

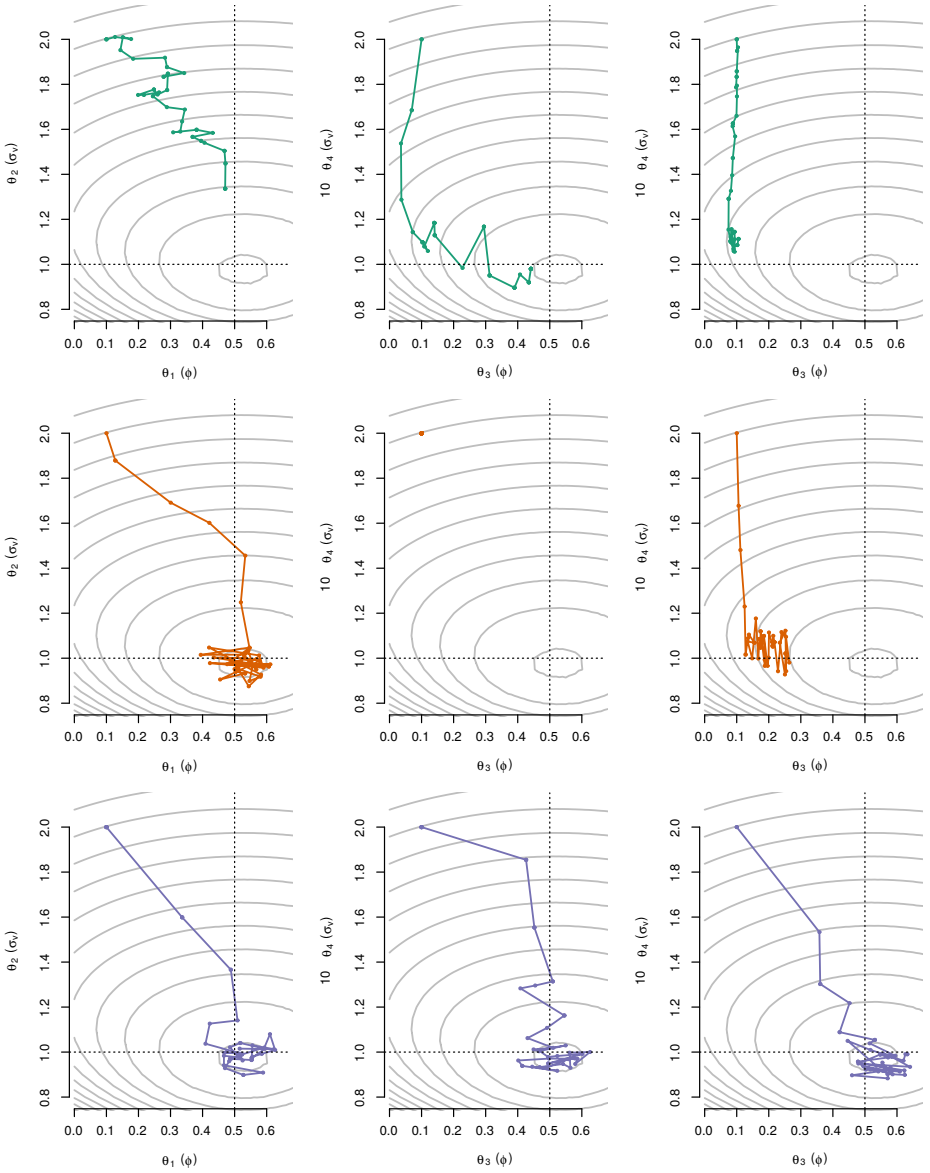


Figure 3. The trace plots of the first 50 steps using PMH0 (green), PMH1 (orange) and PMH2 (purple). The dotted lines show the *true* parameters of the LGSS model. The gray contours show the log-posterior.

chains for the proposed algorithms reach the mode of the posterior quicker than the random walk used in PMH0.

To illustrate the scale invariance of the PMH2 algorithm, we reparametrise the LGSS model by $\{\theta_3, \theta_4\} = \{\phi, \sigma_v/10\}$. 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 PMH1-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, we recalibrate the three algorithms on the new parametrisation using the same procedure as before and 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 Hessian information is used to rescale the proposal in each dimension resulting in a more efficient exploration of the posterior than for PMH0 and PMH1.

The mixing of the Markov chains at stationarity

We continue by investigating the mixing of the Markov chains at stationarity using an estimate of the integrated autocorrelation time (IACT) given by

$$\widehat{\mathsf{IACT}}(\theta_{1:K}) = 1 + 2\sum_{k=1}^{K_l} \widehat{\rho}_\tau(\theta_{1:K}), \tag{26}$$

where $\widehat{\rho}_\tau(\theta_{1:K})$ denotes the empirical autocorrelation at lag τ of $\theta_{1:K}$ (after the burn-in has been discarded). A low value of the IACT indicates that we obtain many uncorrelated samples from the target distribution, implying that the chain is mixing well. Here, K_l is determined as the first index for which the empirical autocorrelation satisfies $|\widehat{\rho}_{K_l}(\theta_{1:K})| < 2/\sqrt{K}$, i.e., when the coefficient is statistically insignificant.

We return to the LGSS model in (25) with the original parameterisation $\{\theta_1, \theta_2\} = \{\phi, \sigma_v\}$ using the same settings as before. A total of 25 data sets are generated using the parameters $\theta^{(1)}$ and the algorithms are initialised at the true parameter values to avoid a long burn-in phase. The step sizes are determined using a series of pilot runs on the first generated dataset to minimise the total IACT for each algorithm. This is done to make a fair comparison between the different algorithms at their near *optimal* performance. The resulting step sizes are obtained as $\{0.08, 0.075, 1.50\}$.

Finally, we estimate the mixing in each of the 25 simulated data sets during $K = 30,000$ MCMC iterations (discarding the first 10,000 iterations as burn-in). The results are presented in Table 1, where the median and interquartile range (IQR; the distance between the 25% and 75% quartiles) are presented for each PMH algorithm. Here, we present the results from the standard version of Algorithm 2.

		Acc. rate	IACT(ϕ)		IACT(σ_v)	
		Median	Median	IQR	Median	IQR
PMH0	BPF(500)	0.02	257	146	265	371
	BPF(1000)	0.06	83	129	79	118
	BPF(2000)	0.15	29	23	15	24
	FAPF(50)	0.37	9	8	8	5
	FAPF(100)	0.38	9	6	7	4
	FAPF(200)	0.38	7	6	7	4
PMH1	BPF(500)	0.02	187	271	203	347
	BPF(1000)	0.10	64	85	49	72
	BPF(2000)	0.22	23	16	12	24
	FAPF(50)	0.58	**3**	2	**3**	1
	FAPF(100)	0.59	4	2	**3**	1
	FAPF(200)	0.58	**3**	1	**3**	1
PMH2	BPF(500)	0.03	170	211	164	190
	BPF(1000)	0.10	59	73	65	80
	BPF(2000)	0.24	13	10	19	17
	FAPF(50)	0.66	**3**	1	4	2
	FAPF(100)	0.66	**3**	1	5	2
	FAPF(200)	0.66	**3**	1	4	2

Table 1. Median and IQR for the acceptance rate and IACT using different SMC algorithms. The values are computed using 25 different data sets from the LGSS model.

We see that the added information decreases the IACT about 2 times for PMH1 and PMH2 compared with PMH0. We conclude that the extra information brought by the gradient and the Hessian improves the mixing of the Markov chains 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.

Parameter inference in a Poisson count model

In this section, we analyse the annual number of major earthquakes[1] (over 7 on the Richter scale) during the period from year 1900 to 2014. Following Langrock (2011), we model the data using

$$x_{t+1} \,|\, x_t \sim \mathcal{N}\big(x_{t+1}; \phi x_t, \sigma^2\big), \qquad y_t \,|\, x_t \sim \mathcal{P}\big(y_t; \beta \exp(x_t)\big), \tag{27}$$

with parameters $\theta = \{\phi, \sigma, \beta\}$ and uniform priors over $|\phi| < 1$, $\sigma > 0$ and $\beta > 0$. Here, $\mathcal{P}(\lambda)$ denotes a Poisson distribution with parameter λ.

We repeat the procedure from the previous subsection and obtain the step lengths using pilot runs given by $\{0.06, 0.006, 0.85\}$. Here, we use $K = 30,000$ MCMC iterations (discarding the first $10,000$ iterations as burn-in), the BPF with systematic resampling, $\Delta = 12$, $\theta_0 = \{0.5, 0.5, 18\}$ and $L = 2,500$. In this model, the estimate of the negative Hessian is often non–pd (during about half of the iterations) and the choice of regularisation is therefore important. To explore the properties of the regularisation, we apply both the standard and hybrid version of the PMH2 algorithm discussed in Section 3.5. We compare these methods to standard and pre-conditioned versions of the the PMH0 and PMH1 algorithms, using the sample posterior covariance matrix calculated in the same manner as for the hybrid PMH2.

In Table 2, we present the resulting acceptance rates and IACT values for each parameter and algorithm. We note the large decrease in IACT for β when using the Hessian information, where the hybrid PMH2 seems to perform better than standard version for this model. The improved mixing by using PMH2 is due to the scale invariance property, as the parameter β is at least an order of magnitude larger than ϕ and σ (c.f. Figure 3). Note that a reparameterisation or using separate step lengths for the parameters could possibly have helped in improving the mixing in β for the standard versions of PMH0 and PMH1.

Using the standard and hybrid version of PMH2, decreases the overall computational cost by a factor of about 100 for a specific number of effective samples. The poor performance of the pre-conditioned algorithms is probably due to that the sample posterior covariance matrix does not fully capture the geometry of the posterior distribution.

In Figure 4, we present the trace and posterior estimates for β using the standard versions of PMH0 and PMH1 as well as hybrid PMH2. The posterior estimates are obtained by pooling the 10 parallel Markov chains after the burn-ins have been discarded. We see that the traces behave rather differently with hybrid PMH2 exploring the space well compared with the other methods.

[1]The data is obtained from the Earthquake Data Base System of the U.S. Geological Survey, which can be accessed at `http://earthquake.usgs.gov/earthquakes/eqarchives/`.

	Version	SMC alg.	**Acc. rate**	IACT(ϕ)		IACT(σ)		IACT(β)	
			Median	Median	IQR	Median	IQR	Median	IQR
PMH0	Standard	BPF(500)	0.26	497	712	16	3	2639	1163
	Standard	BPF(1000)	0.30	89	150	15	3	2680	438
	Pre-cond.	BPF(500)	0.43	35	17	16	1	107	105
	Pre-cond.	BPF(1000)	0.45	38	28	16	2	129	131
PMH1	Standard	BPF(500)	0.76	665	442	277	162	2651	364
	Standard	BPF(1000)	0.82	490	134	205	30	2875	1007
	Pre-cond.	BPF(500)	0.62	266	187	**9**	3	1728	1638
	Pre-cond.	BPF(1000)	0.70	98	209	**9**	3	1480	1732
PMH2	Standard	BPF(500)	0.24	91	17	53	14	222	37
	Standard	BPF(1000)	0.28	60	14	47	17	139	59
	Hybrid	BPF(500)	0.45	20	3	17	4	30	15
	Hybrid	BPF(1000)	0.49	**17**	4	18	3	**23**	5

Table 2. Median and IQR for the acceptance rate and IACT using different number of particles. The values are computed using 10 runs on the Earthquake count data model.

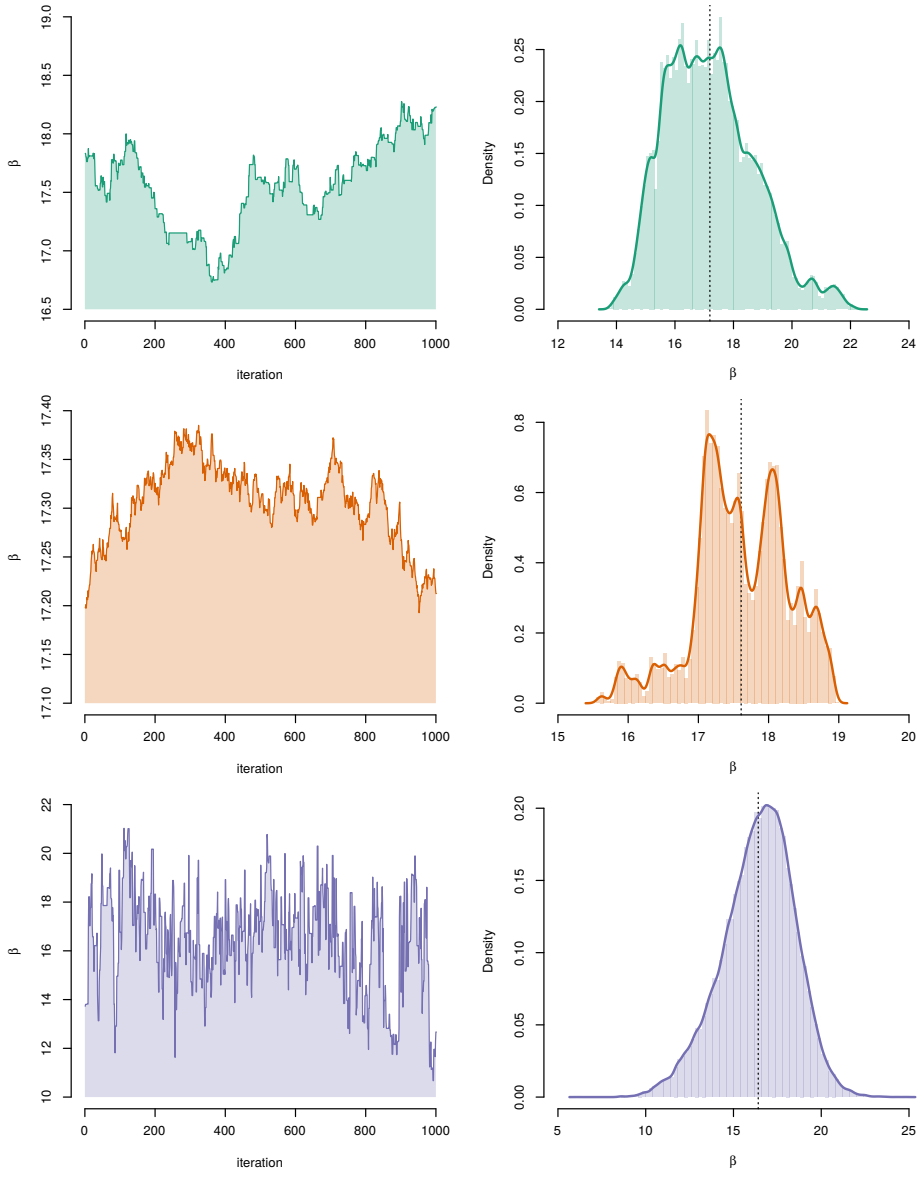


Figure 4. Part of the trace (left) and posterior estimates (right) for the β parameter in the earthquake count model using standard versions of PMH0 (green), PMH (orange) and hybrid version of PMH2 (purple). Dotted lines indicate the posterior means.

Using the parameter posterior estimate, we can compute point estimates for the parameters of the model. The posterior mean for hybrid PMH2 is obtained as $\{0.88, 0.15, 16.58\}$ with standard deviations $\{0.07, 0.03, 2\}$. The parameter estimate is comparable to the estimate $\{0.88, 0.15, 17.65\}$ obtained by a maximum likelihood-based method using the same data and model in Dahlin (2014, Example 4.9).

Robustness in the lag and step size

The PMH2 algorithm requires a number of parameters to be select by the user for each parameter inference problem. It is therefore interesting to discuss the robustness of the method with respect to these parameters. In the previous illustrations, we have seen that the number of particles N is an important factor in determining the mixing.

Two other important parameters are the step length ϵ and the lag in the fl-smoother Δ. To illustrate the impact of these quantities on the iact, we return to the Earthquake model in (27) using the standard PMH2 algorithm with the same settings but with $K = 15,000$ (discarding the first $5,000$ iterations as burn-in) and $N = 1,500$. In Figure 5, we present the IACT for the three parameters in the model when varying ϵ and Δ, keeping everything else fixed. The standard PMH2 algorithm seems to be rather robust to both the choice of Δ and ϵ after a certain threshold. Recall the discussion in Section 4.1 for the FL smoother. We conclude that a suitable standard choice for the step length could be $\epsilon = 1$, which can be fine tuned if the performance is not good enough. This recommendation is also common in the literature concerning Newton-type algorithms.

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 and stationary phases. We have seen empirically that the proposed methods can improve upon the original PMH0 during both of these phases but the *best* choices for the step lengths can differ between these two 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); Pitt et al. (2012)), to automatically tune the step lengths during the different phases of the algorithms. Another interesting possibility is

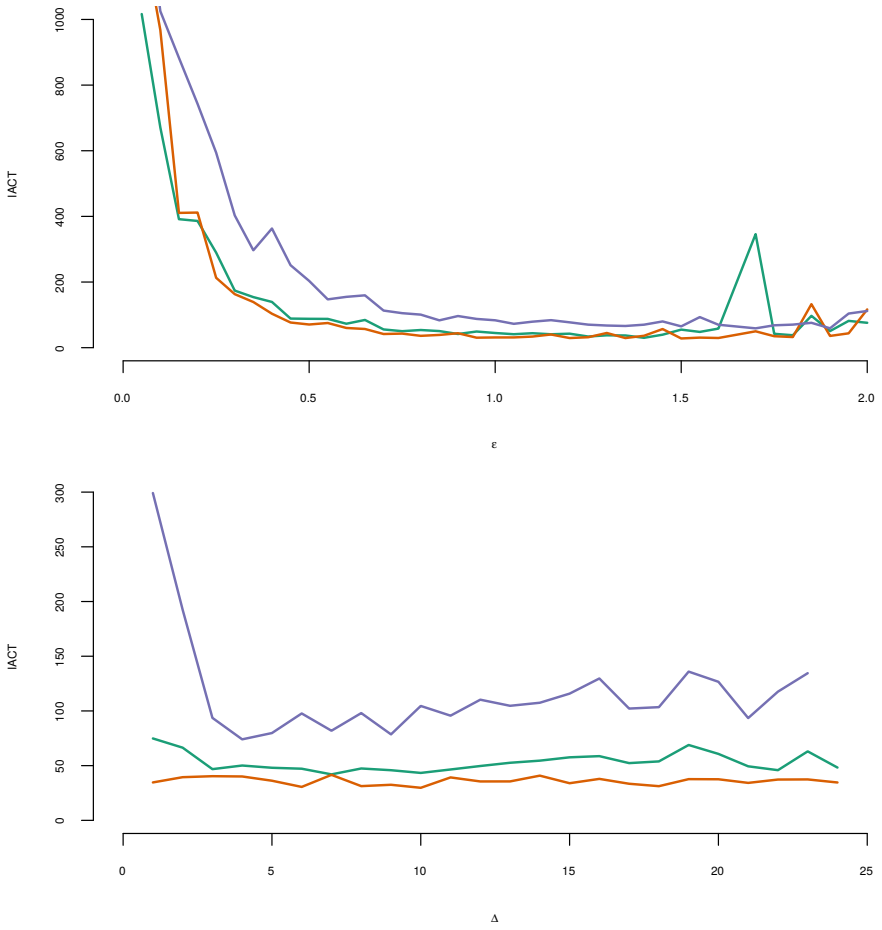


Figure 5. The IACT for ϕ (green), σ (orange) and β (purple) for varying step sizes ϵ (upper) and lag Δ (lower). The values are computed as the median of 10 runs using standard PMH2 with the same data.

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.

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. *Annals of Applied Probability*, 25(2):1030–1077, 2015.

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 Verlag, 2005.

J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings Radar, Sonar and Navigation*, 146(1):2–7, 1999.

J. Dahlin. *Sequential Monte Carlo for inference in nonlinear state space models*. Licentiate's thesis no. 1652, Linköping University, May 2014.

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 2014.

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015.

P. Del Moral. *Feynman-Kac formulae - genealogical and interacting particle systems with applications*. Springer Verlag, 2004.

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. *The 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, 2011.

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.

A. Doucet, M. K. Pitt, G. Deligiannidis, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2): 295–313, 2015.

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.

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.

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 Verlag, 2001.

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.

C. Nemeth and P. Fearnhead. Particle Metropolis adjusted Langevin algorithms for state-space models. *Pre-print*, 2014. arXiv:1402.0694v1.

C. Nemeth, C. Sherlock, and P. Fearnhead. Particle Metropolis adjusted Langevin algorithms. *Pre-print*, 2014. arXiv:1412.7299v1.

J. Nocedal and S. Wright. *Numerical optimization*. Springer Verlag, 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 Verlag, 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. *The Annals of Statistics*, 43(1):238–275, 2015.

Paper C

Quasi-Newton particle Metropolis-Hastings

Authors: J. Dahlin, F. Lindsten and T. B. Schön

This paper is published by Elsevier:
http://dx.doi.org/10.1016/j.ifacol.2015.12.258.

Edited version of the paper:

J. Dahlin, F. Lindsten, and T. B. Schön. Quasi-Newton particle Metropolis-Hastings. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 981–986, Beijing, China, October 2015c.

Quasi-Newton particle Metropolis-Hastings

J. Dahlin*, F. Lindsten† and T. B. Schön†

*Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden.
johan.dahlin@liu.se

†Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
fredrik.lindsten@it.uu.se
thomas.schon@it.uu.se

Abstract

Particle Metropolis-Hastings enables Bayesian parameter inference in general non-linear state space models (SSMs). However, in many implementations a random walk proposal is used and this can result in poor mixing if not tuned correctly using tedious pilot runs. Therefore, we consider a new proposal inspired by quasi-Newton algorithms that may achieve similar (or better) mixing with less tuning. An advantage compared to other Hessian based proposals, is that it only requires estimates of the gradient of the log-posterior. A possible application is parameter inference in the challenging class of SSMs with intractable likelihoods. We exemplify this application and the benefits of the new proposal by modelling log-returns of future contracts on coffee by a stochastic volatility model with α-stable observations.

Keywords

Bayesian parameter inference, state space models, approximate Bayesian computations, particle Markov chain Monte Carlo, α-stable distributions.

Data and source code in Python

`https://github.com/compops/qpmh-sysid2015`

Financial support from

The projects *Learning of complex dynamical systems* (Contract number: 637-2014-466), *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) and CADICS, a Linnaeus Center, all funded by the Swedish Research Council.

Introduction

We are interested in Bayesian parameter inference in the non-linear state space model (SSM) possibly with an intractable likelihood. An SSM with latent states $x_{0:T} = \{x_t\}_{t=0}^{T}$ and observations $y_{1:T}$ is given by

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

with $x_0 \sim \mu_\theta(x_0)$ and where $\theta \in \Theta \subseteq \mathbb{R}^p$ denotes the static unknown parameters. Here, we assume that it is possible to simulate from the distributions $\mu_\theta(x_0)$, $f_\theta(x_{t+1} \,|\, x_t)$ and $g_\theta(y_t \,|\, x_t)$, even if the respective densities are unavailable.

The main object of interest in Bayesian inference is the *parameter posterior* distribution,

$$\pi(\theta) = p(\theta \,|\, y_{1:T}) \propto p_\theta(y_{1:T}) p(\theta), \tag{2}$$

which is often intractable and cannot be computed in closed form. The problem lies in that the likelihood $p_\theta(y_{1:T}) = p(y_{1:T} \,|\, \theta)$ cannot be exactly computed. However, it can be estimated by computational statistical methods such as sequential Monte Carlo (SMC; Doucet and Johansen, 2011). The problem is further complicated when $g_\theta(y_t \,|\, x_t)$ cannot be evaluated point-wise, which prohibits direct application of SMC. This could be the result of that the density does not exist or that it is computationally prohibitive to evaluate. In both cases, we say that the likelihood of the SSM (1) is *intractable*.

Recent efforts to develop methods for inference in models with intractable likelihoods have focused on approximate Bayesian computations (ABC; Marin et al., 2012). The main idea in ABC is that data *simulated* from the model (using the correct parameters) should be *similar* to the observed data. This idea can easily be incorporated into many existing inference algorithms, see Dean et al. (2014).

An example of this is the ABC version of particle Metropolis-Hastings (PMH-ABC; Jasra, 2015; Bornn et al., 2014). In this algorithm, the intractable likelihood is replaced with an estimate obtained by the ABC version of SMC (SMC-ABC; Jasra et al., 2012). However, the random walk proposal often used in PMH-(ABC) can result in problems with poor mixing, which leads to high variance in the posterior estimates. The mixing can be improved by *pre-conditioning* the proposal with a matrix $\mathcal{P}$, which typically is chosen as the unknown posterior covariance (Sherlock et al., 2015). However, estimating $\mathcal{P}$ can be challenging when p is large or the posterior (2) is non-isotropic. This typically results in that the user needs to run many tedious pilot runs when implementing PMH-(ABC) for parameter inference in a new SSM.

Our main contribution is to adapt a *limited-memory BFGS algorithm* (Nocedal and Wright, 2006) as a proposal in PMH-(ABC). This is based on earlier work by Dahlin et al. (2015a) and Zhang and Sutton (2011). In the former, we discuss how to make use of gradient ascent and Newton-type proposals in PMH. The advantages of the new proposal are; (i) good mixing when gradient estimates are accurate, (ii) no tedious pilot runs required and (iii) only requires gradients to approximate the local Hessian. These advantages are important as direct estimation of gradients using SMC often is simpler than for Hessians. Note that this new proposal is useful both with and without the ABC approximation of the likelihood.

To demonstrate these benefits we consider a linear Gaussian state space (LGSS) model, where we compare the performance of our proposal with and without ABC. Furthermore, we consider using a stochastic volatility model with α-stable log-returns (Nolan, 2003) to model future contracts on coffee. This model is common in the ABC literature as the likelihood is intractable, see Dahlin et al. (2015c), Jasra (2015) and Yıldırım et al. (2014).

Particle Metropolis-Hastings

A popular approach to estimate the parameter posterior (2) is to make use of statistical simulation methods. PMH (Andrieu et al., 2010) is one such method and it operates by constructing a Markov chain, which has the sought posterior as its stationary distribution. As a result, we obtain samples from the posterior by simulating the Markov chain for a large number of iterations after its has converged.

The Markov chain targeting (2) is constructed by an iterative procedure. During iteration k, we propose a candidate parameter $\theta' \sim q(\theta' \mid \theta_{k-1}, u_{k-1})$ and an *auxiliary variable* $u' \sim m_{\theta'}(u')$ as detailed in the following using proposals q and $m_{\theta'}$. The candidate is then *accepted*, i.e., $\{\theta_k, u_k\} \leftarrow \{\theta', u'\}$, with the probability

$$\alpha(\theta', \theta_{k-1}, u', u_{k-1}) = \frac{\widehat{\pi}(\theta' \mid u')}{\widehat{\pi}(\theta_{k-1} \mid u_{k-1})} \frac{q(\theta_{k-1} \mid \theta', u')}{q(\theta' \mid \theta_{k-1}, u_{k-1})}, \tag{3}$$

otherwise the parameter is *rejected*, i.e., $\{\theta_k, u_k\} \leftarrow \{\theta_{k-1}, u_{k-1}\}$. Here, we introduce the notation $\widehat{\pi}(\theta \mid u) = \widehat{p}_\theta(y_{1:T} \mid u)p(\theta)$ for some *unbiased* estimate of $\pi(\theta)$ constructed using u and $p(\theta)$ denotes the parameter prior distribution.

PMH can be viewed as a Metropolis-Hastings algorithm in which the intractable likelihood is replaced with an unbiased noisy estimate. It is possible to show that this so-called *exact approximation* results in a valid algorithm as discussed by Andrieu and Roberts (2009). Specifically, the Markov chain generated by PMH converges to the desired stationary distribution despite the fact that we are using an approximation of the likelihood. It is also possible to show that u can be included into the proposal q, which is necessary for including gradients and Hessians when proposing θ' as discussed by Dahlin et al. (2015a).

In Section 4, we discuss how to construct the proposal m_θ by running an SMC algorithm. In this case, the auxiliary variable u is the resulting generated particle system. We obtain PMH-ABC as presented in Algorithm 1, when SMC-ABC is used for Step 4. This is a complete procedure for generating K correlated samples $\{\theta_1, \ldots, \theta_K\}$ from (2). By the ergodic theorem, we can estimate any posterior expectation of an integrable *test function* $\varphi : \Theta \to \mathbb{R}$ (e.g., the posterior mean) by

$$\mathbb{E}\left[\varphi(\theta) \mid y_{1:T}\right] \approx \widehat{\varphi}_{\mathrm{MH}} \triangleq \frac{1}{K - K_b} \sum_{k=K_b}^{K} \varphi(\theta_k), \tag{4}$$

which is a strongly consistent estimator if the Markov chain is ergodic (Meyn and Tweedie, 2009). Here, we discard the first K_b samples known as the *burn-in*, i.e., before the chain reaches stationarity. Under geometric mixing conditions, the error of the estimate obeys

Algorithm 1 Particle Metropolis-Hastings (PMH)

Inputs: $K > 0$ (no. MCMC steps), θ_0 (initial parameters) and $\{q, m_\theta\}$ (proposals).
Output: $\{\theta_1, \ldots, \theta_K\}$ (approximate samples from the posterior).

Generate $u_0 \sim m_{\theta_0}$ and compute $\widehat{p}_{\theta_0}(y_{1:T} \mid u_0)$.
for $k = 1$ to K **do**
  Sample $\theta' \sim q(\theta' \mid \theta_{k-1}, u_{k-1})$.
  Sample $u' \sim m_{\theta'}$ using Algorithm 2.
  Compute $\widehat{p}_{\theta'}(y_{1:T} \mid u')$ using (10).
  Sample ω_k uniformly over $[0, 1]$.
  if $\omega_k \leq \min\{1, \alpha(\theta', \theta_{k-1}, u', u_{k-1})\}$ given by (3) **then**
    Accept θ', i.e., $\{\theta_k, u_k\} \leftarrow \{\theta', u'\}$.
  else
    Reject θ', i.e., $\{\theta_k, u_k\} \leftarrow \{\theta_{k-1}, u_{k-1}\}$.
  end if
end for

the central limit theorem (when $(K - K_b) \to \infty$) given by

$$\sqrt{K - K_b}\left[\widehat{\varphi}_{\text{MH}} - \mathbb{E}[\varphi(\theta) \mid y_{1:T}]\right] \xrightarrow{d} \mathcal{N}\left(0, \sigma_\varphi^2\right), \tag{5}$$

where σ_φ^2 denotes the variance of the estimator. The variance is proportional to the integrated autocorrelation time (IACT), which describes the *mixing* of the Markov chain. Hence, we can use IACT in the illustrations presented in Section 5 to compare the mixing between different proposals.

Proposal for parameters

To complete Algorithm 1, we need to specify a proposal q from which we sample θ'. The choice of proposal is important as it is one of the factors that influences the mixing of resulting Markov chain. The general form of a Gaussian proposal discussed in Dahlin et al. (2015a) is

$$q(\theta'' \mid \theta', u') = \mathcal{N}\left(\theta''; \mu(\theta', u'), \Sigma(\theta', u')\right), \tag{6}$$

where different choices of the mean function $\mu(\theta', u')$ and covariance function $\Sigma(\theta', u')$ results in different versions of PMH as presented in Table 1.

Zeroth and first-order proposals (PMH0/1)

PMH0 is referred to as a zero order (or marginal) proposal as it only makes use of the last accepted parameter to propose the new parameter. Essentially, this proposal is a Gaussian random walk scaled by a positive semi-definite (PSD) *preconditioning matrix* $\mathcal{P}^{-1}$. The performance of PMH0 is highly dependent on $\mathcal{P}^{-1}$, which is tedious and difficult to estimate as it should be selected as the unknown posterior covariance, see Sherlock et al. (2015).

Table 1. Different proposals for the PMH algorithm.

Proposal	$\mu(\theta', u')$	$\Sigma(\theta', u')$
PMH0	θ'	$\epsilon_0^2 \mathcal{P}^{-1}$
PMH2	$\theta' + \frac{\epsilon_1^2}{2}\left[\mathcal{P}^{-1}\,\widehat{\mathcal{G}}(\theta' \mid u')\right]$	$\epsilon_1^2 \mathcal{P}^{-1}$
PMH2	$\theta' + \frac{\epsilon_2^2}{2}\left[\widehat{\mathcal{H}}(\theta' \mid u')\right]^{-1} \widehat{\mathcal{G}}(\theta' \mid u')$	$\epsilon_2^2\left[\widehat{\mathcal{H}}(\theta' \mid u')\right]^{-1}$

Furthermore, it is known that gradient information can be useful to give the proposal a mode-seeking behaviour. This can be beneficial both initially to find the mode and for increasing mixing by keeping the Markov chain in areas with high posterior probability. This information can be included by making use of noisy gradient ascent update, where $\widehat{\mathcal{G}}(\theta' \mid u')$ denotes the particle estimate of the gradient of the log-posterior given by $\mathcal{G}(\theta') = \nabla \log \pi(\theta)|_{\theta=\theta'}$. Again, we scale the step size and the gradient by $\mathcal{P}^{-1}$ and this results in the PMH1 proposal.

Second-order proposal (PMH2)

An alternative is to make use of a noisy Newton update as the proposal by replacing $\mathcal{P}$ with $\widehat{\mathcal{H}}(\theta' \mid u')$, which denotes the particle estimate of the negative Hessian of the log-posterior given by $\mathcal{H}(\theta') = -\nabla^2 \log \pi(\theta)|_{\theta=\theta'}$. This results in the second-order PMH2 proposal discussed by Dahlin et al. (2015a), which relies on accurate estimates of the Hessian but these often require many particles and therefore incur a high computational cost. This problem is encountered in e.g., the ABC approximation of the α-stable model in (9).

The new quasi-PMH2 (QPMH2) proposal circumvents this problem by constructing a local approximation of the Hessian based on a quasi-Newton update, which only makes use of gradient information. The update is inspired by the limited-memory BFGS algorithm (Nocedal, 1980; Nocedal and Wright, 2006) given by

$$B_{l+1}^{-1}(\theta') = \left(\mathbf{I}_p - \rho_l s_l g_l^\top\right) B_l^{-1} \left(\mathbf{I}_p - \rho_l g_l s_l^\top\right) + \rho_l s_l s_l^\top, \tag{7}$$

with $\rho_l^{-1} = g_l^\top s_l$, $s_l = \theta_{I(l)} - \theta_{I(l-1)}$ and $g_l = \widehat{\mathcal{G}}(\theta_{I(l)} \mid u_{I(l)}) - \widehat{\mathcal{G}}(\theta_{I(l-1)} \mid u_{I(l-1)})$. The update is iterated over $l \in \{1, 2, \ldots, M-1\}$ with $I(l) = k - l$, i.e., over the $M-1$ previous states of the Markov chain. Hence, we refer to M as the memory length of the proposal. We initialise the update with $B_1^{-1} = \rho_1^{-1}(g_1^\top g_1)^{-1}\mathbf{I}_p$ and make use of a PMH0 proposal with $\mathcal{P} = \delta^{-1}\mathbf{I}_p$ for the first M iterations, where $\delta > 0$ is defined by the user. The resulting estimate of the negative Hessian is given by $\widehat{\mathcal{H}}(\theta' \mid u') = -B_M(\theta')$. See Appendix B for more details regarding the implementation of the QPMH2 proposal and the complete algorithm in Algorithm 3.

An apparent problem with using a quasi-Newton approximation of the Hessian is that the resulting proposal is no longer Markov (resulting in a non-standard MCMC). However, as shown by Zhang and Sutton (2011), it is still possible to obtain a valid algorithm by viewing the chain as an M-dimensional Markov chain. In effect, this amounts to using the sample at lag M as the basis for the proposal. Hence, we set $\{\theta', u'\} = \{\theta_{k-M}, u_{k-M}\}$

and $\epsilon_2 = 1$ in the PMH2 proposal in Table 1. Furthermore, in case of a rejection we set $\{\theta_k, u_k\} = \{\theta_{k-M}, u_{k-M}\}$. We refer to Zhang and Sutton (2011) for further details.

The approximate Hessian $\widehat{\mathcal{H}}(\theta' \mid u')$ has to be PSD to be a valid covariance matrix. This can be problematic when the Markov chain is located in areas with low posterior probability or sometimes due to noise in the gradient estimates. In our experience, this happens occasionally in the stationary regime, i.e., after the burn-in phase. However, when it happens $\widehat{\mathcal{H}}(\theta' \mid u')$ is corrected by the hybrid approach discussed by Dahlin et al. (2015a).

Proposal for auxiliary variables

To implement QPMH2, we require estimates of the likelihood and the gradient of the log-posterior. These are obtained by running SMC-ABC which corresponds to simulating the auxiliary variables u. In this section, we show how to estimate the likelihood and its gradient by the fixed-lag (FL; Kitagawa and Sato, 2001) smoother.

SMC-ABC algorithm

SMC-ABC (Jasra et al., 2012) relies on a reformulation of the non-linear SSM (1). We start by perturbing the observations y_t to obtain $\check{y}_{1:T}$ by

$$\check{y}_t = \psi(y_t) + \epsilon z_t, \quad z_t \sim \rho_\epsilon, \quad \text{for } t = 1, \ldots, T, \tag{8}$$

where ψ denotes a one-to-one transformation and ρ_ϵ denotes a kernel, e.g., Gaussian or uniform, with ϵ as the bandwidth or *tolerance parameter*. We continue with assuming that there exists some random variables $v_t \sim \nu_\theta(v_t \mid x_t)$ such that we can generate a sample from $g_\theta(y_t \mid x_t)$ by the transformation $y_t = \tau_\theta(v_t, x_t)$. An example is the *Box-Muller transformation* to obtain a Gaussian random variable from two uniforms, see Appendix A.

To obtain the perturbed SSM, we introduce $\check{x}_t^\top = (x_t^\top, v_t^\top)$ as the new state variable with the dynamics

$$\check{x}_{t+1} \mid \check{x}_t \sim \Xi_\theta(\check{x}_{t+1} \mid \check{x}_t) = \nu_\theta(v_{t+1} \mid x_{t+1}) f_\theta(x_{t+1} \mid x_t), \tag{9a}$$

and the likelihood is modelled by

$$\check{y}_t \mid \check{x}_t \sim h_{\theta,\epsilon}(\check{y}_t \mid \check{x}_t) = \rho_\epsilon(\check{y}_t - \psi(\tau_\theta(\check{x}_t))), \tag{9b}$$

which follows from the perturbation in (8). With this reformulation, we can construct SMC-ABC as outlined in Algorithm 2, which is a standard SMC algorithm applied to the perturbed model. Note that, we do not require any evaluations of the intractable density $g_\theta(y_t \mid x_t)$. Instead, we only simulate from this distribution and compare the simulated and observed (perturbed) data by ρ_ϵ.

The accuracy of the ABC approximation is determined by ϵ, where we recover the original formulation in the limit when $\epsilon \to 0$. In practice, this is not possible and we return to study the impact of a non-zero ϵ in Section 5.1.

Algorithm 2 Sequential Monte Carlo with approximate Bayesian computations

Inputs: $\check{y}_{1:T}$ (perturbed data), the SSM (9), $N \in \mathbb{N}$ (no. particles), $\epsilon > 0$ (tolerance parameter), $\Delta \in [0, T) \subset \mathbb{N}$ (lag).
Outputs: $\widehat{p}_\theta(\check{y}_{1:T} \mid u)$, $\widehat{\mathcal{G}}(\theta \mid u)$ (est. of likelihood and gradient).
Note: all operations are carried out over $i, j = 1, \ldots, N$.

1: Sample $\check{x}_0^{(i)} \sim \mu_\theta(x_0)\nu_\theta(v_0 \mid x_0)$ and set $w_0^{(i)} = 1/N$.
2: **for** $t = 1$ to T **do**
3: Resample the particles by sampling a new ancestor index $a_t^{(i)}$ from a multinomial distribution with $\mathbb{P}(a_t^{(i)} = j) = w_{t-1}^{(j)}$.
4: Propagate the particles by sampling $\check{x}_t^{(i)} \sim \Xi_\theta(\check{x}_t^{(i)} | \check{x}_{t-1}^{a_t^{(i)}})$ and extending the trajectory by $\check{x}_{0:t}^{(i)} = \{\check{x}_{0:t-1}^{a_t^{(i)}}, \check{x}_t^{(i)}\}$.
5: Calculate the particle weights by $\widetilde{w}_t^{(i)} = h_{\theta,\epsilon}(\check{y}_t, \check{x}_t^{(i)})$ which by normalisation (over i) gives $w_t^{(i)}$.
6: **end for**
7: Estimate $\widehat{p}_\theta(\check{y}_{1:T} \mid u)$ by (10) and $\widehat{\mathcal{G}}(\theta \mid u)$ by (11).

Estimation of the likelihood

From Section 2, we require an unbiased estimate of the likelihood to compute the acceptance probability (3). This can be achieved by using u generated by SMC-ABC. In this case, the auxiliary variables $u \triangleq \{\{\check{x}_{0:t}^{(i)}\}_{i=1}^N\}_{t=0}^T$ are the *particle system* composed of all the particles and their trajectories. The resulting likelihood estimator is given by

$$\widehat{p}_\theta(y_{1:T} \mid u) = \prod_{t=1}^{T} \left[\frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \right], \tag{10}$$

where the unnormalised particle weights $\widetilde{w}_t^{(i)}$ are deterministic functions of u. This is an unbiased and N-consistent estimator for the likelihood in the *perturbed* model. However, the perturbation itself introduces some bias and additional variance compared with the original unperturbed model (Dean et al., 2014). The former can result in biased parameter estimates and the latter can result in poor mixing of the Markov chain. We return to study the impact on the mixing numerically in Section 5.1.

Estimation of the gradient of the log-posterior

We also require estimates of the gradient of the log-posterior given u to implement the proposals introduced in Table 1. In Dahlin et al. (2015a), this is accomplished by using the FL smoother together with the *Fisher identity*. However, this requires accurate evaluations of the gradient of $\log g_\theta(y_t \mid x_t)$ with respect to θ. As discussed by Yıldırım et al. (2014), we can circumvent this problem by the reformulation of the SSM in (9) if the gradient of

$\tau_\theta(\check{x}_t)$ can be evaluated. This results in the gradient estimate

$$\widehat{\mathcal{G}}(\theta' \mid u') = \nabla \log p(\theta)\big|_{\theta=\theta'} + \sum_{t=1}^{T} \sum_{i=1}^{N} w_{\kappa_t}^{(i)} \xi_{\theta'}\left(\tilde{z}_{\kappa_t,t}^{(i)}, \tilde{z}_{\kappa_t,t-1}^{(i)}\right), \tag{11}$$

$$\xi_{\theta'}(\check{x}_t, \check{x}_{t-1}) \triangleq \nabla \log \Xi_\theta(\check{x}_t \mid \check{x}_{t-1})\big|_{\theta=\theta'} + \nabla \log h_\theta(\check{y}_t \mid \check{x}_t)\big|_{\theta=\theta'},$$

where $\tilde{z}_{\kappa_t,t}^{(i)}$ denotes the ancestor at time t of particle $\check{x}_{\kappa_t}^{(i)}$ and $\tilde{z}_{\kappa_t,t-1:t}^{(i)} = \{\tilde{z}_{\kappa_t,t-1}^{(i)}, \tilde{z}_{\kappa_t,t}^{(i)}\}$.

The estimator in (11) relies on the assumption that the SSM is mixing quickly, which means that past states have a diminishing influence on future states and observations. More specifically, we assume that $p_\theta(x_t \mid y_{1:T}) \approx p_\theta(x_t \mid y_{1:\kappa_t})$, with $\kappa_t = \min\{t + \Delta, T\}$ and lag $\Delta \in [0, T) \subset \mathbb{N}$. Note that this estimator is biased, but this is compensated for by the accept-reject step in Algorithm 1 and does not effect the stationary distribution of the Markov chain. See Dahlin et al. (2015a) for details.

Numerical illustrations

We evaluate QPMH2 by two illustrations with synthetic and real-world data. In the first model, we can evaluate $g_\theta(y_t \mid x_t)$ exactly in closed-form, which is useful to compare standard PMH and PMH-ABC. In the second model, the likelihood is intractable and therefore only PMH-ABC can be used. See Appendix A for implementation details.

Linear Gaussian SSM

Consider the following LGSS model

$$x_{t+1} \mid x_t \sim \mathcal{N}\left(x_{t+1}; \mu + \phi(x_t - \mu), \sigma_v^2\right), \qquad y_t \mid x_t \sim \mathcal{N}\left(y_t; x_t, 0.1^2\right), \tag{12}$$

with $\theta = \{\mu, \phi, \sigma_v\}$ and $\mu \in \mathbb{R}$, $\phi \in (-1, 1)$ and $\sigma_v \in \mathbb{R}_+$. A synthetic data set consisting of a realisation with $T = 250$ observations is simulated from the model using the parameters $\{0.2, 0.8, 1.0\}$. We begin by investigating the accuracy of Algorithm 2 for estimating the log-likelihood and the gradients of the log-posterior with respect to θ. The error of these estimates are computed by comparing with the true values obtain by a Kalman smoother.

In Figure 1, we present the log-L_1 error of the log-likelihood and the gradients for different values of ϵ. The error in the gradient with respect to ϕ is not presented here, but is similar to the gradients for μ. We see that the error in both the log-likelihood and the gradient are minimized when $\epsilon \approx 0.05 - 0.10$. Here, SMC-ABC achieve almost the same error as standard SMC. However, when ϵ grows larger SMC-ABC suffers from an increasing bias resulting from a deteriorating approximation in (8). We conclude that this results in a bias in the parameter estimates as the bias in the log-likelihood estimate propagates to the parameter posterior estimate.

We now consider estimating the parameters in (12). In this model, we can compare standard PMH with PMH-ABC to study the effects of using ABC to approximate the log-likelihood. For this comparison, we quantify the mixing of the Markov chain using the estimated IACT

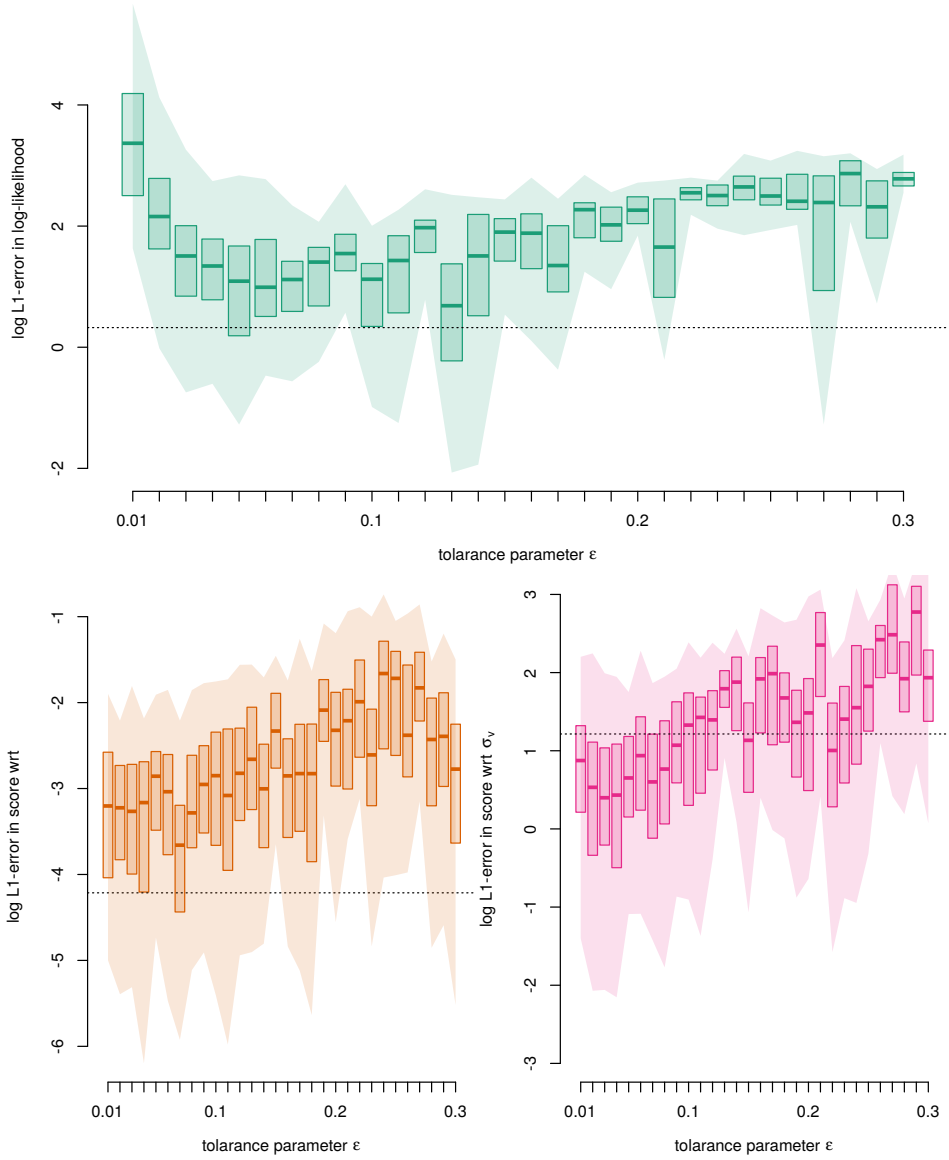


Figure 1. The log-L_1-error in the log-likelihood (top) and gradient with respect to μ (lower left) and σ_v (lower right) using SMC-ABC when varying ϵ. The shaded area and the dotted lines indicate maximum/minimum error and the standard SMC error, respectively. The plots are created from the output of 100 Monte Carlo runs on a single synthetic data set.

			min IACT		max IACT	
	Alg.	Acc.	Median	IQR	Median	IQR
L adapted	PMH0	0.28	12.13	1.53	13.71	1.23
	PMH1	0.78	11.28	0.50	14.50	1.45
	QPMH2	0.55	**3.00**	0.03	**3.01**	0.07
	PMH0-ABC	0.14	29.66	10.37	34.04	9.36
	PMH1-ABC	0.31	33.09	5.45	38.32	14.42
	QPMH2-ABC	0.45	**3.00**	0.02	**3.03**	0.08
$L = 1,000$	PMH0	0.28	7.96	9.60	10.92	6.00
	PMH1	0.78	9.47	4.39	10.60	8.19
	QPMH2	0.55	**5.40**	3.01	**8.98**	6.90
	PMH0-ABC	0.14	12.91	8.86	35.68	3.22
	PMH1-ABC	0.31	27.34	23.63	35.84	30.31
	QPMH2-ABC	0.45	**6.65**	5.14	**10.96**	6.71

Table 2. The IACT values for the LGSS model (12).

given by

$$\widehat{\mathsf{IACT}}(\theta_{K_b:K}) = 1 + 2\sum_{l=1}^{L} \widehat{\rho}_l(\theta_{K_b:K}), \tag{13}$$

where $\widehat{\rho}_l(\theta_{K_b:K})$ denotes the empirical autocorrelation at lag l of $\theta_{K_b:K}$, and K_b is the *burn-in* time. A small value of IF indicates that we obtain many uncorrelated samples from the target distribution. This implies that the chain is mixing well and that σ_φ^2 in (5) is rather small. We make use of two different L to compare the mixing: (i) the smallest L such that $|\widehat{\rho}_L(\theta_{K_b:K})| < 2/\sqrt{K - K_b}$ (i.e., when statistical significant is lost) and (ii) a fixed number of lags $L = 1,000$.

In Table 2, we present the minimum and maximum IACTs as the median and interquartile range (IQR) computed using 10 Monte Carlo runs over the same data set. We note the good performance of the QPMH2 proposal which achieves similar (or better) mixing compared to the pre-conditioned proposals. Remember that PMH0/1 are tuned to their optimal performance using tedious pilot runs, see Sherlock et al. (2015) and Nemeth et al. (2014). We present some additional diagnostic plots (trace, autocorrelation and posterior estimates) for PMH0 and QPMH2 (without ABC) in Appendix C.

In our experience, the performance of QPMH2 seems to be connected with the variance of $\widehat{\mathcal{G}}(\theta'|u')$. This is similar to the existing theoretical results for PMH1 (Nemeth et al., 2014). For the LGSS model, we can compute the gradient with a small variance and therefore QPMH2 performs well. However, this might not be the case for all models and the performance of QPMH2 thus depends on both the model and which particle smoother is applied to estimate the gradient. Finally, note that using ABC results in a smaller acceptance

rate and worse mixing. This problem can probably be mitigated by adjusting ϵ and N as discussed by Bornn et al. (2014) and Jasra (2015).

Modelling the volatility in coffee futures

Consider the problem of modelling the volatility of the log-returns of future contracts on coffee using the $T = 399$ observations in Figure 2. A prominent feature in this type of data is jumps (present around March, 2014). These are typically the result of sudden changes in the market due to e.g., news arrivals and it has been proposed to make use of an α-stable distribution to model these jumps, see Lombardi and Calzolari (2009) and Dahlin et al. (2015c). Therefore, we consider a stochastic volatility model with symmetric α-stable returns (αSV) given by

$$x_{t+1} \mid x_t \sim \mathcal{N}\left(x_{t+1}; \mu + \phi(x_t - \mu), \sigma_v^2\right), \qquad y_t \mid x_t \sim \mathcal{A}\left(y_t; \alpha, \exp(x_t)\right), \tag{14}$$

with $\theta = \{\mu, \phi, \sigma_v, \alpha\}$. Here, $\mathcal{A}(\alpha, \eta)$ denotes a symmetric α-stable distribution with *stability parameter* $\alpha \in (0, 2)$ and *scale parameter* $\eta \in \mathbb{R}_+$. As previously discussed, we cannot evaluate $g_\theta(y_t \mid x_t)$ for this model but it is possible to simulate from it. See Appendices A and D for more details regarding the α-stable distribution and methods for simulating random variables.

In Figure 3, we present the posterior estimates obtained from QPMH2, which corresponds to the estimated parameter posterior mean $\widehat{\theta} = \{0.214, 0.931, 0.268, 1.538\}$. This indicates a slowly varying log-volatility with heavy-tailed log-returns as the Cauchy and Gaussian distribution corresponds to $\alpha = 1$ and $\alpha = 2$, respectively. We also compare with the posterior estimates from PMH0, which are quite similar in location and scale.

Finally, we present the smoothed estimate of the log-volatility using $\widehat{\theta}$ in Figure 2. The estimate seems reasonable and tracks the periods with low volatility (around October, 2013) and high volatility (around May, 2014).

Conclusions

We have demonstrated that QPMH2 exhibits similar or improved mixing when compared with pre-conditioned proposals with/without the ABC approximation. The main advantage is that QPMH2 does not require extensive tuning of the step sizes in the proposal to achieve good mixing, which can be a problem in practice for PMH0/1. The user only needs to choose δ and M, which in our experience are simpler to tune. Finally, QPMH2 only requires gradient information, which are usually simpler to obtain than directly estimate the Hessian of the log-posterior.

In future work, it would be interesting to analyse the impact of the variance of the gradient estimates on the mixing of the Markov chain in QPMH2. In our experience, QPMH2 performs well when the variance is small or moderate. However when the variance increases, the mixing can be worse than for PMH0. This motivates further theoretical study and development of better particle smoothing techniques for gradient estimation to obtain gradient estimates with lower variance.

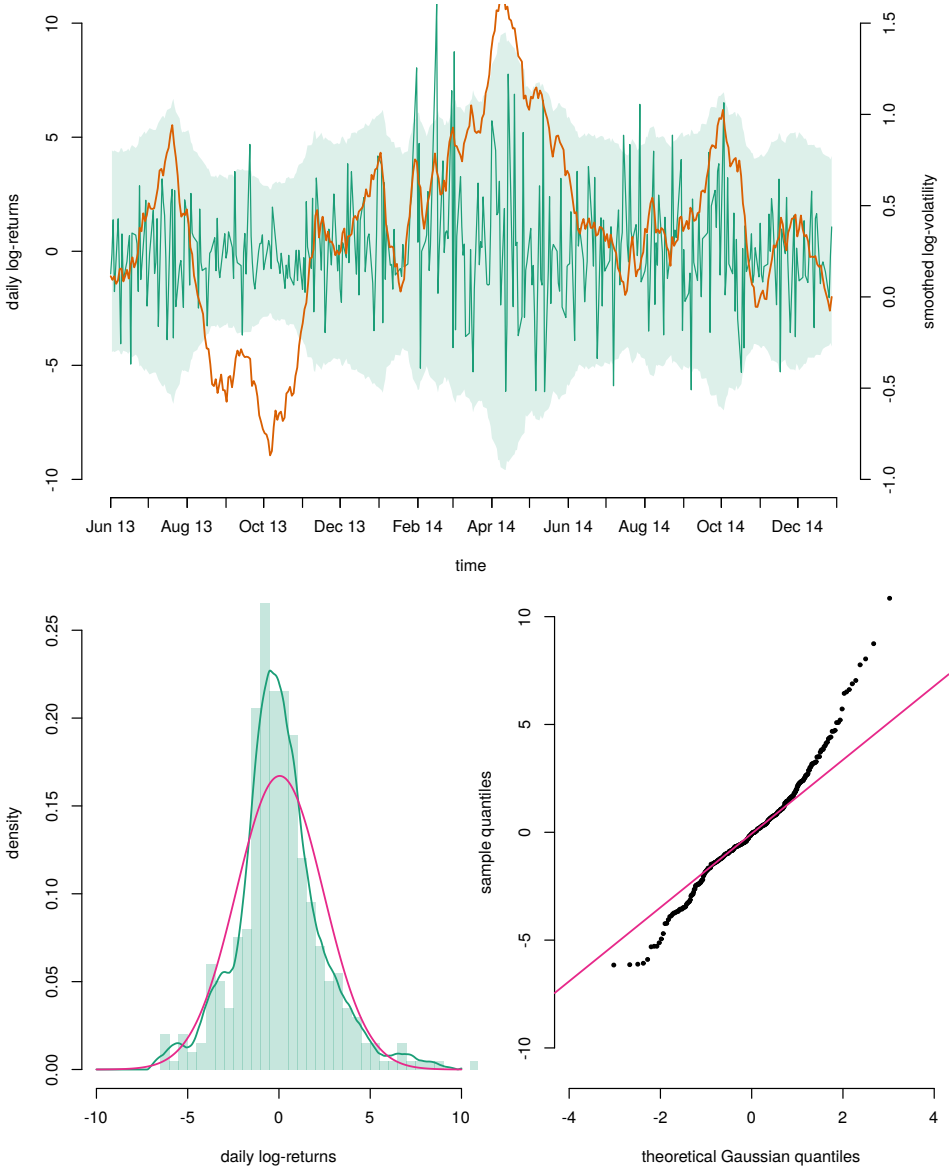


Figure 2. Upper: log-returns (green) and smoothed log-volatility (orange) of futures on coffee between June 1, 2013 and December 31, 2014. The shaded area indicate the approximate 95% credibility region for the log-returns estimated from the model. Lower left: kernel density estimate (green) and a Gaussian approximation (magenta). Lower right: QQ-plot comparing the quantiles of the data with the best Gaussian approximation (magenta).

Another extension of this work is to consider models where p is considerable larger than discussed in this paper. This would probably lead to an even greater increase in mixing when using QPMH2 compared with PMH0. This effect is theoretically and empirically examined by Nemeth et al. (2014).

In this context, it would also be interesting to implement a quasi-Newton proposal in a particle Hamiltonian Monte Carlo (HMC) algorithm in the spirit of Zhang and Sutton (2011). This as HMC algorithms are known to greatly increase mixing for some models when the gradient and log-likelihood can be evaluated analytically. However, no particle version of HMC has yet been proposed in the literature but the possibility is considered in the discussions following Girolami and Calderhead (2011).

The source code and data for the LGSS model as well as some supplementary material are available from `https://github.com/compops/qpmh2-sysid2015/`.

Acknowledgements

The simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at Linköping University, Sweden.

Appendix

Implementation details

In Algorithm 2 for the LGSS model, we use a fully adapted SMC algorithm with $N = 50$ and SMC-ABC with $N = 2,500$, lag $\Delta = 12$, $\epsilon = 0.10$ and ρ_ϵ as the Gaussian density with standard deviation ϵ. For the αSV, we use the same settings expect for $N = 5,000$. For QPMH2, we use the memory length $M = 100$, $\delta = 1,000$ and $n_{\text{hyb}} = 2,500$ samples for the hybrid method. We use $K = 15,000$ iterations (discarding the first $K_b = 5,000$ as burn-in) for all PMH algorithms and initialise in the maximum a posteriori (MAP) estimate obtained accordingly to Dahlin et al. (2015c).

The pre-conditioning matrix $\mathcal{P}$ is estimated by pilot runs using PMH0, with step sizes based on the Hessian estimate obtained in the MAP estimation. The final step sizes are given by the rules of thumb by Sherlock et al. (2015) and Nemeth et al. (2014), i.e.,

$$\epsilon_0^2 = 2.562^2 p^{-1}, \qquad \epsilon_1^2 = 1.125^2 p^{-1/3}.$$

Finally, we use the following prior densities

$$\begin{aligned} p(\mu) &\sim \mathcal{TN}_{(0,1)}(\mu; 0, 0.2^2), & p(\phi) &\sim \mathcal{TN}_{(-1,1)}(\phi; 0.9, 0.05^2), \\ p(\sigma_v) &\sim \mathcal{G}(\sigma_v; 0.2, 0.2), & p(\alpha) &\sim \mathcal{B}(\alpha/2; 6, 2), \end{aligned}$$

where $\mathcal{TN}_{(a,b)}(\cdot)$ denotes a truncated Gaussian distribution on $[a, b]$, $\mathcal{G}(a, b)$ denotes the Gamma distribution with mean a/b and $\mathcal{B}(a, b)$ denotes the Beta distribution.

For SMC-ABC, we require the transformation $\tau(\overset{\circ}{x}_t)$ to simulate random variables from the two models. For the LGSS model, we use the identity transformation $\psi(x) = x$ and the

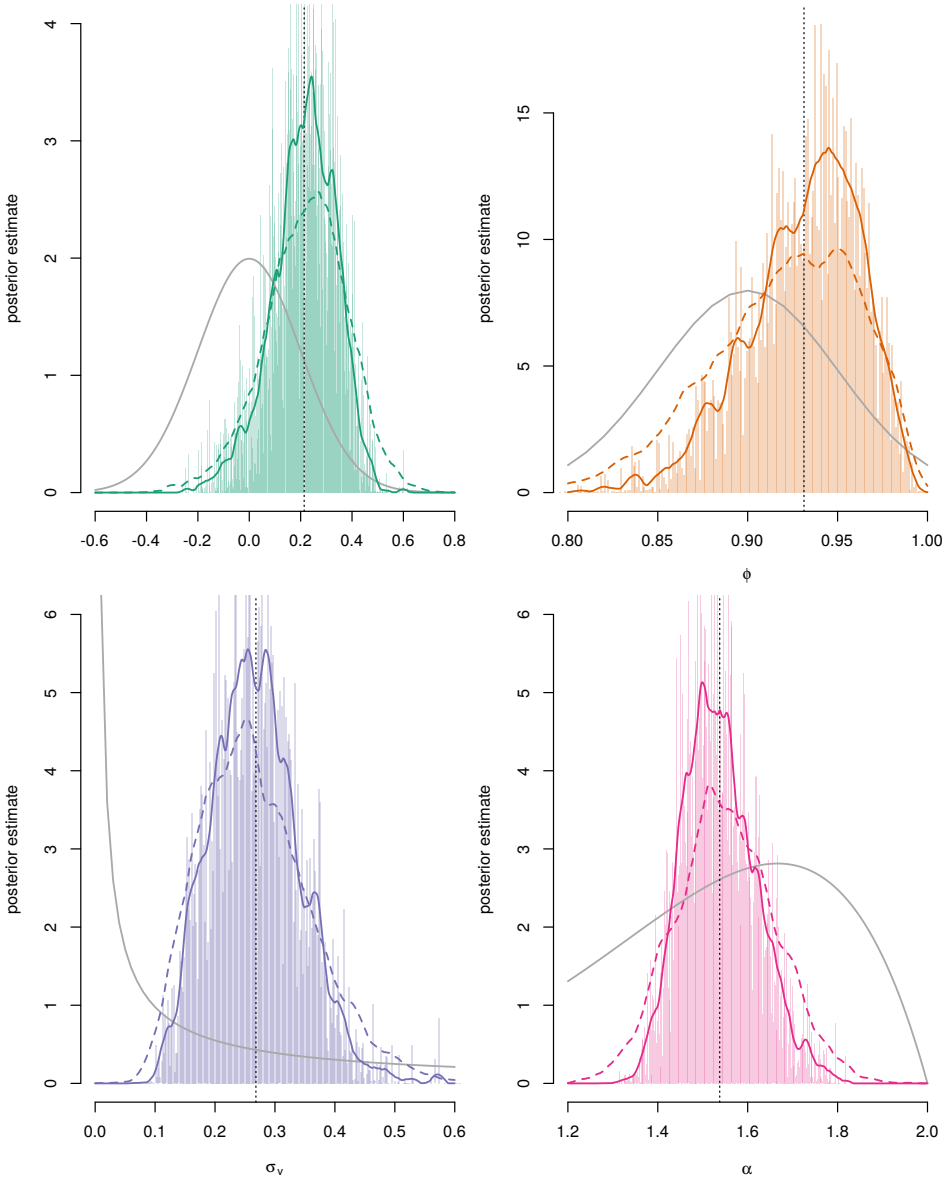


Figure 3. Parameter posteriors for (14) for μ (green), ϕ (orange), σ_v (purple) and α (magenta) obtained by pooling the output from 10 runs using QPMH2. The dashed lines indicate the corresponding estimates from PMH0. Dotted lines and grey densities indicate the estimate of the posterior means and the prior densities, respectively.

Box-Muller transformation to simulate y_t by

$$\tau_\theta(\check{x}_t) = x_t + \sigma_e\sqrt{-2\log v_{t,1}}\cos(2\pi v_{t,2}),$$

where $\{v_{t,1}, v_{t,2}\} \sim \mathcal{U}[0,1]$.

For the αSV model, we use $\psi(x) = \arctan(x)$ as proposed by Yıldırım et al. (2014) to make the variance in the gradient estimate finite. We generate samples from $\mathcal{A}(\alpha, \exp(x_t))$ for the stability parameter $\alpha \neq 1$ by

$$\tau_\theta(\check{x}_t) = \exp(x_t/2)\frac{\sin(\alpha v_{t,2})}{[\cos(v_{t,2})]^{1/\alpha}}\left[\frac{\cos\left[(\alpha-1)v_{t,2}\right]}{v_{t,1}}\right]^{\frac{1-\alpha}{\alpha}},$$

where $\{v_{t,1}, v_{t,2}\} \sim \{\mathbf{Exp}(1), \mathcal{U}(-\pi/2, \pi/2)\}$. The real-world data in the αSV model is computed as $y_t = 100[\log(s_t) - \log(s_{t-1})]$, where s_t denotes the price of a future contract on coffee obtained from `https://www.quandl.com/CHRIS/ICE_KC2`.

Implementation details for quasi-Newton proposal

The quasi-Newton proposal adapts the Hessian estimate at iteration k using the previous M states of the Markov chain. Hence, the current proposed parameter depends on the previous M states and therefore this can be seen as a Markov chain of order M. Following Zhang and Sutton (2011), we can analyse this chain as a first-order Markov chain on an extended M-fold product space Θ^M. This results in that the stationary distribution,

$$\boldsymbol{\pi}(\theta_{1:M}) = \prod_{i=1}^{M}\pi(\theta_i),$$

where $\pi(\theta_{k,i})$ is defined as in (2). To proceed with the analysis, we introduce the notation $\psi_k = \{\theta_k, u_k\}$ and $\psi_{k,1:M\setminus i}$ for the vector $\psi_{k,1:M}$ with $\psi_{k,i}$ removed for brevity. We can then update a component of $\psi_{k,1:M}$ using some transition kernel T_i defined by

$$T_i\left(\psi_i, \psi_i'|\psi_{k,1:M\setminus i}\right) = \delta\left(\psi_{k,1:M\setminus i}, \psi'_{k,1:M\setminus i}\right)B\left(\psi_i, \psi_i'|\psi_{k,1:M\setminus i}\right),$$

where $B\left(\psi_i, \psi_i'|\psi_{k,1:M\setminus i}\right)$ denotes the QPMH2 proposal adapted using the last $M-1$ samples $\psi_{k,1:M\setminus i}$, analogously to (7). This is similar to a Gibbs-type step in which we update a component conditional on the remaining $M-1$ components. Here, this conditioning is used to construct the local Hessian approximation using the information in the remaining components. As this update leaves $\pi(\theta_{k,i})$ invariant, it follows that

$$T\left(\psi_{k,1:M}, \psi'_{k,1:M}\right) = T_1 \circ T_2 \circ \cdots \circ T_M\left(\psi_{k,1:M}, \psi'_{k,1:M}\right),$$

leaves $\boldsymbol{\pi}(\theta_{k,1:M})$ invariant. Hence, we can update all components of the M-dimensional Markov chain during each iteration k of the sampler.

To implement the algorithm, we instead interpret the proposal as depending on the last $M-1$ states of the Markov chain. This results in a sliding window of samples, which we use to adapt the proposal. This results in two minor differences from a standard PMH algorithm.

Algorithm 3 Quasi-Newton proposal

Inputs: $\{\theta_{k-M:k-1}, u_{k-M:k-1}\}$ (last M states of the Markov chain), $\delta > 0$ (initial Hessian).
Outputs: θ' (proposed parameter).

Extract the $M^\star$ unique elements from $\{\theta_{k-M+1:k-1}, u_{k-M+1:k-1}\}$ and sort them in ascending order (with respect to the log-likelihood) to obtain $\{\theta^\star, u^\star\}$.
if $M^\star \geq 2$ **then**
  Calculate s_l and y_l for $l = 1, \ldots, M-2$ using the ordered pairs in $\{\theta^\star, u^\star\}$ and their corresponding gradient estimates.
  Initialise the Hessian estimate $B_1^{-1} = \rho_1^{-1}(y_1^\top y_1)^{-1}\mathbf{I}_p$.
  for $l = 1$ to $M^\star - 1$ **do**
    Carry out the update (7) to obtain B_{l+1}^{-1}.
  end for
  Set $\Sigma_{\text{BFGS}}(\psi_{k-M+1:k-1}) = -B_{M^\star}(\theta')$.
else
  Set $\Sigma_{\text{BFGS}}(\psi_{k-M+1:k-1}) = \delta\mathbf{I}_p$.
end if
Sample from (15) to obtain θ'.

The first is that we center the proposal around the position of the Markov chain at $k - M$,

$$q(\theta' \,|\, \psi_{k-M+1:k-1}) = \mathcal{N}\big(\theta'; \theta_{k-M}, \Sigma_{\text{BFGS}}(\psi_{k-M+1:k-1})\big), \tag{15}$$

where $\Sigma_{\text{BFGS}}(\psi_{k-M+1:k-1}) = -B_M(\theta')$ obtain by iterating (7). The second difference is that we set $\{\theta_k, u_k\} \leftarrow \{\theta_{k-M}, u_{k-M}\}$ is the candidate parameter θ' is rejected. The complete procedure for proposing from $q(\theta'_k \,|\, \psi_{k-M+1:k-1})$ is presented in Algorithm 3.

Some possible extensions to this noisy quasi-Newton inspired update are discussed by Schraudolph et al. (2007) and Zhang and Sutton (2011). These include how to rearrange the update such that the Hessian estimate always is a PSD matrix. In this paper, we instead make use of the hybrid method discussed by Dahlin et al. (2015a) to handle these situations. In Schraudolph et al. (2007), the authors also discuss possible alternations to handle noisy gradients, which could be useful in some situations. In our experience, these alternations do not always improve the quality of the Hessian estimate as the noisy is stochastic and not the result of a growing data set as in the aforementioned paper.

Additional results

In this section, we present some additional plots for the LGSS example in Section 5.1 using PMH0 in Figure 4 and QPMH2 in Figure 5. Note that the mixing is better for QPMH2 as the ACF decreases quicker as the lag increased compared with PMH0. However due to the quasi-Newton proposal, we obtain a large correlation coefficient at lag M. This is also reflected in the trace plots, which exhibits a periodic behaviour. Hence, we conclude that the mixing is increased in some sense when using QPMH2 compared with PMH0. The exact magnitude of this improvement depends on the method to compute the IACT values.

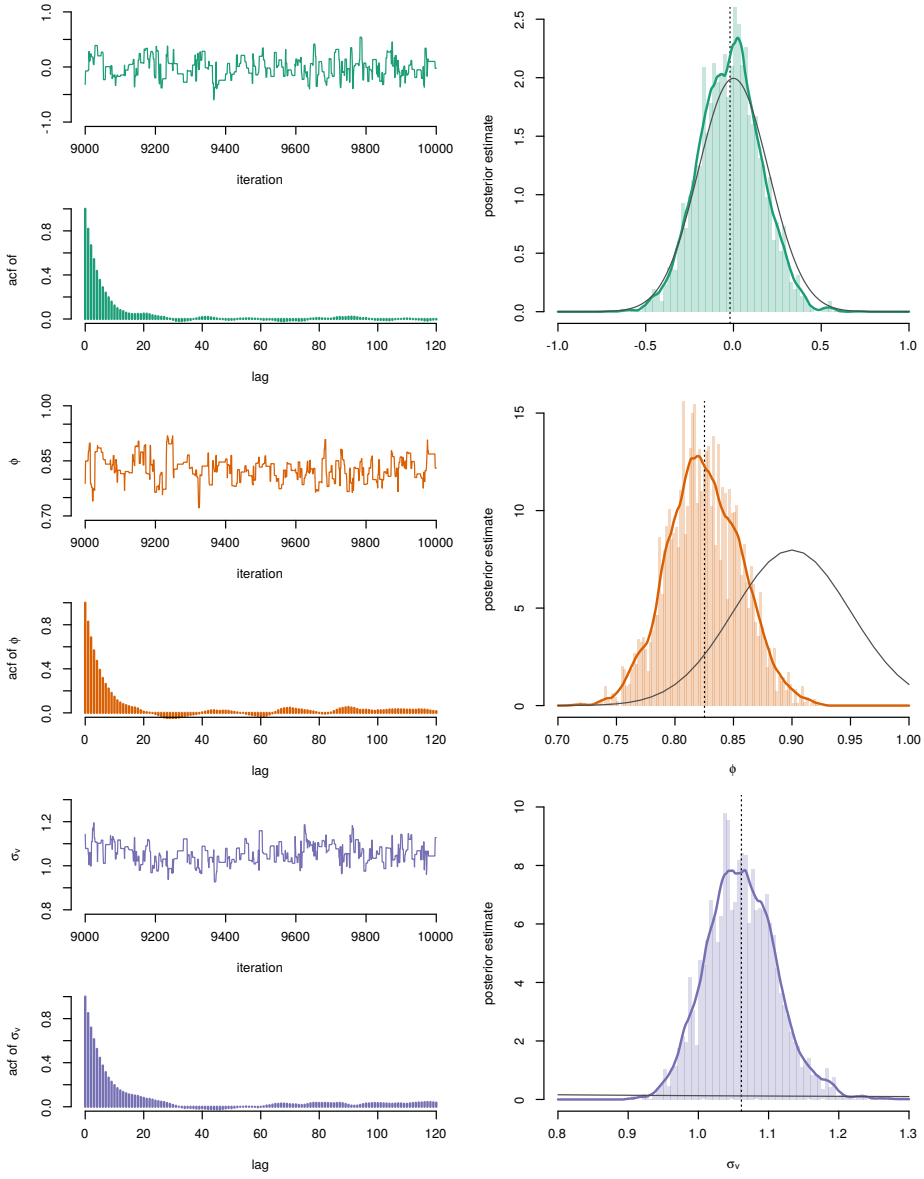


Figure 4. Trace plots and ACF estimates (left) for μ (green), ϕ (orange) and σ_v (purple) in the LGSS model using PPMHO-ABC. The resulting posterior estimates (right) are also presented as histograms and kernel density estimates. The dotted vertical lines in the estimates of the posterior indicate its estimated mean. The grey lines indicate the parameter prior distributions.

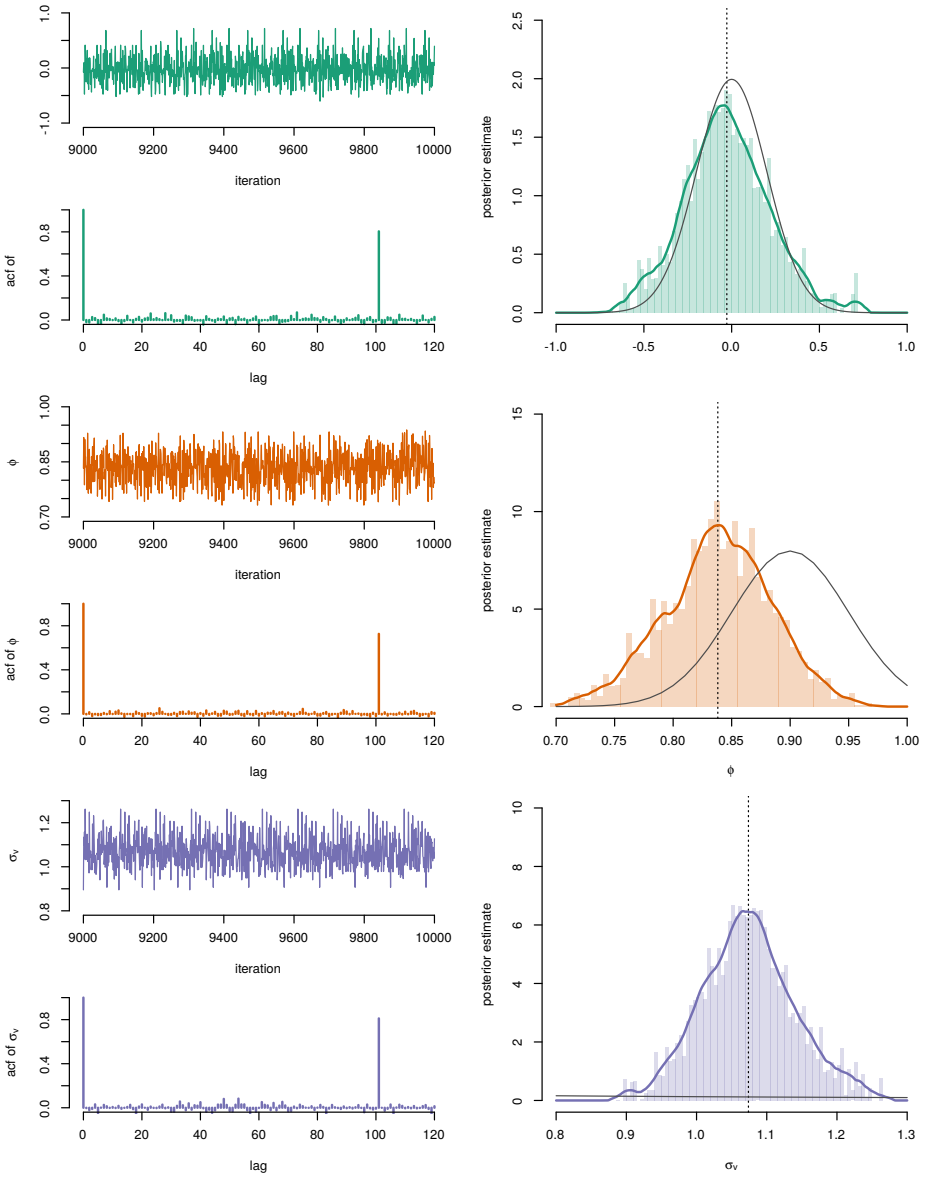


Figure 5. Trace plots and ACF estimates (left) for μ (green), ϕ (orange) and σ_v (purple) in the LGSS model using QPMH2-ABC. The resulting posterior estimates (right) are also presented as histograms and kernel density estimates. The dotted vertical lines in the estimates of the posterior indicate its estimated mean. The grey lines indicate the parameter prior distributions.

α-stable distributions

This appendix summarises some important results regarding α-stable distributions. For a more detailed presentation, see Nolan (2003), Peters et al. (2012) and references therein.

Definition 1 (α-stable distribution (Nolan, 2003)). An univariate α-stable distribution denoted by $\mathcal{A}(\alpha, \beta, c, \mu)$ has the characteristic function

$$\phi(t;\alpha,\beta,c,\mu) = \begin{cases} \exp\left\{it\mu - |ct|^\alpha \left[1 - i\beta\tan\left(\frac{\pi\alpha}{2}\right)\mathsf{sign}(t)\right]\right\} & \text{if } \alpha \neq 1, \\ \exp\left\{it\mu - |ct|^\alpha \left[1 + \frac{2i\beta}{\pi}\mathsf{sign}(t)\ln|t|\right]\right\} & \text{if } \alpha = 1, \end{cases}$$

where $\alpha \in [0, 2]$ denotes the stability parameter, $\beta \in [-1, 1]$ denotes the skewness parameters, $c \in \mathbb{R}_+$ denotes the scale parameter and $\mu \in \mathbb{R}$ denotes the location parameter.

The α-stable distribution is typically defined through its characteristic function given in Definition 1. Except for some special cases, we cannot recover the probability distribution function (pdf) from $\phi(t;\alpha,\beta,c,\mu)$ as it cannot be computed analytically. These exceptions are; (i) the Gaussian distribution $\mathcal{N}(\mu, 2c^2)$ is recovered when $\alpha = 2$ for any β (as the α-stable distribution is symmetric for this choice of α), (ii) the Cauchy distribution $\mathcal{C}(\mu, c)$ is recovered when $\alpha = 1$ and $\beta = 0$, and (iii) the Lévy distribution $\mathcal{L}(\eta, c)$ is recovered when $\alpha = 0.5$ and $\beta = 1$.

For all other choices of α and β, we cannot recover the pdf from $\phi(t;\alpha,\beta,c,\mu)$ due to the analytical intractability of the Fourier transform. However, we can often approximate the pdf using numerical methods as discussed in Peters et al. (2012). In this work, we make use of another approach based on ABC approximations to circumvent the intractability of the pdf. For this, we require to be able to sample from $\mathcal{A}(\alpha,\beta,c,\mu)$ for any parameters. An procedure for this discussed by Chambers et al. (1976) is presented in Proposition 2.

Proposition 2 (Simulating α-stable variable (Chambers et al., 1976)). *Assume that we can simulate $w \sim \mathcal{E}xp(1)$ 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 $\mathcal{A}(\alpha,\beta,c,\mu)$ is obtained by the transformation

$$y = \begin{cases} c\bar{y} + \mu & \text{if } \alpha \neq 1, \\ c\bar{y} + \left(\mu + \beta\frac{2}{\pi}c\log c\right) & \text{if } \alpha = 1. \end{cases}$$

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, 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.

L. Bornn, N. Pillai, A. Smith, and D. Woordward. A pseudo-marginal perspective on the abc algorithm. *Pre-print*, 2014. arXiv:1404.6298v1.

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.

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015a.

J. Dahlin, F. Lindsten, and T. B. Schön. Quasi-Newton particle Metropolis-Hastings. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 981–986, Beijing, China, October 2015b.

J. Dahlin, M. Villani, and T. B. Schön. Efficient approximate Bayesian inference for models with intractable likelihoods. *Pre-print*, 2015c. arXiv:1506.06975v1.

T. A. Dean, S. S. Singh, A. Jasra, and G. W. Peters. Parameter estimation for hidden Markov models with intractable likelihoods. *Scandinavian Journal of Statistics*, 41(4): 970–987, 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.

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. Jasra. Approximate Bayesian computation for a class of time series models. *International Statistical Review*, 83(3):405–435, 2015.

A. Jasra, S. S. Singh, J. S. Martin, and E. McCoy. Filtering via approximate Bayesian computation. *Statistics and Computing*, 22(6):1223–1237, 2012.

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 Verlag, 2001.

M. J. Lombardi and G. Calzolari. Indirect estimation of α-stable stochastic volatility models. *Computational Statistics and Data Analysis*, 53(6):2298–2308, 2009.

J-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

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

C. Nemeth, C. Sherlock, and P. Fearnhead. Particle Metropolis adjusted Langevin algorithms. *Pre-print*, 2014. arXiv:1412.7299v1.

J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

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

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

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.

N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-Newton method for online convex optimization. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, San Juan, Puerto Rico, March 2007.

C. Sherlock, A. H. Thiery, G. O. Roberts, and J. S. Rosenthal. On the efficency of pseudo-marginal random walk Metropolis algorithms. *The Annals of Statistics*, 43(1):238–275, 2015.

S. Yıldırım, S. S. Singh, T. Dean, and A. Jasra. Parameter estimation in hidden Markov models with intractable likelihoods using sequential Monte Carlo. *Journal of Computational and Graphical Statistics*, 24(3):846–865, 2014.

Y. Zhang and C. A. Sutton. Quasi-Newton methods for Markov chain Monte Carlo. In *Proceedings of the 2011 Conference on Neural Information Processing Systems (NIPS)*, Granada, Spain, December 2011.

Paper D

Accelerating pseudo-marginal Metropolis-Hastings using correlated likelihood estimators

Authors: J. Dahlin, F. Lindsten, J. Kronander and T. B. Schön

Edited version of the paper:

J. Dahlin, F. Lindsten, J. Kronander, and T. B. Schön. Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables. *Pre-print*, 2015a. arXiv:1512.05483v1.

Accelerating pseudo-marginal Metropolis-Hastings using correlated likelihood estimators

J. Dahlin[⋆], F. Lindsten[†], J. Kronander[‡] and T. B. Schön[†]

[⋆]Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden.
johan.dahlin@liu.se

[†]Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
fredrik.lindsten@it.uu.se
thomas.schon@it.uu.se

[‡]Dept. of Science and Technology,
Linköping University,
SE-581 85 Linköping, Sweden.
joel.kronander@liu.se

Abstract

Pseudo-marginal Metropolis-Hastings (PMMH) is a powerful method for Bayesian inference in models where the posterior distribution is analytical intractable or computationally costly to evaluate directly. It operates by introducing additional auxiliary variables into the model and form an extended target distribution, which then can be evaluated point-wise. In many cases, the standard Metropolis-Hastings is then applied to sample from the extended target and the sought posterior can be obtained by marginalisation. However, in some implementations this approach suffers from poor mixing as the auxiliary variables are sampled from an independent proposal. We propose a modification to the PMMH algorithm in which a Crank-Nicolson (CN) proposal is used instead. This results in that we introduce a positive correlation in the auxiliary variables. We investigate how to tune the CN proposal and its impact on the mixing of the resulting PMMH sampler. The conclusion is that the proposed modification can have a beneficial effect on both the mixing of the Markov chain and the computational cost for each iteration of the PMMH algorithm.

Keywords

Bayesian inference, pseudo-marginal algorithms, auxiliary variables, Crank-Nicolson, particle filtering, importance sampling.

Data and source code in Python

https://github.com/compops/pmmh-correlated2015

Financial support from

The projects *Learning of complex dynamical systems* (Contract number: 637-2014-466), *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) and CADICS, a Linnaeus Center, all funded by the Swedish Research Council. The Swedish Foundation for Strategic Research (SSF) through grant IIS11-0081.

Introduction

We are interested in approximating some probability distribution $\bar{\pi}(\theta, u)$ using simulation methods based on Markov chain Monte Carlo (MCMC; Robert and Casella, 2004). In Bayesian inference, $\bar{\pi}(\theta, u)$ could represent the posterior distribution of the parameters $\theta \subset \Theta$ and some auxiliary variables $u \subset \mathcal{U}$, which e.g., can be latent states in the model or missing data. Often, we are interested in the marginal *posterior distribution of the parameters* given by

$$\pi_\theta(\theta) = \frac{p(\theta)p(y\,|\,\theta)}{\int_\Theta p(\theta)p(y\,|\,\theta)\mathrm{d}\theta} = \frac{p(\theta)p_\theta(y)}{p(y)}, \tag{1}$$

where $p_\theta(y) \triangleq p(y\,|\,\theta)$ denotes the *likelihood function* and $p(\theta)$ denotes the *parameter prior distribution*. The denominator $p(y)$ is usually referred to as the *marginal likelihood* or the model evidence.

In principle, we can apply MCMC methods to sample directly from (1) using e.g., the Metropolis-Hastings (MH; Metropolis et al., 1953; Hastings, 1970) algorithm. However, in practice $\pi_\theta(\theta)$ can be analytically intractable or too costly from a computationally perspective to evaluate point-wise. By introducing the auxiliary variables u, we can sometimes mitigate these problems and obtain an algorithm which can be of practical use.

This approach is discussed by Andrieu and Roberts (2009), where the pseudo-marginal MH (PMMH) algorithm is introduced to sample from $\bar{\pi}(\theta, u)$ using a standard MH sampler. This approach can be seen as an *exact approximation* of the ideal algorithm, where $\pi_\theta(\theta)$ can be recovered by marginalisation. A concrete example is the particle MH (PMH; Andrieu et al., 2010), where the auxiliary variables are all the random variables generated in a run of a sequential Monte Carlo (SMC; Del Moral et al., 2006) algorithm. In this setting, the SMC algorithm is used to provide an unbiased estimate of the likelihood function. This is useful for inference in latent variables models such as state space models (SSMs; Flury and Shephard, 2011; Pitt et al., 2012), mixture models (Fearnhead and Meligkotsidou, 2015) and generalised linear mixed models (Tran et al., 2014).

A typical problem in PMMH is that the resulting Markov chain can be highly autocorrelated, which corresponds to a sticky chain and poor exploration of the posterior. This is the result of that occasionally we obtain an estimate of the posterior which is much larger than its true value. This problem can typically be mitigated by increasing the number of auxiliary variables N_u, which increase the accuracy of the estimate of the target. However, this increases the computational cost for each iteration of the algorithm, which can limit practical use of the PMMH algorithm for many interesting but challenging models. The trade-off between increasing N_u and the number of iterations K in the PMMH algorithm is studied by Pitt et al. (2012), Sherlock et al. (2015) and Doucet et al. (2015). They conclude that N_u should be selected such that the standard deviation in the log-target estimate is roughly between 1 and 2.

The main contribution of this paper is to introduce a positive correlation in the auxiliary variables u between two consecutive iterations of the PMMH algorithm. Intuitively, this allows us to decrease N_u and therefore the computational cost while keeping the mixing of

the Markov chain constant. The correlation is introduced by changing the proposal for u from the standard independent proposal to a random walk proposal in the space $\mathcal{U}$.

The idea for this originates in the field of computer graphics, where a similar approach is used for rendering images by sampling light paths connecting the light sources in the scene to the camera using MCMC methods. In Kelemen et al. (2002), the authors propose the primary sample path space Metropolis light transport algorithm that operates directly on a set of uniform variates used in a second step to construct a light ray through the scene, using sequential importance sampling. Recently, Hachisuka et al. (2014) extended the original algorithm to include multiple importance sampling (MIS; Owen and Zhou, 2000; Veach and Guibas, 1997; Kronander and Schön, 2014) to improve the efficiency further.

The idea of introducing correlation in u between iterations has previously been suggested (independently) in the original particle MCMC (PMCMC) paper by Andrieu et al. (2010) and in the connected discussions by Lee and Holmes (2010). Furthermore, a similar idea of introducing correlation in u is proposed by Andrieu et al. (2012). The main difference in our contribution is the use of the Crank-Nicolson (SN; Beskos et al., 2008; Cotter et al., 2013; Hairer et al., 2014) proposal to update u at every iteration of the algorithm. This in contrast with the standard PMMH proposal for u, which samples new random variables independently at each iteration of the algorithm. The CN proposal is a natural choice as it is known to scale independently with the dimension of the space. Furthermore, we provide the reader with both a theoretical and numerical analysis of how the mixing and the computational cost is affected and propose an adaptive algorithm based on the theoretical results. Deligiannidis et al. (2015) also proposes (independently) to make use of the CN proposal in this setting.

We consider two different numerical examples to illustrate the properties of the proposed modifications to the PMMH algorithm. In the first example, we infer the parameters in IID data from the Gaussian distribution to analyse how to tune the CN proposal and show how it improves upon the idea proposed by Lee and Holmes (2010). In the second example, we conduct inference of the log-volatility using a stochastic volatility (SV) model with leverage on real-world stock index data.

We continue this paper by introducing the proposed modifications in Section 2 and analyse its theoretical properties, in a simplified setting, in Section 3. We end the paper by some numerical illustrations of the proposed method in Section 4 from which we draw some general conclusions in Section 5.

Introducing correlation into the auxiliary variables

We would like to sample from the parameter posterior $\pi_\theta(\theta)$ in (1). However, this is not possible using a direct implementation of the MH algorithm as we cannot evaluate $\pi_\theta(\theta)$ point-wise. Instead, we introduce the auxiliary variables $u = (u_1, \ldots, u_{N_u})$ as N_u independent standard Gaussian random variables, i.e., $u_i \sim \mathcal{N}(0, 1)$. We make use of the setup used by Andrieu and Roberts (2009) and Doucet et al. (2015). Furthermore, we make no notational distinction between a random variable and its realisation for brevity.

Let the *potential function* $\Phi_\theta(u) : \Theta \times \mathbb{R}^{N_u} \to \mathbb{R}$ be defined such that

$$\mathbb{E}\Big[\exp\big(-\Phi_\theta(u)\big)\Big] = c\,\pi_\theta(\theta), \qquad \forall\,\theta \in \Theta,$$

and some constant $c > 0$. Hence, we can define the *extended target distribution* as

$$\bar{\pi}(\theta, u) = \frac{\exp\big(-\Phi_\theta(u)\big)}{c}\,\mathcal{N}\big(u; 0, \mathbf{I}_{N_u}\big), \tag{2}$$

where $\mathcal{N}(u; 0, \mathbf{I}_{N_u})$ denotes a N_u-dimensional multivariate standard Gaussian distribution. Note that the parameter posterior is the marginal of $\bar{\pi}(\theta, u)$ as

$$\int_{\mathbb{R}^{N_u}} \bar{\pi}(\theta, u)\,\mathrm{d}u = \frac{1}{c}\mathbb{E}\Big[\exp\big(-\Phi_\theta(u)\big)\Big] = \pi_\theta(\theta),$$

as required in the exact approximation scheme used in the PMMH algorithm. Hence, we conclude that this is a valid *extended target* for sampling from the parameter posterior $\pi_\theta(\theta)$. A concrete example of a suitable potential function follows from the definition in (1). In this case, we have

$$\begin{aligned}
\Phi_\theta(u) &= -\Big(\log \widehat{p}_\theta(y; u) + \log p(\theta)\Big), \\
\exp\big(-\Phi_\theta(u)\big) &= \widehat{p}_\theta(y; u)p(\theta), \\
\mathbb{E}\Big[\exp\big(-\Phi_\theta(u)\big)\Big] &= p_\theta(y)p(\theta) = \underbrace{p(y)}_{=c}\,\pi_\theta(\theta),
\end{aligned}$$

where the constant c corresponds to the marginal likelihood and $\widehat{p}_\theta(y; u)$ denotes the non-negative and unbiased estimator of the likelihood constructed by the auxiliary variables. (The typical application of the PMMH algorithm makes use of importance sampling or particle filtering to construct such an estimator.)

From (2), we have that the target is high-dimensional in u whenever N_u is large, and that the extended target correspond to a change of measure from the Gaussian prior. In this setting, we know that the CN proposal is a suitable choice as it is dimension independent (Cotter et al., 2013). An intuition for this can be given by realising that the CN proposal is a discretisation of the Ornstein-Uhlenbeck stochastic differential equation, see e.g., Kloeden and Platen (1992). The main difference between using the CN proposal and a standard random walk proposal is its autoregressive nature, which results in the mean-reverting behaviour. As a consequence, the CN proposal has the standard Gaussian distribution as its limiting distribution, which suits us well in this setting. Therefore, we assume a proposal distribution for u and θ with the form

$$\begin{aligned}
q_{\theta,u}\big(\{\theta', u'\} \,|\, \{\theta, u\}\big) &= q_\theta\big(\theta' \,|\, \{\theta, u\}\big) q_u\big(u' \,|\, u\big) \\
&= \mathcal{N}\big(\theta'; \mu(\theta, u), \Sigma(\theta, u)\big)\,\mathcal{N}\big(u'; \sqrt{1 - \sigma_u^2}u, \sigma_u^2 \mathbf{I}_{N_u}\big).
\end{aligned} \tag{3}$$

This corresponds to using a standard Gaussian random walk for θ, where $\mu(\theta, u)$ and $\Sigma(\theta, u)$ denote a mean and covariance function possibly depending on the auxiliary variables in the previous step, respectively. Note that by introducing u into q_θ, we can make use of gradient and Hessian information as proposed by Dahlin et al. (2015b) to improve the performance of the algorithm. The CN proposal for u is parametrised by the step length $\sigma_u \in (0, 1)$,

Algorithm 1 Pseudo-marginal Metropolis-Hastings (PMMH)

Inputs: $K > 0$ (no.
textsc mcmc steps), θ_0 (initial parameters), $\Phi_\theta(u)$ (potential) and $q_{\theta,u}$ (proposal).
Output: $\{\{\theta_1, u_1\} \ldots, \{\theta_K, u_K\}\}$ (approximate samples from $\bar{\pi}(\theta, u)$).

1: Generate $u_0 \sim p(u_0)$ and compute $\Phi_{\theta_0}(u_0)$.
2: **for** $k = 1$ to K **do**
3: Sample $\{\theta', u'\}$ using the proposal in (3).
4: Compute $\exp(-\Phi_{\theta'}(u'))$ and $\alpha(\{\theta', u'\}, \{\theta_{k-1}, u_{k-1}\})$ given by (4).
5: Sample ω_k uniformly over $[0, 1]$.
6: **if** $\omega_k \leq \alpha(\{\theta', u'\}, \{\theta_{k-1}, u_{k-1}\})$ **then**
7: Accept $\{\theta', u'\}$, i.e., $\{\theta_k, u_k\} \leftarrow \{\theta', u'\}$.
8: **else**
9: Reject $\{\theta', u'\}$, i.e., $\{\theta_k, u_k\} \leftarrow \{\theta_{k-1}, u_{k-1}\}$.
10: **end if**
11: **end for**

which is determined by the user. We return to discussing how to tune σ_u in Section 3. Note that (3) is in contrast with the standard independent PMMH proposal, which we recover for the choice $\sigma_u = 1$.

The corresponding PMMH algorithm for sampling from (2) follows directly from the choice of proposal in (3). During iteration k of the algorithm, we first sample from (3) to obtain the *candidate parameter* $\{\theta', u'\}$ given $\{\theta_{k-1}, u_{k-1}\}$. We then compute the acceptance probability by

$$\alpha\big(\{\theta', u'\}, \{\theta_{k-1}, u_{k-1}\}\big) = 1 \wedge \exp\Big(\Phi_{\theta_{k-1}}(u_{k-1}) - \Phi_{\theta'}(u')\Big) \frac{q_\theta\big(\theta_{k-1} \,|\, \{\theta', u'\}\big)}{q_\theta\big(\theta' \,|\, \{\theta_{k-1}, u_{k-1}\}\big)}, \quad (4)$$

where we make use of the notation $a \wedge b \triangleq \min\{a, b\}$. This follows directly from the properties of the CN proposal (Cotter et al., 2013). In the last step, we accept the candidate parameter, i.e., set $\{\theta_k, u_k\} \leftarrow \{\theta', u'\}$, with the probability given by (4). Otherwise, we reject the candidate parameter and set $\{\theta_k, u_k\} \leftarrow \{\theta_{k-1}, u_{k-1}\}$.

The resulting PMMH is presented in Algorithm 1. In Line 4, we need to evaluate the potential function in $\{\theta', u'\}$. In this paper, we make use of importance sampling and particle filtering in Section 4 to construct $\Phi_\theta(u)$. Note that Algorithm 1 only differs from a standard PMMH algorithm in Lines 3 and 4, where we add a CN proposal for u and evaluate the potential function $\Phi_\theta(u)$. Hence, implementing the new algorithm only requires minor changes to the existing code.

In many models, we are required to make use of non-Gaussian random variables to generate samples from the proposal distributions in e.g., importance sampling and particle filtering algorithms. Furthermore, we require uniform random variables for the resampling step in the particle filter, see e.g., Doucet and Johansen (2011). These variables can be obtained by using a inverse cumulative distribution function (CDF) transformation also known as a quantile transformation. Firstly, we compute a uniform random variable by $u^\star = \Phi(u)$,

where $\Phi(u)$ denotes the CDF of the standard Gaussian distribution. Secondly, we compute $\tilde{u}$ by a quantile transform of $u^\star$ using the inverse CDF for the distribution that we would like to simulate a random variable from. There exists other approaches e.g., accept-reject sampling that also can be useful for simulating random variables when the CDF cannot be inverted in closed-form, see e.g., Ross (2012) or Robert and Casella (2004).

Theoretical analysis

The aim of the theoretical analysis in this section is to give some guidance of how to tune σ_u to obtain a reasonable performance in the PMMH algorithm. More specifically, we are interested in determining the value of σ_u that maximises the mixing and to determine how this translates into a suitable acceptance rate in the algorithm. We begin by setting up the model for the analysis in Section 3.1, which depends on the multivariate random variable u. We then reformulate the model to replace u with a univariate random variable z. This enables use to make use of an analysis of a discretised model (Gelman et al., 1996; Yang and Rodríguez, 2013) in Section 3.2 to tune the CN proposal to optimise the mixing in the Markov chain. In Section 3.3, we report the results of this analysis. We return to the analysis in Section 4, the optimal tuning of σ_u is investigated in some practical scenarios.

Setting up the model

To accomplish the aforementioned aim we will study the algorithm in a simplified setting. Firstly, since we are primarily interested in the effect of the correlation in the u-variables, we will start out by making the simplifying assumption that we fix θ in the extended target $\bar{\pi}(\theta, u)$. Hence, we would like to analyse sampling from

$$\bar{\pi}(u) = \frac{\exp\big(-\Phi(u)\big)}{c}\mathcal{N}\big(u; 0, \mathbf{I}_{N_u}\big), \tag{5}$$

using the CN proposal $q_u(u'|u)$ in (3). We believe that this is a reasonable proxy for the complete model (2) since in many cases only local moves are made in the θ variable. Following Pitt et al. (2012) and Doucet et al. (2015) we also assume that $\Phi(u)$ (which depend on N_u) follows a central limit theorem (CLT) and that it therefore can be accurately approximated as being Gaussian distributed:

$$-\Phi(u) \sim \mathcal{N}\left(-\frac{\sigma_\Phi^2}{2}, \sigma_\Phi^2\right), \qquad \text{when } u \sim \mathcal{N}\big(u; 0, \mathbf{I}_{N_u}\big). \tag{6}$$

for some $\sigma_\Phi > 0$. This assumption follows from the properties of the log-likelihood estimator based on (sequential) importance sampling; see Pitt et al., 2012; Doucet et al., 2015 for further details. This case is of interested to us as we make use of this type of estimator in Section 4 for computing an estimate of the parameter posterior distribution.

We can readily check that assumption (6) implies that

$$-\Phi(u) \sim \mathcal{N}\left(\frac{\sigma_\Phi^2}{2}, \sigma_\Phi^2\right), \qquad \text{when } u \sim \bar{\pi}(u), \tag{7}$$

where $\bar{\pi}$ is the N_u-dimensional distribution defined in (5). Consequently, if we define the random variable

$$z \triangleq \frac{\sigma_\Phi}{2} - \frac{1}{\sigma_\Phi}\Phi(u)$$

it follows that the law of z under $\bar{\pi}$ is given by

$$\widetilde{\pi}(z) = \mathcal{N}(z; \sigma_\Phi, 1).$$

Note that z is a one-dimensional variable, which means that we have reduced the high-dimensional target (5) to one of its one-dimensional marginals. We will study the properties of the CN proposal in this one-dimensional marginal space.

Indeed, assume that $u \sim \bar{\pi}(u)$ is distributed according to the target (5) (i.e., it can be viewed as the state of the Markov chain at stationarity) and that $u'|u \sim q_u(u'|u)$ is simulated from the CN proposal in (3). If can define z and z' in the same manner as above, i.e. as transformations of u and u', respectively. From a bivariate CLT, we can then obtain a bivariate Gaussian approximation for the vector $(z, z')^\mathsf{T}$, suggesting that the CN proposal for u corresponds to the proposal

$$\widetilde{q}(z'|z) = \mathcal{N}\left(z'; \sqrt{1-\sigma_z^2}, \sigma_z^2\right), \tag{8}$$

for $z \in \mathbb{R}$, for some $\sigma_z > 0$. Note that σ_z in general differs from σ_u and that it depends on the non-linear transformation $\Phi(u)$. However, it is clear that $\sigma_u = 0 \Rightarrow \sigma_z = 0$ and $\sigma_u = 1 \Rightarrow \sigma_z = 1$ and, intuitively, σ_z is a monotone function of σ_u. We investigate the dependence between these two variables numerically in Section 4.1.

Note that the resulting acceptance probability, for the reduced one-dimensional MH algorithm with target $\widetilde{\pi}(z)$ and proposal $\widetilde{q}(z'|z)$, is given by

$$\widetilde{\alpha}(z, z') = 1 \wedge \exp(\sigma_\Phi z' - \sigma_\Phi z), \tag{9}$$

which is due to the symmetry of the proposal.

Analysis by discretisation of the state space

We follow Gelman et al. (1996) and Yang and Rodríguez (2013) and analyse the mixing of the Markov chain using a state space discretisation approach. We know that the expectation of any integrable *test function* $\varphi : \mathcal{Z} \to \mathbb{R}$ under the target distribution

$$\widetilde{\pi}[\varphi] = \mathbb{E}_{\widetilde{\pi}}[\varphi(z)] = \int_{\mathcal{Z}} \varphi(z)\widetilde{\pi}(z)\,\mathrm{d}z,$$

can be approximated by the ergodic theorem (Meyn and Tweedie, 2009; Tierney, 1994) as

$$\widehat{\widetilde{\pi}}[\varphi] = \frac{1}{K}\sum_{k=1}^{K} \varphi(z_k),$$

using the samples $\{z_1, z_2, \ldots, z_K\} \triangleq z_{1:K}$ obtained from the PMMH algorithm. Under the assumption of geometric ergodicity, we have that the error of the estimate obeys a CLT

(Meyn and Tweedie, 2009; Tierney, 1994) given by

$$\sqrt{K}\left(\widetilde{\pi}[\varphi] - \widehat{\widetilde{\pi}}[\varphi]\right) \xrightarrow{d} \mathcal{N}(0, v),$$

where v denotes the asymptotic variance,

$$v = \mathbb{V}_{\widetilde{\pi}}[\varphi] \cdot \underbrace{\left[1 + 2\sum_{\tau=1}^{\infty} \widehat{\rho}_\tau(z)\right]}_{\triangleq \mathsf{IACT}(z)}, \tag{10}$$

where $\mathbb{V}_{\widetilde{\pi}}[\varphi]$ and $\mathsf{IACT}(z)$ denote the variance of $\varphi(z)$ over $\widetilde{\pi}(z)$ and the integrated autocorrelation time (IACT), respectively. Here, $\rho_\tau(z) = \mathsf{corr}(\varphi(z_k), \varphi(z_{k+\tau}))$ denotes the lag-τ autocorrelation of φ.

To optimise the mixing, we would like to determine σ_z such that $1/v$ is as large as possible. To estimate the asymptotic variance, we partition the interval $(z_{\min}, z_{\max})$ into L bins of equal length $\Delta = (z_{\max} - z_{\min})/L$. Hence, the center point of bin l is given by $z_l = z_{\min} + (l - 0.5)\Delta$. We can then define the transition matrix $P = \{p_{lm}\}$ following Yang and Rodríguez (2013) by

$$p_{lm} = \begin{cases} \widetilde{q}(z_m \,|\, z_l)\widetilde{\alpha}(z_l, z_m)\Delta, & \text{for } l, m \in \{1, \ldots, K\}, l \neq m \\ 1 - \sum_{k \neq l} p_{lk}, & \text{for } l = m \end{cases}, \tag{11}$$

where the proposal (8) and acceptance probability (9) enters into p_{lm}. Hence, we can estimate the *acceptance rate* (the probability of a jump) by

$$P_{\text{jump}} = \sum_{l=1}^{L} \pi_l (1 - p_{ll}), \tag{12}$$

where the *stationary probability* is calculated as

$$\pi_l = \mathcal{N}(z_l; \sigma_\Phi, 1).$$

Furthermore from the asymptotic results in Peskun (1973) and Kemeny and Snell (1976), we can estimate v by

$$\widehat{v}(f, \pi, P) = \varphi^\top (2BZ - B - BA)\varphi, \tag{13}$$

where $\varphi = [\varphi(z_1), \varphi(z_2), \ldots, \varphi(z_L)]$ and $B = \mathsf{diag}(\pi_1, \pi_2, \ldots, \pi_L)$. Here, we introduce the *fundamental matrix* by $Z = [I - (P - A)]^{-1}$ and the *limiting matrix* by $A = \{a_{lm}\}$ with $a_{lm} = \pi_m$.

Tuning the CN proposal for the univariate case

We make use of a modified version of the C-code provided by Yang and Rodríguez (2013) to implement the discretisation approach. We set $z_{\min} = -4$, $z_{\max} = \sigma_\Phi + 4$ and $L = 1,000$. Here, we use $\varphi(z) = z$ to optimise the mixing of the posterior mean. We vary σ_Φ in the target and σ_z in the proposal in $\{0, 0.25, \ldots, 3.5\}$ and $\{0.05, 0.075, \ldots, 1\}$, respectively. For each σ_Φ, we find the σ_z that maximises P_{jump} computed by (12) and $1/\widehat{v}(f, \pi, P)$ calculated by (13). In Figure 1, we present the resulting σ_z is as a function of σ_Φ. We

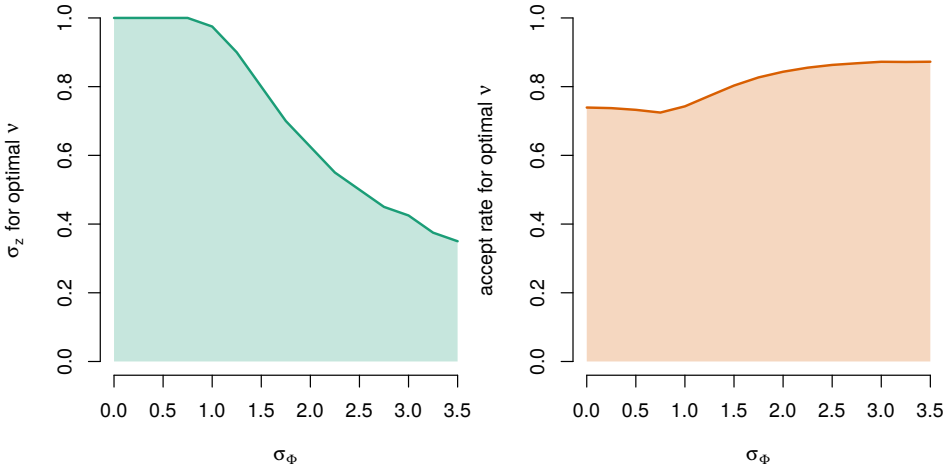


Figure 1. Optimal σ_u (left) and acceptance rate (right) for minimising the $1/\widehat{v}(f, \pi, P)$.

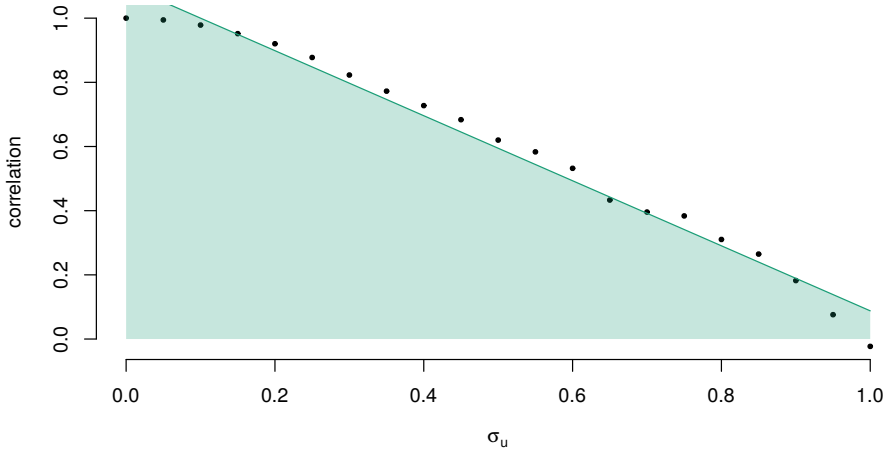


Figure 2. The estimated correlation (dots) in $\widehat{p}_\theta(y; u)$ for the Gaussian IID model (15) as a function of σ_u and a linear regression (green). The linear approximation provides a reasonable approximation for much of the central part of the interval.

note that for $\sigma_\Theta \in [1.0, 1.8]$ as recommended by e.g., Doucet et al. (2015), we should have $\sigma_z \in [0.95, 0.7]$ to obtain the optimal mixing and this corresponds to an acceptance rate of around 75% to 85% in the Markov chain for u.

Note that these results hold for the case when u can be described by the univariate Gaussian random variable z. We return to numerically investigate how to optimal value of σ_z in the CN proposal for the univariate random variable z relates to the corresponding value σ_u used in the CN proposal for the multivariate random variable u in Section 4.

Numerical illustrations

In this section, we investigate the numerical properties of the proposed alterations to the PMMH algorithm. We are especially interested in how the mixing is affected by making use of the CN proposal instead of an independent proposal for u. For this end with compare the mixing in two different models: (i) a Gaussian IID model with synthetic data and (ii) a non-linear state space model with real-world data. The implementation details are presented in Appendix A, where we also describe how to estimate the value of the target distribution (log-likelihood) for each model.

We quantify the mixing by estimating the IACT using the empirical autocorrelation function. The estimate is computed by

$$\widehat{\mathsf{IF}}(\theta_{K_b:K}) = 1 + 2\sum_{\tau=1}^{100} \widehat{\rho}_\tau(\theta_{K_b:K}), \tag{14}$$

where $\widehat{\rho}_\tau(\theta_{K_b:K})$ denotes the empirical lag-τ autocorrelation of $\theta_{K_b:K}$, and K_b denotes the *burn-in* time.

Gaussian IID model

Consider the independent and identically (IID) distributed Gaussian model given by

$$x_0 \sim \delta_0, \qquad x_t \sim \mathcal{N}\left(x_t; \mu, \sigma_v^2\right), \qquad y_t \,|\, x_t \sim \mathcal{N}\left(y_t; x_t, \sigma_e^2\right), \tag{15}$$

with parameters $\theta = \{\mu, \sigma_v, \sigma_e\}$ and where $\delta_{x'}$ denotes the Dirac measure placed at $x = x'$. We generate a realisation with $T = 10$ observations using the parameters $\theta^\star = \{0.5, 0.3, 0.1\}$.

We begin by investigating how the correlation of the log-likelihood estimated using importance sampling (see Appendix A.1) depends on σ_u. In Figure 2, we present the correlation between two consecutive estimates of the log-likelihood (keeping μ fixed) when $\sigma_u \in \{0, 0.05, \ldots, 1.0\}$. We note that the correlation is almost linear when $\sigma_u > 0.2$ and can be described as $\mathsf{corr}(\widehat{p}_\theta(y; u), \widehat{p}_\theta(y; u')) = 1 - \sigma_u$. We also estimate the standard deviation in the log-likelihood and obtain 3.5. Hence, we conclude that this implies from Figure 1 that the optimal correlation in the log-likelihood estimator is approximately 0.5.

We proceed by fixing σ_v and σ_e to their true values and would like to infer the parameter posterior of μ given the data using Algorithm 1. Here, the potential function corresponds to the log-likelihood estimator for the importance sampler as discussed in Appendix A.1. The

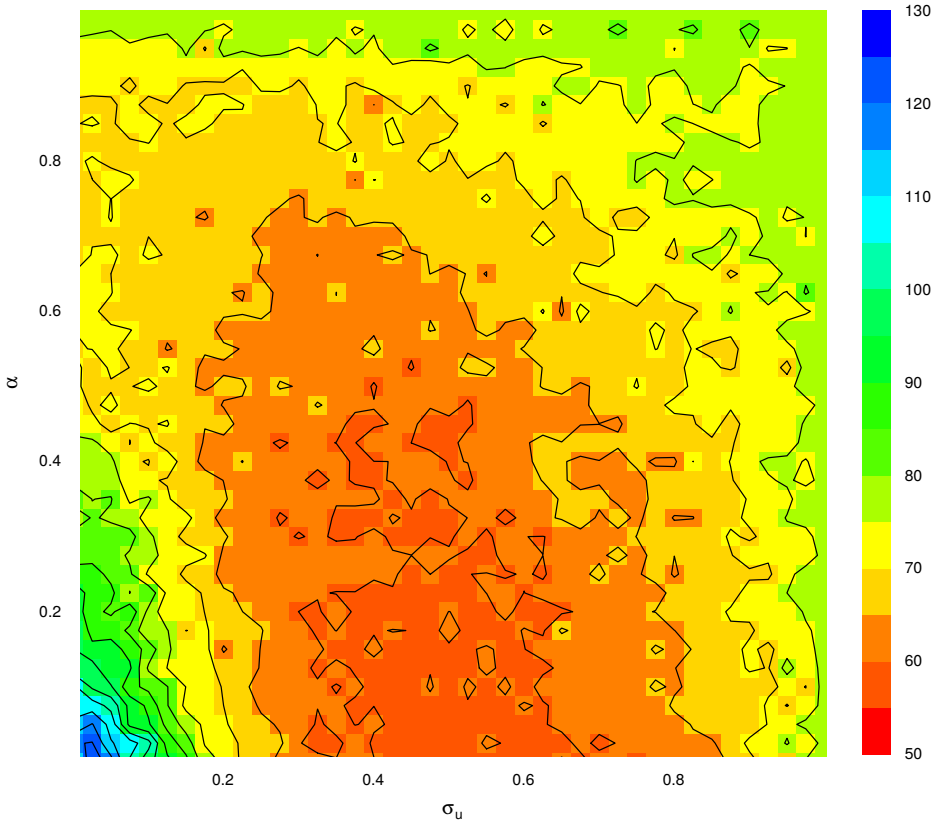


Figure 3. A heatmap of the IACT for the Gaussian IID model (15) when varying α and σ_u in (16). The results presented are the median from 32 independent Monte Carlo runs.

aim is to compute the IACT over a grid of σ_v and compare with the optimal theoretical results from Section 3. We would also like to compare the proposed changes to PMMH with the suggestions discussed by Lee and Holmes (2010). For this end, we consider a generalisation of the CN proposal (3) in which we introduce so-called *global moves*. The mixture of local and global moves is a popular approach in e.g., computer graphics (Kelemen et al., 2002) to promote exploration of the entire target space. In our setup, we can introduce such a mixture into $q_u(u' \mid u_{k-1})$ and obtain the mixture proposal given by

$$\tilde{q}_{\theta,u}(\{\theta', u\} \mid \{\theta, u\}) = q_\theta(\theta' \mid \{\theta, u\}) \Big[\alpha\, \mathcal{N}(u'; 0, \mathbf{I}_{N_u}) + (1-\alpha)\, q_u(u' \mid u_{k-1})\Big] \tag{16}$$

where we make use of the original proposals for θ and u from (3). Here, $\alpha \in [0, 1]$ denotes the probability of a global move for u, where we recover the proposal in (3) by $\alpha = 0$ and the proposal discussed by Lee and Holmes (2010) by $\sigma_u = 0$ and $\alpha \neq 0$. We recover the standard PMMH proposal for u when $\alpha = 1$ and/or $\sigma_u = 1$.

In Figure 3, we present a heatmap of the median IACT using (16) when varying $\alpha, \sigma_u \in \{0, 0.025, \ldots, 1.0\}$. The smallest IACT values are obtain by using $\sigma_u \in [0.4, 0.6]$ with a small (or zero) value of α. This range of σ_u corresponds well with the results in Figure 1 with $\sigma_\Phi = 3.5$, which results in the optimal $\sigma_z = 0.4$. Furthermore, we note that using $\sigma_u = 0$ results in worse performance than if $\sigma_u > 0$ independently of the value of α. Hence, we conclude that there is some benefit of using our proposed modification compared with the approach discussed by Lee and Holmes (2010) in this specific example. That is, using local moves seem to be more beneficial than global moves but there could be some merit to consider a mixture proposal in any case. However, we set $\alpha = 0$ to simplify the tuning and make the conclusions more precise regarding the choice of σ_u.

Stochastic volatility model with leverage

Consider the problem of modelling the volatility in the daily closing prices of the NASDAQ OMX30 index, i.e., a weighted average of the 30 most traded stocks at the Stockholm stock exchange. We extract the $T = 747$ log-returns using Quandl[1] for the period January 2, 2011 and January 2, 2014. To model the underlying volatility, we make use of a stochastic volatility model with leverage given by

$$x_0 \sim \mathcal{N}\left(x_0; \mu, \frac{\sigma_v^2}{(1-\phi^2)^2}\right), \tag{17a}$$

$$\begin{bmatrix} x_{t+1} \\ y_t \end{bmatrix} \Bigg| x_t \sim \mathcal{N}\left(\begin{bmatrix} x_{t+1} \\ y_t \end{bmatrix}; \begin{bmatrix} \mu + \phi(x_t - \mu) \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_v^2 & \rho \\ \rho & \exp(x_t) \end{bmatrix}\right), \tag{17b}$$

with parameters $\theta = \{\mu, \phi, \sigma_v, \rho\}$. In this model, we would like to infer the parameter posterior of θ given the data using Algorithm 1. Here, the potential function corresponds to the log-likelihood estimator using the particle filter as discussed in Appendix A.2. The aim is to again compute the IACT and compare with the theoretical results from Section 3.

In Figure 4, we present the median acceptance rate and IACT for the four parameters in the model when varying $\sigma_u \in \{0.05, 0.10, \ldots, 1.00\}$. The maximum IACT (over the four variables) is minimised when $\sigma_u = 0.55$. However, the variation in the IACT is significant

[1]The data is available for download from: `https://www.quandl.com/data/NASDAQOMX/OMXS30`.

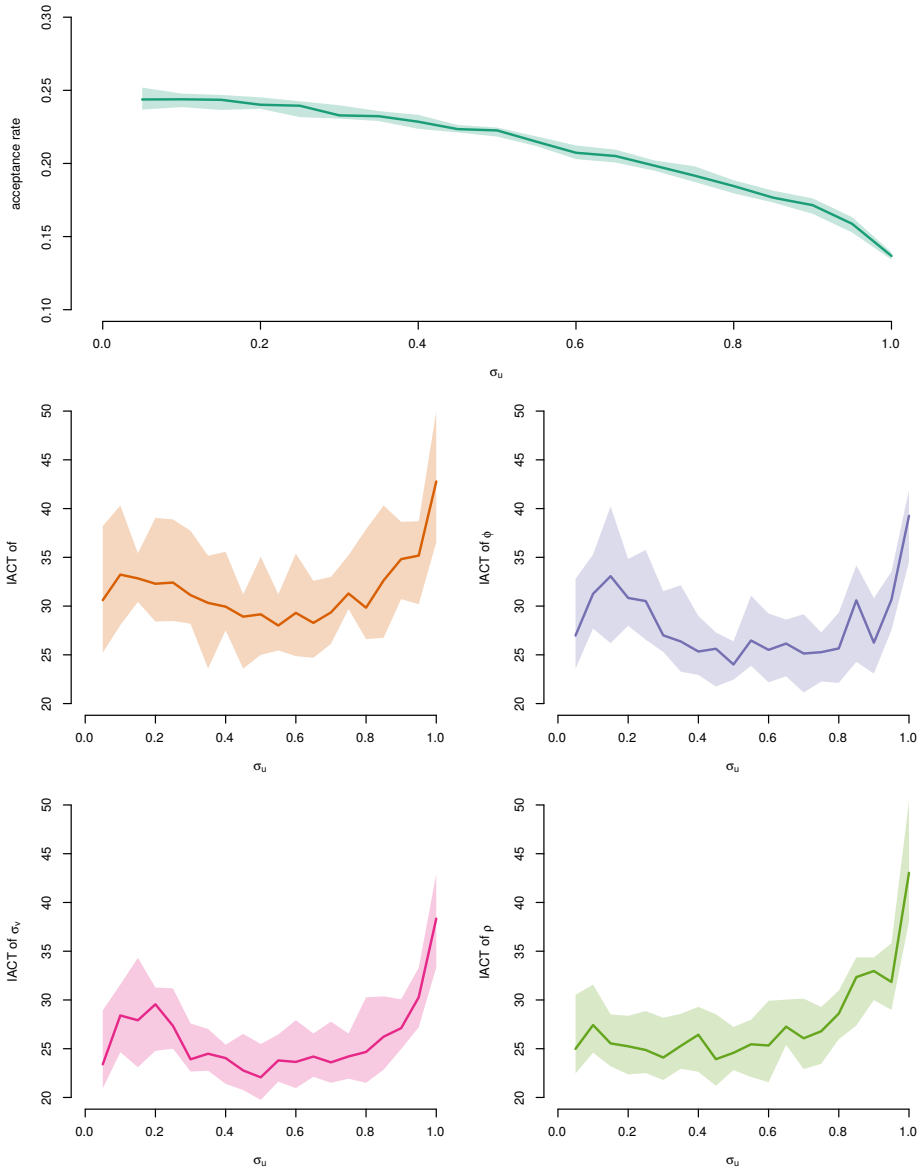


Figure 4. The acceptance probability (upper) and the resulting IACT (middle and lower) for μ (orange), ϕ (purple), σ_v (magenta) and ρ (light green) when varying σ_u. The results presented are the median (line) and first and third quantiles (shaded area) from 32 independent Monte Carlo runs.

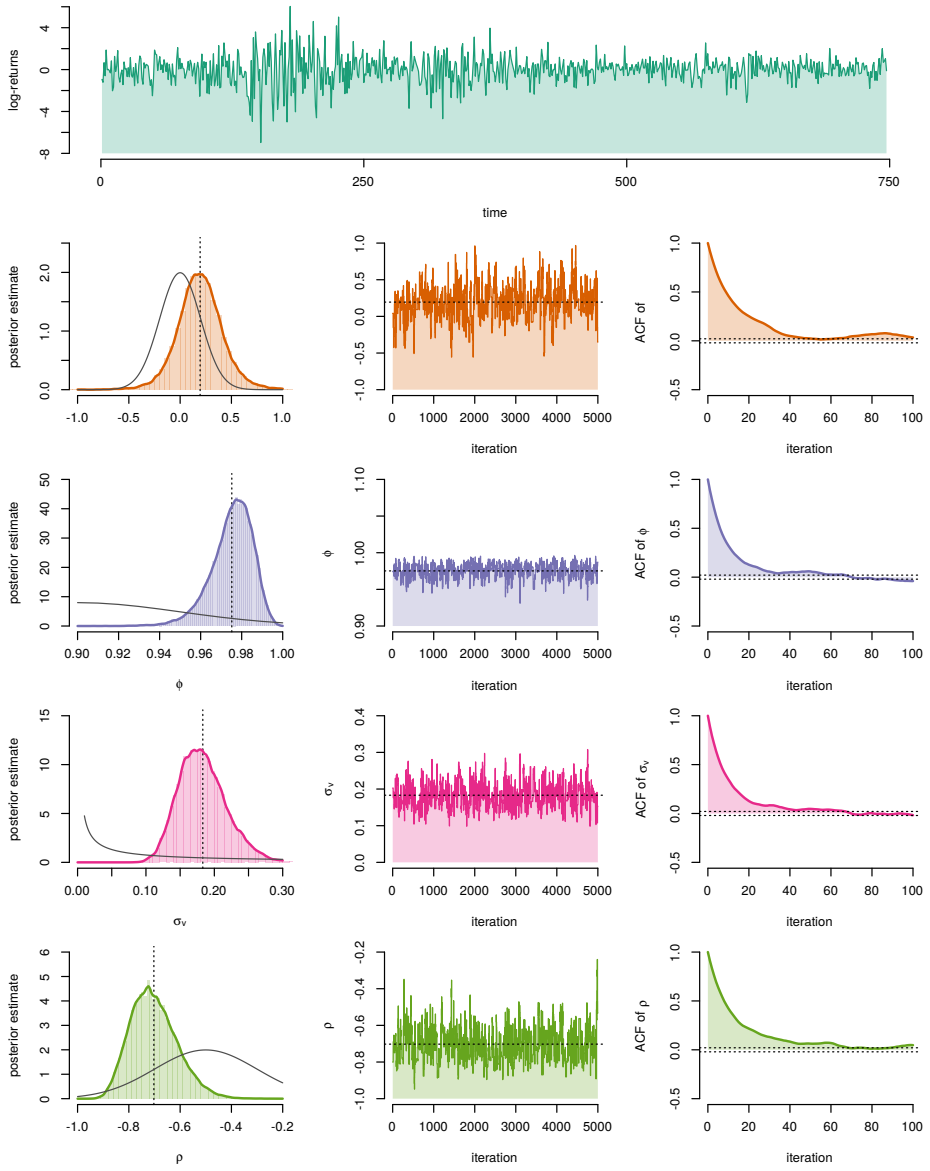


Figure 5. Upper: the closing log-returns for the NASDAQ OMXS30 index between January 2, 2011 and January 2, 2014. Lower: the parameter trace (left), autocorrelation function (center) and resulting posterior estimate (right) for the SV model (17) for μ (orange), ϕ (purple), σ_v (magenta) and ρ (light green). We make use of $\sigma_\mu = 0.55$ in the CN proposal (3) and the histograms are computed using the output from 32 independent Monte Carlo runs.

and a range between 0.4 and 0.8 seems as suitable choices for σ_u. The standard deviation of the log-likelihood estimator (around the estimated posterior mean) is 1.2, which by Section 3 would imply an optimal σ_z of around 0.9. This is clearly not an appropriate choice for this model. This can be due to that the Gaussian assumption for $\Phi_\theta(u)$ is not fulfilled or that the standard deviation of the log-likelihood estimate varies with θ.

From Figure 4, we conclude that using $\sigma_u < 1$ results in a decrease of the IACT comparing with using an independent proposal for u. The improvement in the mixing is about 1.5 times. We present a specific case in Figure (5), where we fix $\sigma_u = 0.55$. We conclude that the chains are mixing well and the posterior estimates are reasonable with the estimated posterior mean $\theta = \{0.19, 0.98, 0.18, -0.70\}$ with standard deviations $\{0.22, 0.01, 0.04, 0.09\}$.

Conclusions and future work

The numerical illustrations in Section 4 indicate that we can obtain improvements in the mixing of the PMMH algorithm by introducing correlation in u. In this paper, we consider using a CN proposal for introducing the correlation and it seems that selecting $\sigma_u = 0.5$ offers good performance in the settings that we have considered. Hence, implementing the proposed alterations to the PMMH algorithm only requires changing a small number of lines of code and hopefully does not introduce any additional tuning parameters for the user. We are currently working on an adaptive method to further alleviate the work of tuning σ_u for the user. The main benefit of our proposed modifications of the PMMH algorithm is that we can decrease the number of particles N_u and obtain better mixing at the same time. Hence, this results in a significant decrease in the computational cost for Bayesian inference based on the pseudo-marginal framework.

Some important additional future work is to extended the theoretical analysis in Section 3 to a more realistic setting, where the influence of θ also is taking into the account. Also it would be interesting to make use of Hilbert curve resampling proposed by Gerber and Chopin (2015) or some tree methods discussed by Lee (2008) in the particle filter. This would allow for inference in state space models with a multivariate state process.

It is also possible to make use of u to construct estimates of the gradient and Hessian of the log-posterior. This information can be included into the proposal for θ as discussed by Dahlin et al. (2015b,c) to further improve the mixing in the Markov chain. The gradient can also be used to reduce the variance in a post-processing step, see Mira et al. (2013).

The proposed alterations of the PMMH algorithm can also be useful in models where the likelihood is intractable as discussed by e.g., Jasra (2015) and Dahlin et al. (2015c). In this class of models, approximate Bayesian computations (ABC; Marin et al., 2012 can be used to approximate the log-likelihood using importance sampling and particle filtering. However, in practice these estimates suffer from a large variance with results in bad mixing for the Markov chain. This approach can possible result in a large decrease in the computational cost for using the PMMH algorithm for inference in models with intractable likelihoods.

Acknowledgements

The simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at Linköping University.

Appendix

Implementation details

In this appendix, we outline the implementation details of the numerical illustrations presented in Section 4. The implementations for estimating the log-likelihood are based on standard importance sampling and particle filtering. Extensive treatments of these methods are found in e.g., Doucet and Johansen (2011) and Robert and Casella (2004).

Gaussian IID model

We make use of an importance sampler with $N = 10$ to estimate the log-likelihood with the prior as the importance distribution. This results in that the log-likelihood can be estimated by the importance weights according to

$$\log \widehat{p}_\theta(y; u) = \sum_{t=1}^{T} \log \left[\sum_{i=1}^{N} w_t^{(i)} \right] - T \log N, \tag{18}$$

where the weights $w_t^{(i)}$ are generated using the procedure outlined in Algorithm 2.

For PMMH, we use $K = 10,000$ iterations (discarding the first $K_b = 1,000$ as burn-in) in Algorithm 1 and initialise in the true parameter $\theta_0 = 0.5$. The proposal for θ is a standard Gaussian random walk proposal (3) with $\mu(\theta, u) = \theta$ and $\Sigma(\theta, u) = 0.1^2$. Finally, we make use of the following prior

$$p(\mu) \sim \mathcal{TN}_{(0,1)}(\mu; 0, 1),$$

where $\mathcal{TN}_{(-1,1)}(\mu; 0, 1)$ denotes a Gaussian distribution truncated to the interval $(-1, 1)$ with mean 0 and scale 1.

Stochastic volatility model with leverage

For any SSM, we make use of a bootstrap particle filter (BPF; Doucet and Johansen, 2011) to estimate the log-likelihood. An SSM with latent states $x_{0:T} = \{x_t\}_{t=0}^{T}$ and observations $y_{1:T}$ is given by

$$x_0 \sim \mu_\theta(x_0), \qquad x_{t+1} \,|\, x_t \sim f_\theta(x_{t+1} \,|\, x_t), \qquad y_t \,|\, x_t \sim g_\theta(y_t \,|\, x_t), \tag{19}$$

where $\theta \in \Theta \subseteq \mathbb{R}^p$ denotes the static unknown parameters. Here, we assume that it is possible to simulate from the distributions $\mu_\theta(x_0)$ and $f_\theta(x_{t+1} \,|\, x_t)$ and evaluate $g_\theta(y_t \,|\, x_t)$ point-wise. A BPF to estimate $\widehat{p}_\theta(y; u)$ is presented in Algorithm 3.

Hence, the log-likelihood is computed using the same expression (18) as for importance sampling but the particles $x_t^{(i)}$ are instead generated by sequential importance sampling

with resampling. Note that we are required to sort the particles after the propagation step, see the discussion about smooth particle filter by Malik and Pitt (2011) for more details. We make use of the probability transform to generate the uniform random variables required for the systematic resampling step. Hence, u is a $(N_u + 1) \cdot (T + 1)$-variate Gaussian random variable, where $u_{2:N+1,t}$ is used directly in the propagation step and $u_{1,t}$ is used in the resampling step after a transformation into a uniform random variable. Here, we make use of $N_u = 50$ particles.

For PMMH, we use $K = 10,000$ iterations (discarding the first $K_b = 1,000$ as burn-in) in Algorithm 1 and initialise the parameters at $\theta_0 = \{0.23, 0.98, 0.18, -0.72\}$ obtained using the approach discussed by Dahlin et al. (2015d). The proposal for θ is a standard Gaussian random walk proposal (3) with $\mu(\theta, u) = \theta$. Here, we make use of the rules of thumb by Sherlock et al. (2015) to select the covariance function. We extract an estimate of the Hessian using the method by Dahlin et al. (2015d) and set

$$\Sigma(\theta, u) = \frac{2.562^2}{p} \cdot 10^{-4} \cdot \begin{bmatrix} 384 & 3 & -5 & -16 \\ 3 & 1 & -3 & -2 \\ -5 & -3 & 12 & 3 \\ -16 & -2 & 3 & 65 \end{bmatrix}.$$

Finally, we use the following prior distributions

$$\begin{aligned} p(\mu) &\sim \mathcal{N}(\mu; 0, 2^2), & p(\phi) &\sim \mathcal{TN}_{(-1,1)}(\phi; 0.9, 0.05^2), \\ p(\sigma_v) &\sim \mathcal{G}(\sigma_v; 2.0, 0.05), & p(\rho) &\sim \mathcal{N}(\rho; -0.5, 0.2^2), \end{aligned}$$

where $\mathcal{G}(a, b)$ denotes the Gamma distribution with mean a/b.

Algorithm 2 Likelihood estimation using importance sampling with fixed random numbers

Inputs: y (vector of T observations), $N_u \in \mathbb{N}$ (no. samples) and $u \in \mathcal{U}$ ($T \cdot N_u$ standard Gaussian random variables).
Outputs: $\widehat{p}_\theta(y; u)$ (est. of the likelihood).
Note: all operations are carried out over $i = 1, \ldots, N_u$.

1: **for** $t = 1$ to T **do**
2: Simulate from the proposal by
$$x_t^{(i)} = \mu + \sigma_v^2 u_t^{(i)},$$
using random variables $u_{1:N_u,t}$.
3: Calculate the weights by
$$w_t^{(i)} = \mathcal{N}(y_t; x_t^{(i)}, \sigma_e^2).$$
4: **end for**
5: Estimate $\widehat{p}_\theta(y; u)$ by (18).

Algorithm 3 Likelihood estimation using particle filtering with fixed random numbers

Inputs: y (vector of T observations), an SSM (19), $N_u \in \mathbb{N}$ (no. particles) and $u \in \mathcal{U}$ ($(T+1) \cdot (N_u + 1)$ standard Gaussian random variables).
Outputs: $\widehat{p}_\theta(y; u)$ (est. of the likelihood).
Note: all operations are carried out over $i, j = 1, \ldots, N_u$.

1: Sample $x_0^{(i)} \sim \mu_\theta(x_0)$ using $u_{2:(N+1),1}$.
2: Set $W_0^{(i)} = N_u^{-1}$ as initial weights.
3: **for** $t = 1$ to T **do**
4: Apply the inverse CDF approach to transform $u_{1,t+1}$ into a uniform random number $\bar{u}_{1,t+1}$.
5: Apply systematic resampling with $\bar{u}_{1,t+1}$ to sample the ancestor index $a_t^{(i)}$ from a multinomial distribution with
$$\mathbb{P}\big(a_t^{(i)} = j\big) = W_{t-1}^{(j)}.$$
6: Propagate the particles by sampling
$$x_t^{(i)} \sim f_\theta\Big(x_t^{(i)} \Big| x_{t-1}^{a_t^{(i)}}\Big),$$
using random variables $u_{2:(N_u+1),t+1}$.
7: Extend the trajectory by $x_{0:t}^{(i)} = \Big\{x_{0:t-1}^{a_t^{(i)}}, x_t^{(i)}\Big\}$.
8: Sort the particle trajectories according to the current state $x_t^{(i)}$.
9: Calculate the particle weights by $w_t^{(i)} = g_\theta\big(y_t \,|\, x_t^{(i)}\big)$ which by normalisation (over i) gives $W_t^{(i)}$.
10: **end for**
11: Estimate $\widehat{p}_\theta(y; u)$ by (18).

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, 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. Andrieu, A. Doucet, and A. Lee. Discussion on constructing summary statistics for approximate Bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):451–452, 2012.

A. Beskos, G. Roberts, A. Stuart, and J. Voss. MCMC methods for diffusion bridges. *Stochastics and Dynamics*, 8(03):319–350, 2008.

S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. MCMC methods for functions: modifying old algorithms to make them faster. *Statistical Science*, 28(3):424–446, 2013.

J. Dahlin, F. Lindsten, J. Kronander, and T. B. Schön. Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables. *Pre-print*, 2015a. arXiv:1512.05483v1.

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015b.

J. Dahlin, F. Lindsten, and T. B. Schön. Quasi-Newton particle Metropolis-Hastings. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 981–986, Beijing, China, October 2015c.

J. Dahlin, M. Villani, and T. B. Schön. Efficient approximate Bayesian inference for models with intractable likelihoods. *Pre-print*, 2015d. arXiv:1506.06975v1.

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.

G. Deligiannidis, A. Doucet, and M. K. Pitt. The correlated pseudo-marginal method. *Pre-print*, 2015. arXiv:1511.04992v2.

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, G. Deligiannidis, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2): 295–313, 2015.

P. Fearnhead and L. Meligkotsidou. Augmentation schemes for particle MCMC. *Statistics and Computing (accepted for publication)*, pages 1–14, 2015.

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.

A. Gelman, G. Roberts, and W. Gilks. Efficient Metropolis jumping rules. *Bayesian statistics*, 5:599–607, 1996.

M. Gerber and N. Chopin. Sequential quasi Monte Carlo. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(3):509–579, 2015.

T. Hachisuka, A. S. Kaplanyan, and C. Dachsbacher. Multiplexed Metropolis light transport. *ACM Transactions on Graphics (TOG)*, 33(4):100, 2014.

M. Hairer, A. M. Stuart, and S. J. Vollmer. Spectral gaps for a Metropolis-Hastings algorithm in infinite dimensions. *The Annals of Applied Probability*, 24(6):2455–2490, 2014.

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

A. Jasra. Approximate Bayesian computation for a class of time series models. *International Statistical Review*, 83(3):405–435, 2015.

C. Kelemen, L. Szirmay-Kalos, G. Antal, and F. Csonka. A simple and robust mutation strategy for the Metropolis light transport algorithm. *Computer Graphics Forum*, 21(3): 531–540, 2002.

J. G. Kemeny and J. L. Snell. *Finite Markov chains*. Springer Verlag, 1976.

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

A. Lee. Towards smooth particle filters for likelihood estimation with multivariate latent variables. Msc thesis, University of British Columbia, 2008.

A. Lee and C. Holmes. Discussion on particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):327–328, 2010.

S. Malik and M. K. Pitt. Particle filters for continuous likelihood evaluation and maximisation. *Journal of Econometrics*, 165(2):190–209, 2011.

J-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

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.

A. Mira, R. Solgi, and D. Imparato. Zero variance Markov chain Monte Carlo for Bayesian estimators. *Statistics and Computing*, 23(5):653–662, 2013.

A. Owen and Y. Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143, 2000.

P. H. Peskun. Optimum Monte-carlo sampling using Markov chains. *Biometrika*, 60(3): 607–612, 1973.

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.

C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Verlag, 2 edition, 2004.

S. M. Ross. *Simulation*. Academic Press, 5 edition, 2012.

C. Sherlock, A. H. Thiery, G. O. Roberts, and J. S. Rosenthal. On the efficency of pseudo-marginal random walk Metropolis algorithms. *The Annals of Statistics*, 43(1):238–275, 2015.

L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22 (4):1701–1728, 1994.

M.-N. Tran, M. Scharth, M. K. Pitt, and R. Kohn. Importance sampling squared for Bayesian inference in latent variable models. *Pre-print*, 2014. arXiv:1309.3339v3.

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.

Z. Yang and Carlos E Rodríguez. Searching for efficient Markov chain Monte Carlo proposal kernels. *Proceedings of the National Academy of Sciences*, 110(48):19307–19312, 2013.

Paper E

Approximate Bayesian inference for models with intractable likelihoods

Authors: J. Dahlin, M. Villani and T. B. Schön

Edited version of the paper:

J. Dahlin, M. Villani, and T. B. Schön. Efficient approximate Bayesian inference for models with intractable likelihoods. *Pre-print*, 2015d. arXiv:1506.06975v1.

Approximate Bayesian inference for models with intractable likelihoods

J. Dahlin*, M. Villani† and T. B. Schön‡

*Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden.
johan.dahlin@liu.se

†Dept. of Computer and Information
Science, Linköping University,
SE-581 83 Linköping, Sweden.
mattias.villani@liu.se

‡Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
thomas.schon@it.uu.se

Abstract

We consider the problem of approximate Bayesian parameter inference in non-linear state space models with intractable likelihoods. Sequential Monte Carlo with approximate Bayesian computations (SMC-ABC) is an approach to approximate the likelihood in this type of models. However, such approximations can be noisy and computationally costly which hinders efficient implementations using standard methods based on optimisation and statistical simulation. We propose a novel method based on the combination of Gaussian process optimisation (GPO) and SMC-ABC to create a Laplace approximation of the intractable posterior. The properties of the resulting GPO-ABC method are studied using stochastic volatility (SV) models with both synthetic and real-world data. We conclude that the algorithm enjoys: good accuracy comparable to particle Markov chain Monte Carlo with a significant reduction in computational cost and better robustness to noise in the estimates compared with a gradient-based optimisation algorithm. Finally, we make use of GPO-ABC to estimate the Value-at-Risk for a portfolio using a copula model with SV models for the margins.

Keywords

Bayesian inference, approximate Bayesian computations, Gaussian process optimisation, sequential Monte Carlo, α-stable distributions.

Data and source code in Python

`https://github.com/compops/gpo-abc2015`

Financial support from

The projects *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) and CADICS, a Linnaeus Center, both funded by the Swedish Research Council.

Introduction

Dynamical modelling of time series data is an essential part of modern econometrics. A popular model is the state space model (SSM) used extensively in both macroeconomics and financial economics to capture e.g., the volatility of returns of stocks or other financial commodities. See McNeil et al. (2010), Tsay (2005) and Durbin and Koopman (2012) for a multitude of other applications within econometrics. Modelling of volatility is an important problem in financial risk management, where marginal models often are combined using a copula model to compute the Value-at-Risk (VAR) for a portfolio of assets.

The main problem is that Bayesian inference in non-linear SSMs is computationally expensive, which limits practical application of this type of modelling. Therefore, we aim to develop an efficient Bayesian approach for parameter inference in non-linear SSMs,

$$x_0 \sim \mu_\theta(x_0), \quad x_t \,|\, x_{t-1} \sim f_\theta(x_t \,|\, x_{t-1}), \quad y_t \,|\, x_t \sim g_\theta(y_t \,|\, x_t), \tag{1}$$

where $\theta \in \Theta \subseteq \mathbb{R}^p$ denotes unknown static parameters. Here, $x_t \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ and $y_t \in \mathcal{Y} \subseteq \mathbb{R}^{d_y}$ denotes the latent state and the observations at time $t \in \mathbb{N}$, respectively. For an SSM, we say that the likelihood $p(y_1, y_2, \ldots, y_T \,|\, \theta)$ is *intractable*, when we cannot evaluate $g_\theta(y_t \,|\, x_t)$ point-wise. This can be the result of $g_\theta(y_t \,|\, x_t)$ lacking an analytical closed-form expression, is defined recursively or is computationally prohibitive to evaluate. The main object of interest in Bayesian inference is the *parameter posterior distribution* given by

$$p(\theta \,|\, y_{1:T}) = \frac{p(\theta) p_\theta(y_{1:T})}{\int_\Theta p(\theta') p_{\theta'}(y_{1:T}) \,\mathrm{d}\theta'}, \tag{2}$$

where $p(\theta)$ denotes the parameter prior distribution and we denote the likelihood (with some slight abuse of notation) as $p_\theta(y_{1:T}) \triangleq p(y_1, y_2, \ldots, y_T \,|\, \theta)$. It follows that the posterior is intractable and cannot be evaluated as the likelihood is intractable.

A possible approach for inference in problems involving intractable likelihoods is provided by approximate Bayesian computations (ABC; Marin et al., 2012). This family of methods makes use of a reformulation of the inference problem, where the sought after parameter posterior (2) is perturbed by some noise. This results in that we can apply standard inference algorithms to estimate the parameters of the perturbed model. The error between the parameters of the perturbed model and the true model is in general difficult to quantify but this approach has been successfully applied for many challenging problems.

In our setting, we can make use of an ABC version of sequential Monte Carlo (SMC-ABC; Jasra et al., 2012) to estimate the likelihood of the perturbed SSM. However, these estimates suffer from problems with a large variance and are computationally expensive. We would therefore like to design an inference method that only requires a few estimates of the posterior and can handle the noise in them. With these properties in mind, we propose to make use of Gaussian process optimisation (GPO; Lizotte, 2008; Snoek et al., 2012) for extracting a Laplace approximation of the posterior. This is an iterative method that operates by constructing a *surrogate function* that mimics the parameter posterior. The resulting surrogate is smooth and computationally cheap to evaluate. Therefore, standard optimisation methods can be applied to the surrogate for constructing the Laplace approximation.

The main contribution of this paper is to introduce, develop and numerically study the resulting GPO-ABC algorithm, which is a combination of SMC-ABC and GPO. We note in the passing that GPO-ABC can be a competitive alternative for inference in any model where the likelihood is computationally costly to evaluate/estimate. Here, we exemplify this by combining GPO with ABC for inference in intractable SSMs. However, there are other interesting areas where GPO can be a competitive alternative. An example of this is our earlier work in Dahlin and Lindsten (2014), where we combine SMC and GPO to solve the maximum likelihood inference problem in SSMs with tractable likelihoods. Another example from biology is discussed by Gutmann and Corander (2015), where GPO is used for parameter inference in predator-prey models from computational biology. A similar application using ABC is considered by Meeds and Welling (2014).

We illustrate the usefulness of GPO-ABC by conducting parameter inference in two different stochastic volatility models using both synthetic and real-world data. In this first model, the log-returns are modelled as Gaussian and the likelihood is tractable. Therefore, we can apply the combination of standard SMC and GPO referred to as GPO-SMC to approximate the parameter posterior. This model is therefore useful for investigating the impact of the perturbation in ABC on the accuracy of the posterior approximation. In the second model, we assume that the log-returns are distributed according to an α-stable distribution (Nolan, 2003; Lombardi and Calzolari, 2009) and this results in that the likelihood is intractable. Therefore, we cannot make use of standard SMC and are required to make use of GPO-ABC for the inference.

We also consider an application from financial risk management, where we would like to estimate the VAR for a portfolio of assets. Here, we demonstrate that GPO-SMC and GPO-ABC can be used for inference of the parameters in the margins of a copula model. The use of GPO results in a considerable speed-up compared with alternative methods. This especially for the model with α-stable log-returns and this could encourage its use in this and related applications.

In our illustrations, we compare the GPO-ABC algorithm with some popular alternatives that also only require samples of the log-posterior. We present results that indicate some interesting advantages with the proposed method. In particular, we obtain: (i) good posterior approximations using only a small number of posterior estimates, (ii) a smaller computational cost compared with alternatives, (iii) good robustness to the ABC approximation and noise in the estimates. Altogether, this indicates that the proposed method can be a competitive alternative for approximate Bayesian inference in some models.

Alternative methods for solving the Bayesian parameter inference problem in SSMs with an intractable likelihood are usually based on statistical sampling or optimisation. An example of the former is particle Metropolis-Hastings (PMH; Andrieu et al., 2010), where Jasra (2015) and Dahlin et al. (2015b) discuss similar applications. Examples of the latter include gradient ascent algorithms discussed by Yıldırım et al. (2014) and the simultaneous perturbation and stochastic approximation (SPSA; Spall, 1987) discussed by Ehrlich et al. (2015). Some drawbacks with these approaches are that they in general require many samples from the intractable posterior and are sensitive to noise. The latter problem can be mitigated by increasing the computational cost but this can hinder practical parameter inference.

Gradient ascent methods also require running an smc-based smoother, which typically is more computationally costly than only estimating the posterior point-wise using SMC-ABC.

The paper continues with a discussion of the background and motivation for using GPO together with an overview of the proposed GPO-ABC algorithm. We then continue in Section 3 with discussing the details of SMC-ABC, which we make use of to approximate the intractable log-posterior. In Section 4, we discuss how to construct the surrogate function that approximates the log-posterior inside GPO-ABC. We also show how to make use of the surrogate function to efficiently explore the parameter space by balancing exploration and exploitation. In Sections 6 and 7, we provide some numerical illustrations of GPO-ABC to illustrate its advantages, discuss some conclusions from the experiments and highlight some interesting areas for future developments of GPO-ABC.

An intuitive overview of GPO-ABC

The aim of the proposed method is to construct a *Laplace approximation* (Bishop, 2006) of the parameter posterior given in (2). This approximation is given by completing the square of a second-order Taylor approximation of the log-posterior and can be expressed as

$$\widehat{p}(\theta \,|\, y_{1:T}) = \mathcal{N}\Big(\theta; \widehat{\theta}_{\text{MAP}}, \big[\underbrace{-\nabla^2 \log p(\theta \,|\, y_{1:T})\big|_{\theta=\widehat{\theta}_{\text{MAP}}}}_{\triangleq \mathcal{J}(\widehat{\theta}_{\text{MAP}})}\big]^{-1}\Big), \tag{3}$$

where $\mathcal{N}(x; \mu, \Sigma)$ denotes a Gaussian density with mean vector $\mu \in \mathbb{R}$ and covariance matrix $\Sigma > 0$. Here, $\mathcal{J}(\widehat{\theta}_{\text{MAP}})$ denotes the estimate of the Hessian of the log-posterior at the MAP estimate given by

$$\widehat{\theta}_{\text{MAP}} = \operatorname*{argmax}_{\theta \in \Theta} \log p(\theta \,|\, y_{1:T}), \tag{4}$$

where $y_{1:T}$ denotes the recorded observations. This can be seen as a Gaussian approximation around the mode of the posterior. However, it can also be motivated by the Bernstein-von Mises theorem (Van der Vaart, 2000), which states that the posterior is asymptotically Gaussian concentrated around the maximum likelihood estimate with the inverse expected information matrix as its covariance. It then follows that the Laplace approximation tends to the true posterior in the limit of infinite number of observations. However, often only a finite number of samples are required to obtain reasonable approximations, see e.g., Panov and Spokoiny (2015).

We encounter two main problems when constructing the Laplace approximation of (2): (i) the optimisation problem in (4) cannot be efficiently solved using standard methods and (ii) $\mathcal{J}(\widehat{\theta}_{\text{MAP}})$ is typically difficult to estimate with good accuracy. The first problem is a result of that we can only obtain quite noisy estimates of the log-posterior with a large computational cost. Consequently, it is difficult to apply SPSA and quasi-Newton methods with numerical gradients, which typically requires many samples of the log-posterior. These methods can also suffer from slow convergence when the variance of the estimates is large, which further limits their computational feasibility.

Conversely, it is possible to estimate gradients of the log-posterior and make use of first-order optimisation methods such as gradient ascent algorithms. However, gradient estimates are even more costly to obtain and often suffers from even larger variance than estimates of the log-posterior. In analogue, we can use smoothing to estimate the Hessian of the log-posterior but this approach suffers from the same problems as for gradient estimation. Furthermore, Hessian estimates can often be negative definite, which is problematic as they should be interpreted as an inverse covariance matrix. This is the main reason for the second problem that we encounter when constructing the Laplace approximation.

The main contribution in this paper is that we instead propose to make use of GPO to circumvent these two difficulties by creating the Laplace approximation from the information obtained by a surrogate function (also known as a response surface). The aim is to create this surrogate function using samples from the log-posterior obtained by SMC-ABC. For this approach to work, we require that the surrogate is smooth, computationally cheap to evaluate and closely resembles the log-posterior around its mode. We can therefore make use of standard optimisation methods and finite differences to construct a Laplace approximation of the log-posterior by the use of the surrogate function.

GPO is a specific instance of Bayesian optimisation and operates by sequentially updating the surrogate function using samples from the log-posterior. In this paper, we make use of the predictive distribution of a Gaussian process (GP) as the surrogate function. Hence, we focus on briefly introducing the GPO algorithm and refer the interested reader to Boyle (2007), Lizotte (2008) and Brochu et al. (2010) for more detailed accounts. GPO is a useful method for globally optimising objective functions that are costly to evaluate and possibly quite noisy. This as we may encode certain prior knowledge regarding the log-posterior into the GP prior, which can reduce the number of samples required to explore the log-posterior. Note that the GPO algorithm can possibly be useful for parameter inference in other types of models than the intractable likelihood models discussed in this paper.

The GPO algorithm is an iterative algorithm, which alternates between the following steps:

(i) compute an approximation of the log-posterior $\xi_k = \log \widehat{p}(\theta_k \,|\, y_{1:T})$ using the SMC-ABC algorithm.

(ii) construct a surrogate function by computing the GP predictive posterior given $\{\boldsymbol{\theta}_k, \boldsymbol{\xi}_k\} = \{\theta_j, \xi_j\}_{j=1}^k$.

(iii) evaluate the *acquisition rule* to determine θ_{k+1}.

We have already briefly discussed Steps (i) and (ii). However, Step (iii) requires some additional explanation. The GP predictive posterior can be seen as an infinite dimensional Gaussian distribution, which is described by a mean function and a covariance function. Hence, we have information about areas where the predictive mean and covariance are large and this could be useful in determining the next parameter θ_{k+1} to sample the log-posterior in. This is known as the *exploration and exploration problem*, i.e., when we sample parameters in areas where the covariance and mean are large, respectively. The heuristic that balances these two phases is known as an *acquisition function* in the GPO literature and is the second reason for why GPO is a competitive method for global optimisation. We return to discussing the use of the GP and the acquisition function in more detail in Section 4.

Note that (3) only provide us with an approximation of $p(\theta \mid y_{1:T})$ in (2). Therefore, it could be advantageous to consider GPO-ABC as an initial step for parameter inference using the PMH algorithm. The first alternative is to use the Laplace approximation as an independent proposal as discussed by Andrieu et al. (2010) and Pitt et al. (2012). The second alternative is to use the information to pre-condition a Gaussian random walk as discussed by Girolami and Calderhead (2011) and Sherlock et al. (2015). Hence, GPO-ABC could possibly help with the design of a proposal distribution and initialisation in PMH. Both problems are very challenging in practice when the number of parameters in the model is large.

Estimating the log-posterior

In this section, we discuss how to estimate the log-posterior that is required for carrying out Step (i) of GPO-ABC. Remember that the main difficulty is that the log-likelihood is intractable. For a general SSM, we can write the *complete data log-likelihood* as

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

which we can marginalise to obtain the log-likelihood according to

$$\log p_\theta(y_{1:T}) = \int \log p_\theta(x_{0:T}, y_{1:T}) \, \mathrm{d}x_{0:T}.$$

However, this integral is intractable in all models except in the linear Gaussian case or when the state space is finite. Instead, we propose to use SMC methods known as particle filters to sequentially estimate the predictive log-likelihood $\log p_\theta(y_t \mid y_{1:t-1})$ and make use of

$$\log p_\theta(y_{1:T}) = \log p_\theta(y_1) + \sum_{t=2}^{T} \log p_\theta(y_t \mid y_{1:t-1}),$$

to estimate the log-likelihood. Note that the predictive likelihood can be computed using

$$p_\theta(y_t \mid y_{1:t-1}) = \int p_\theta(y_t, x_t \mid y_{1:t-1}) \, \mathrm{d}x_t = \int g_\theta(y_t \mid x_t) f_\theta(x_t \mid x_{t-1}) p_\theta(x_{t-1} \mid y_{1:t-1}) \, \mathrm{d}x_{t-1:t}. \tag{5}$$

A standard approach to estimate this quantity is to obtain an estimate of the *marginal filtering distribution* $p_\theta(x_{t-1} \mid y_{1:t-1})$ as a collection of weighted Dirac measures

$$p_\theta^N(\mathrm{d}x_{t-1} \mid y_{1:t-1}) = \sum_{i=1}^{N} w_{t-1}^{(i)} \delta_{x_{t-1}^{(i)}}(\mathrm{d}x_{t-1}), \tag{6}$$

where $x_{t-1}^{(i)}$ and $w_{t-1}^{(i)}$ denotes the particle i at time $t-1$ and its normalised weight, respectively. Here, δ_x denotes the Dirac measure placed at x. The particle system $\{w_t^{(i)}, x_t^{(i)}\}_{i=1}^N$ can be generated using the particle filter. However, the standard particle filter cannot be directly used as we assume that $g_\theta(y_t \mid x_t)$ cannot be evaluated point-wise.

Consequently, we cannot estimate the predictive likelihood using (5) and the output of the particle filter. To circumvent this problem, we propose to make use of ABC methods (Marin

et al., 2012) to reformulate the model. This is done so that we only are required to simulate from $g_\theta(y_t \mid x_t)$ and not evaluate it. After this reformulation, we can apply a standard SMC algorithm to estimate the log-likelihood.

Particle filtering with approximate Bayesian computations

ABC is based on the idea that we augment the posterior with an auxiliary variable $\overset{\vee}{y}_{1:T}$, which is the data that we simulate from $g_\theta(y_t \mid x_t)$ for $t = 1, \ldots, T$. The resulting posterior can be expressed as

$$p_\epsilon(\theta, \overset{\vee}{y}_{1:T} \mid y_{1:T}) = \frac{p(\theta) p_\theta\left(\overset{\vee}{y}_{1:T}\right) \rho_\epsilon\left(\overset{\vee}{y}_{1:T} - y_{1:T}\right)}{\int_\Theta p(\theta') p_{\theta'}\left(\overset{\vee}{y}_{1:T}\right) \rho_\epsilon\left(\overset{\vee}{y}_{1:T} - y_{1:T}\right) \mathrm{d}\theta'}.$$

For a small enough *tolerance parameter* $\epsilon > 0$, we assume that we can marginalise the augmented posterior with respect to $\overset{\vee}{y}_{1:T}$ to obtain

$$p_\epsilon(\theta \mid y_{1:T}) = \int p_\epsilon(\theta, \overset{\vee}{y}_{1:T} \mid y_{1:T}) \, \mathrm{d}\overset{\vee}{y}_{1:T},$$

where $p_\epsilon(\theta \mid y_{1:T})$ denotes the posterior of some perturbed model. Here, we make use of some density ρ_ϵ with tolerance parameter ϵ (in non-parametric statistics, ϵ is usually referred to as the bandwidth of the kernel ρ_ϵ) to compute the distance between the simulated observations $\overset{\vee}{y}_t$ and the true observations y_t. A common choice that we make use of in this paper is the Gaussian density with standard deviation ϵ, i.e., $\rho_\epsilon = \mathcal{N}(0, \epsilon^2)$. The fundamental assumption of ABC is therefore that data $\overset{\vee}{y}_t$ generated from $g_\theta(y_t \mid x_t)$ should be similar to the observed data y_t if θ is selected properly.

To make use of this in the particle filter, we reformulate the model following the procedure discussed by Jasra et al. (2012). First, the observed data $y_{1:T}$ is perturbed by noise generated from the density ρ_ϵ resulting in

$$\overset{\vee}{y}_t = \psi(y_t) + z_t, \quad z_t \sim \rho_\epsilon, \tag{7}$$

where ψ denotes some suitable one-to-one transformation. For example, Yıldırım et al. (2014) propose to make use of $\psi(x) = \arctan(x)$ to stabilise the variance of the likelihood (and gradient estimate) of an SSM. We later adopt this transformation in Section 6.2, where we consider a stochastic volatility model with α-stable log-returns.

Secondly, we assume that it is possible to simulate from the model using a transformation of some random variables. This corresponds to that we can write $\overset{\vee}{y}_t = \tau_\theta(v_t, x_t)$ for some deterministic function τ_θ, where $v_t \sim \nu_\theta(v_t \mid x_t)$ denotes some random variable that can be easily generated using standard methods. An example of this is simulation from the zero-mean symmetric α-stable distribution $\mathcal{A}(\alpha, \gamma)$ discussed in Section 6.2. First, we sample $v_t^{(1)} \sim \mathbf{Exp}(1)$ and $v_t^{(2)} \sim \mathcal{U}(-\pi/2, \pi/2)$. Then, we can obtain a sample (when $\alpha \neq 1$) by

$$\overset{\vee}{y}_t = \gamma \frac{\sin\left(\alpha v_t^{(2)}\right)}{\left[\cos(v_t^{(2)})\right]^{1/\alpha}} \left[\frac{\cos\left[(\alpha - 1) v_t^{(2)}\right]}{v_t^{(1)}}\right]^{\frac{1-\alpha}{\alpha}},$$

Algorithm 1 Sequential Monte Carlo with approximate Bayesian computations

Inputs: $\check{y}_{1:T}$ (perturbed data), the reformulated SSM (8), $N \in \mathbb{N}$ (no. particles), ρ_ϵ (density) with $\epsilon \in \mathbb{R}_+$ (tolerance parameter).
Outputs: $\log \widehat{p}_\theta^N(\check{y}_{1:T})$ (est. of log-likelihood).
Note: all operations are carried out over $i, j = 1, \ldots, N$.

1: Sample $\check{x}_0^{(i)} \sim \mu_\theta(x_0)\nu_\theta(v_0 \mid x_0)$ and set $w_0^{(i)} = 1/N$.
2: **for** $t = 1$ to T **do**
3: Resample the particles using systematic resampling with probabilities given by (9).
4: Propagate the particles by (10) and extend the trajectory by $\check{x}_{0:t}^{(i)} = \left\{\check{x}_{0:t-1}^{a_t^{(i)}}, \check{x}_t^{(i)}\right\}$.
5: Calculate the particle weights by (11) which by normalisation (over i) gives $w_t^{(i)}$.
6: **end for**
7: Compute $\log \widehat{p}_\theta^N(\check{y}_{1:T}^\star)$ by (12).

see Nolan (2003) for more information on how to generate α-stable random numbers. By making use of the perturbation in (7) and the simulation of $\check{y}_t$ using v_t, we can rewrite the SSM (1) as

$$\check{x}_t \mid \check{x}_{t-1} \sim \Xi_\theta(\check{x}_t \mid \check{x}_{t-1}) = \nu_\theta(v_t \mid x_t) f_\theta(x_t \mid x_{t-1}), \tag{8a}$$

$$\check{y}_t \mid \check{x}_t \sim h_{\theta,\epsilon}(\check{y}_t \mid \check{x}_t) = \rho_\epsilon\Big(\check{y}_t; \psi(\tau_\theta(\check{x}_t))\Big), \tag{8b}$$

where $\check{x}_t^\top = (x_t^\top, v_t^\top)$ denotes the new state vector. We can now apply a standard particle filter for this new model, which does not require us to evaluate the $g_\theta(y_t \mid x_t)$ point-wise but only that we can simulate from it using $\tau_\theta(v_t, x_t)$.

A particle filter (Doucet and Johansen, 2011) is an iterative algorithm with three steps that allows us to generate a particle system to construct the empirical distribution in (6). Assume that we have the particle system at time $t-1$ and would like to construct the particle system at time t. To achieve this, we apply three operations: (i) resampling, (ii) propagation and (iii) weighting.

In the first step, we sample a so-called *ancestor index* $a_t^{(i)}$ from a multinomial distribution given by

$$\mathbb{P}\left(a_t^{(i)} = j\right) = w_{t-1}^{(j)}, \qquad \text{for } i, j = 1, \ldots, N. \tag{9}$$

The ancestor index $a_t^{(i)}$ can be interpreted as the parent from which the particle $x_t^{(i)}$ is a descendent. This step is carried out to randomly multiply particles with a large weight and discard particles with a small weight. While introducing some variance into the estimate, this step is essential to focus our attention to the parts of the state space of interest. In this paper, we make use of the implementation known as *systematic resampling* (Kitagawa, 1996), which is usually recommended for many practical applications.

In the second step, we propagate the particle system to time t by simulating the system

forward in time. This is done by sampling

$$\overset{\circ}{x}_t^{(i)} \sim \Xi_\theta\left(\overset{\circ}{x}_t \,|\, \overset{\circ}{x}_{t-1}^{a_t^{(i)}}\right), \qquad \text{for } i = 1, \ldots, N. \tag{10}$$

Finally in the third step, we compute the (unnormalised) weight of each particle by

$$W_t^{(i)} = h_{\theta,\epsilon}\left(\overset{\circ}{y}_t \,|\, \overset{\circ}{x}_t^{(i)}\right), \qquad \text{for } i = 1, \ldots, N. \tag{11}$$

which after normalisation (over i) gives $w_t^{(i)}$. The complete procedure is presented in Algorithm 1 and has a computational cost in the order $\mathcal{O}(\text{NT})$, where N denotes the number of particles. This implementation is known as the bootstrap particle filter (BPF) but there are other interesting alternatives. For example, the alive particle filter (Jasra, 2015) is useful when the boxcar kernel is used, i.e., $\rho_\epsilon(x) = \mathcal{U}_{-\epsilon/2,\epsilon/2)}(x)$, as the particle filter otherwise collapses when all weights are zero. Another interesting alternative when using the boxcar kernel is to adapt ϵ using the approach discussed by Del Moral et al. (2012).

The estimator and its statistical properties

An estimator for the log-likelihood of the perturbed model (8) is given by

$$\log \widehat{p}_\theta^N(y_{1:T}) = \sum_{t=1}^{T} \log\left\{\sum_{i=1}^{N} W_t^{(i)}\right\} - T \log N, \tag{12}$$

where we use the unnormalised weights from the particle system generated by Algorithm 1.

It is known from Pitt et al. (2012) that the estimator (12) is biased for a finite number of particles but it is still consistent and asymptotically normal. Furthermore, we have that the error in the log-likelihood estimate fulfils a CLT given by

$$\sqrt{N}\left[\log p_\theta(y_{1:T}) - \log \widehat{p}_\theta^N(y_{1:T}) + \frac{\gamma^2(\theta)}{2N}\right] \xrightarrow{d} \mathcal{N}\left(0, \gamma^2(\theta)\right), \qquad N \to \infty, \tag{13}$$

for some unknown variance $\gamma(\theta)$. 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 particle filter on the same data.

Note that the log-likelihood estimator in (12) is asymptotically unbiased with respect to the perturbed model (8), not the true model. Therefore, inference for $\widehat{\theta}$ using (12) corresponds to inference in a misspecified model as discussed by Wilkinson (2013). This view is also adopted by Dean and Singh (2011) and Dean et al. (2014), where it is concluded that this results in a bias in the parameter estimates (compared with the unperturbed model) that decrease proportional to $\mathcal{O}(\epsilon^2)$ under some regularity assumptions.

However, the asymptotic normality of the log-likelihood estimator holds even with a non-zero ϵ. Furthermore, Dean and Singh (2011) shows that we also retain Bernstein-von Mises type theorem for the posterior estimate based on the estimator for a small enough ϵ. This is an important fact to motivate the Laplace approximation as discussed in Section 2. In practice, it is difficult to quantify the size of the bias and we return to a numerical investigation of this in Section 6.1.

Constructing the surrogate of the log-posterior

In this section, we discuss Steps (ii) and (iii) of GPO-ABC, where we construct a surrogate function to mimic the log-posterior. Good references for further information about general GPO approaches are found in Boyle (2007), Lizotte (2008) and Brochu et al. (2010).

Gaussian process prior

GPs (Rasmussen and Williams, 2006) are an instance of *Bayesian non-parametric models* and can be seen as a generalisation of the multivariate Gaussian distribution to an infinite dimensional setting. 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 infinite vector of values, which can be seen as a function over the real space $\mathbb{R}^p$. This is why the GP can be viewed as a *prior over functions* on $\mathbb{R}^p$. To construct the surrogate function, we assume a priori that the log-posterior is distributed according to a GP as

$$\log p(\theta \,|\, y_{1:T}) \sim \mathcal{GP}\big(m(\theta), \kappa(\theta, \theta')\big). \tag{14}$$

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* $m(\theta)$ and a *covariance function* $\kappa(\theta, \theta')$ defined by

$$\begin{aligned} m(\theta) &= \mathbb{E}\Big[\log p(\theta \,|\, y_{1:T})\Big], \\ \kappa(\theta, \theta') &= \mathbb{E}\Big[\Big(\log p(\theta \,|\, y_{1:T}) - m(\theta)\Big)\Big(\log p(\theta' \,|\, y_{1:T}) - m(\theta')\Big)\Big]. \end{aligned}$$

Here, the mean function specifies the average value of the process and the covariance function 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 log-posterior before we observed the samples from it. From the CLT in (13), we know that the error in the log-posterior is asymptotically Gaussian. Therefore, we assume that

$$\xi_k = \log \widehat{p}^N(\theta_k \,|\, y_{1:T}) = \log p(\theta_k \,|\, y_{1:T}) + \sigma_\xi z_k, \qquad z_k \sim \mathcal{N}(0, 1), \tag{15}$$

where σ_ξ^2 denotes some unknown variance estimated in a later stage of the algorithm. Consequently, we have that both the prior (14) and the estimates of the log-posterior (15) 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 posterior* for any test point $\theta_\star \in \Theta$ given by

$$\log p(\theta_\star \,|\, y_{1:T}) \,|\, \mathcal{D}_k \sim \mathcal{GP}\Big(\mu(\theta_\star \,|\, \mathcal{D}_k), \sigma^2(\theta_\star \,|\, \mathcal{D}_k) + \sigma_\xi^2\Big), \tag{16}$$

where we have introduced the notation $\mathcal{D}_k = \{\boldsymbol{\theta}_k, \boldsymbol{\xi}_k\}$ for the information available about the parameter log-posterior at iteration k. Here, the posterior mean and covariance functions are given by

$$\mu(\theta_\star \,|\, \mathcal{D}_k) = m(\theta_\star) + \kappa(\theta_\star, \boldsymbol{\theta}_k)\Big[\kappa(\boldsymbol{\theta}_k, \boldsymbol{\theta}_k) + \sigma_\xi^2 \mathbf{I}_{k\times k}\Big]^{-1}\Big\{\boldsymbol{\xi}_k - m(\theta_\star)\Big\}, \tag{17a}$$

$$\sigma^2(\theta_\star \,|\, \mathcal{D}_k) = \kappa(\theta_\star, \theta_\star) - \kappa(\theta_\star, \boldsymbol{\theta}_k)\Big[\kappa(\boldsymbol{\theta}_k, \boldsymbol{\theta}_k) + \sigma_\xi^2 \mathbf{I}_{k\times k}\Big]^{-1}\kappa(\boldsymbol{\theta}_k, \theta_\star). \tag{17b}$$

Hence, we can construct the surrogate function of the log-posterior by using (16) obtained from the GP. The major cost in computing the predictive posterior is incurred by the matrix inversion which has a computational cost of order $\mathcal{O}(K^3)$, where K denotes the maximum number of iterations. However, in all our applications it is sufficient with $K < 300$, which results in a calculation that is easily handled by modern computers and is much less demanding than running the particle filter. However, it could be of interest to use a recursive and sparse formulation of the GP to decrease computational cost and memory requirements. Especially, when p is large and therefore K needs to grow to properly explore the parameter space. We return to discuss this issue is Section 7.

It is common that the log-posterior varies widely over the parameter space Θ, where it in some areas can be almost flat and fall off quickly in others. This problem often arises when the number of parameters p increases above four or five. It is therefore important to make use of ARD covariance functions (with different length scales for each parameters) and to normalise the log-posterior estimates before calculating the predictive posterior. In some models, this is not enough and it could be necessary to consider non-stationary covariance functions. However, we did not experience any major problems in the models considered in this paper but it is always recommended to plot at the marginal predictive distributions to check for potential problems.

Design of priors and estimating hyperparameters

In this paper, we assume a zero prior mean function $m(\theta) = 0$ and the combination of a bias and the Matérn 5/2 covariance functions with ARD given by

$$\kappa(\theta, \theta') = \sum_{r=1}^{p} \left[\kappa_{0,r} + \sigma_\kappa^2 \left(1 + \frac{\sqrt{5}(\theta_r - \theta_r')}{\ell_r} + \frac{5(\theta_r - \theta_r')^2}{3\ell_r^2} \right) \exp\left(- \frac{\sqrt{5}(\theta_r - \theta_r')}{\ell_r} \right) \right], \quad (18)$$

where $\lambda = \{\kappa_{0,1:p}, \sigma_\kappa^2, \ell_{1:p}\}$ denotes the hyperparameters of the covariance function. Note, that the bias term $\kappa_{1:p}$ and *length scales* $\ell_{1:p}$ are vector valued with p elements (one for each parameter in θ). This type of prior is equivalent with using a constant mean function, i.e., $m(\theta) = c$ for some constant $c \in \mathbb{R}$, and the Matérn 5/2 function to describe the covariance structure with a non-zero mean.

The choice of covariance function in (18) results in a prior for the log-posterior with some non-zero mean and two continuous derivatives. These are reasonable assumptions as this kind of smoothness is assumed in the Laplace approximation. Both stronger and weaker smoothness properties can be assumed by replacing the Matérn 5/2 covariance function with either a squared exponential or Matérn 3/2 covariance function, respectively. Another interesting choice for the mean function is the log-prior, i.e., $m(\theta) = \log p(\theta)$. See Rasmussen and Williams (2006) for more information regarding the design of GP prior.

In GP models, it is common to infer the hyperparameters of the covariance function from the data. There are a number of different approaches to this based on both maximum likelihood and Bayesian inference. Some common Bayesian approaches include *Hamiltonian Markov chain Monte Carlo* (Saatçi et al., 2010; Neal, 2010) and SMC (Gramacy and Polson, 2011; Svensson et al., 2015).

In this paper, we make use of a maximum likelihood approach and estimate the hyperparameters using *empirical Bayes* (EB), i.e., by optimising the marginal likelihood with respect to λ. The marginal likelihood can be computed in closed-form by marginalisation and can be optimised by a standard gradient-based algorithm. This procedure is often carried out for a number of random initialisations, so that the optimisation does not get stuck in some local optima.

Acquisition function

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 a heuristic that selects the next sampling point given the information available up until the current iteration of the algorithm. Using the acquisition rule $\mathsf{AQ}(\theta_\star \,|\, \mathcal{D}_k)$, the next point in which to sample the log-posterior is selected as

$$\theta_{k+1} = \operatorname*{argmax}_{\theta_\star \in \Theta_{\text{GPO}}} \mathsf{AQ}(\theta_\star \,|\, \mathcal{D}_k),$$

where Θ_{GPO} denotes a *search space* defined by the user. Note that this optimisation is relatively cheap to carry out as we only need to evaluate the GP predictive posterior for a number of test points $\theta_\star$ and not sample the log-posterior in these points.

There are a number of different acquisition rules in the literature and constructing new alternatives is a current and active research field. In this paper, we make use of the *expected improvement* (EI) (Jones, 2001) as it is generally recommended for GPO applications (Lizotte, 2008). To derive the EI rule, consider the *predicted improvement* defined as

$$\mathsf{PI}(\theta_\star) = \max\left\{0, \log p(\theta_\star \,|\, y_{1:T}) - \mu_{\max} - \zeta\right\}, \qquad \forall \theta_\star \in \Theta_{\text{GPO}}, \tag{19}$$

where $\mu_{\max}$ denotes the maximum value of $\mu(\theta)$ for the sampled points $\theta \in \boldsymbol{\theta}_k$. Here, we introduce ζ as an user defined parameter that controls the exploitation/exploration behaviour as discussed by Lizotte (2008). Hence for $\zeta = 0$, we have that $\mathsf{PI}(\theta_\star)$ is the difference between the posterior and the maximum value it assumes in the sampled points. Therefore, it is positive for points where the log-posterior is larger than the current peak and zero for all other points.

From (16), we know that the predictive posterior is Gaussian with mean $\mu(\theta_\star \,|\, \mathcal{D}_k)$ and variance $\sigma^2(\theta_\star \,|\, \mathcal{D}_k)$. Using this, we can compute the expectation of (19) with respect to the predictive posterior. From this calculation, we obtain the EI rule (Jones, 2001) as

$$\begin{aligned}
\mathsf{EI}(\theta_\star \,|\, \mathcal{D}_k) &= \mathbb{E}\Big[\mathsf{PI}(\theta_\star) \,|\, \mathcal{D}_k\Big] \\
&= \int_{\mu_{\max}+\zeta}^{\infty} \mathsf{PI}(\theta_\star) \frac{1}{\sqrt{2\pi\sigma^2(\theta_\star \,|\, \mathcal{D}_k)}} \exp\left[-\frac{1}{2}\left(\frac{\theta_\star - \mu(\theta_\star \,|\, \mathcal{D}_k)}{\sigma^2(\theta_\star \,|\, \mathcal{D}_k)}\right)^2\right] \mathrm{d}\theta_\star, \\
&= \sigma(\theta_\star \,|\, \mathcal{D}_k)\Big[Z(\theta_\star)\Phi\big(Z(\theta_\star)\big) + \phi\big(Z(\theta_\star)\big)\Big], \text{ with} \\
Z(\theta) &= \sigma^{-1}(\theta_\star \,|\, \mathcal{D}_k)\Big[\mu(\theta_\star \,|\, \mathcal{D}_k) - \mu_{\max} - \zeta\Big],
\end{aligned} \tag{20}$$

where ϕ and Φ denotes the density and distribution function of the standard Gaussian distribution, respectively. Here, we introduce the standardised variable From practical expe-

rience 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 so-called *jittering* amount to adding some noise to the estimate, i.e.,

$$\theta_{k+1} = \underset{\theta_\star \in \Theta_{\text{GPO}}}{\operatorname{argmax}} \, \mathsf{EI}(\theta_\star \mid \mathcal{D}_k) + z_k, \qquad z_k \sim \mathcal{N}(0, \Sigma), \tag{21}$$

where Σ denotes a covariance matrix determined by the user. Similar approaches are discussed by Bull (2011) and Gutmann and Corander (2015) to improve the convergence rate of the algorithm and the exploration of the parameter space, respectively.

The optimisation in (21) is possibly non-convex and therefore difficult to carry out in a global setting. Two common approaches in GPO are: (i) using a few local gradient-based algorithms initialised in randomly selected points in Θ_{GPO} (Lizotte, 2008), and (ii) using some global optimisation algorithm e.g., the gradient-free dividing rectangles (DIRECT; Jones et al., 1993) algorithm over Θ_{GPO}. Note that we can determine a suitable search space Θ_{GPO} from the support of the prior distribution $p(\theta)$.

The GPO-ABC algorithm

In this section, we combine the developments in the two previous sections to put together GPO-ABC and discuss its properties. In Algorithm 2, we outline the complete procedure that combines SMC-ABC from Algorithm 1 to approximate the log-posterior and GPO to create a surrogate function that mimics the log-posterior.

Initialisation and convergence criteria

We initialise Algorithm 2 at Line 1 to find some suitable hyperparameters for the GP prior. This is important as GPO-ABC may fail to converge to an accuracy parameter estimate if it is not properly initialised. The hyperparameters are determined by EB using L samples from the log-posterior obtain by some sampling scheme. There are at least three different alternative for this: (i) uniform sampling, (ii) quasi Monte Carlo sampling and (iii) Latin hypercube sampling. See Glasserman (2004) for a discussion. All these approaches are carried out over the set Θ_{GPO} discussed in the previous section.

We then execute Algorithm 1 for each of the sampled parameters $\{\theta_1^\star, \theta_2^\star, \ldots, \theta_L^\star\}$ to obtain $\mathcal{D}_L^\star$ by the analogue of Line 4 in Algorithm 2. After the initialisation of the algorithm, we can update the hyperparameters with Line 5 at every iteration or at some pre-defined interval. Estimating the hyperparameters is computationally costly and it is therefore recommended to reestimate them at some fixed interval.

GPO-ABC can be executed for some pre-defined number of iterations K or using some other suitable convergence criteria. An interesting alternative in the GPO literature is to run the algorithm until the EI is smaller than some $\Delta\mathsf{EI} > 0$, i.e., until k satisfies $\mathsf{EI}(\theta_k \mid \mathcal{D}) < \Delta\mathsf{EI}$.

Algorithm 2 Gaussian process optimisation with sequential Monte Carlo using approximate Bayesian computations (GPO-ABC)

Inputs: Algorithm 1, $p(\theta)$ (parameter prior), $m(\theta)$ (mean function), $\kappa(\theta, \theta')$ (covariance function), θ_1 (initial parameter), Σ (jittering covariance matrix) and Θ_{GPO} (optimisation bounds).
Output: $\widehat{\theta}_{\text{MAP}}$ (est. of the parameter) and $\widehat{\mathcal{J}}(\widehat{\theta}_{\text{MAP}})$ (est. of posterior covariance).

1: Initialise the GP prior by running EB on some initial sampled data $\mathcal{D}_L^\star$.
2: Initialise the parameter estimate in θ_1 and set $k = 1$.
3: **while** *convergence criteria is not satisfied* **do**
4: Estimate $\log \widehat{p}^N(\theta_k \,|\, y_{1:T}) = \log \widehat{p}^N_{\theta_k}(y_{1:T}) + \log p(\theta_k)$ using Algorithm 1 and set $\mathcal{D}_k = \{\mathcal{D}_L^\star, \boldsymbol{\theta}_k, \boldsymbol{\xi}_k\}$.
5: (*if required*) Run EB to update the hyperparameters of the GP prior using $\mathcal{D}_k$.
6: Construct $\log p(\theta_\star \,|\, y_{1:T}) \,|\, \mathcal{D}_k$ using (16) and (17).
7: Compute $\mu_{\max} = \operatorname{argmax}_{\theta \in \boldsymbol{\theta}_k} \mu(\theta \,|\, \mathcal{D}_k)$.
8: Compute θ_{k+1} by (21) with (20) using e.g., DIRECT over Θ_{GPO}.
9: Set $k = k + 1$.
10: **end while**
11: Compute the MAP estimate $\widehat{\theta}$ by optimising $\mu(\theta \,|\, \mathcal{D}_{k-1})$ using DIRECT over Θ_{GPO}.
12: Extract the Hessian estimate $\mathcal{J}(\widehat{\theta}_{\text{MAP}})$ using e.g., finite differences.

Extracting the Laplace approximation

After running Algorithm 2, we compute the predictive posterior mean function $\mu(\theta_\star \,|\, \mathcal{D})$, which hopefully is an accurate surrogate for the log-posterior. We can then proceed to extract the MAP estimate $\widehat{\theta}_{\text{MAP}}$ defined by (4). As the surrogate is smooth and cheap to evaluate, we can carry out the optimisation using standard methods, e.g., the DIRECT algorithm, gradient-based methods or a quasi-Newton algorithm. Hence, we circumvent the two problems discussed in Section 2, i.e., that we can only get noisy point-wise estimates of the log-posterior.

To finally construct the Laplace approximation of the posterior (3), we require an estimate of the Hessian of the log-posterior at $\widehat{\theta}_{\text{MAP}}$ denoted by $\mathcal{J}(\widehat{\theta}_{\text{MAP}})$. This can be computed by taking the Hessian with respect to $\theta_\star$ of the predictive mean function (17). Hence, computing the Hessian of the mean function amounts to computing Hessians of the covariance functions. This can be implemented efficiently for some covariance functions (e.g., the squared exponential).

For other choices of covariance functions, we can apply some finite difference scheme to approximate the Hessian numerically. We note that this estimate is obtained directly as a by-product when using a quasi-Newton algorithm to solve (4). When we use the DIRECT algorithm, we can compute an estimate of the Hessian in a separate step using finite differences with an adaptive scheme with Romberg extrapolation to obtain estimates with a good accuracy. Another approach is to make use of a SMC-ABC smoother to estimate the Hessian (Poyiadjis et al., 2011; Dahlin et al., 2015a). However, this is computationally costly and can result in a negative-definite estimate of the posterior covariance.

Convergence properties

As previously discussed, GPO(-ABC) is an efficient approach for global optimisation when the objective function is costly to evaluate and possibly noisy. The convergence properties typically depends on the choice of the GP prior and are discussed by Bull (2011) and Vazquez and Bect (2010). They conclude that GPO using the EI rule samples the log-posterior densely if it is continuous with respect to the GP prior. Also, GPO achieves an *optimal* convergence rate of the order $\mathcal{O}\big((n \log n)^{-5/p} (\log n)^{1/2}\big)$ for the Matérn 5/2 covariance function, where n and p denote the number of samples and parameters to infer, respectively.

Numerical illustrations and real-world applications

In this section, we provide three illustrations of the usefulness of the proposed algorithm. In the first illustration, we study the accuracy and convergence rate of the GPO-ABC algorithm as well as the impact of the tolerance parameter ϵ on the accuracy of the posterior approximation. In the second illustration, we make use of GPO-ABC for modelling log-returns of future contracts on coffee. In the third illustration, we consider an application from financial risk management, where we would like to calculate the VAR for a portfolio of oil futures. This problem serves as an illustration of a setting where the GPO-ABC algorithm can be a competitive alternative in an important practical setting. All implementation details are collected in Appendix A and the source code is available from `https://github.com/compops/gpo-abc2015/`.

Stochastic volatility model with Gaussian log-returns

Consider the stochastic volatility model with Gaussian log-returns (GSV) given by

$$x_0 \sim \mathcal{N}\Big(x_0; \mu, \sigma_v^2/\big(1-\phi^2\big)\Big), \tag{22a}$$

$$x_{t+1} \sim \mathcal{N}\Big(x_{t+1}; \mu + \phi(x_t - \mu), \sigma_v^2\Big), \tag{22b}$$

$$y_t \sim \mathcal{N}\Big(y_t; 0, \exp(x_t)\Big), \tag{22c}$$

with parameters $\theta = \{\mu, \phi, \sigma_v\}$. Here, the latent log-volatility is assumed to follow a mean-reverting random walk with mean $\mu \in \mathbb{R}$, persistence $\phi \in (-1, 1)$ and standard deviation of the increments $\sigma_v \in \mathbb{R}_+$. We generate a single synthetic data set from this model with $T = 500$ observations, *true parameters* $\theta^\star = \{0.20, 0.96, 0.15\}$ and initial state $x_0 = 0$.

For this model, we can make use of standard SMC to estimate the log-likelihood as we can evaluate $g_\theta(y_t \,|\, x_t)$ using a closed-form expression. Therefore, we implement Algorithm 2 using standard SMC to obtain GPO-SMC introduced by Dahlin and Lindsten (2014). This corresponds in some sense to the ideal GPO-ABC implementation with tolerance parameter $\epsilon \searrow 0$. Furthermore, we make use of the quasi-Newton PMH2 (QPMH2) algorithm introduced by Dahlin et al. (2015b) to estimate the parameter posteriors. This can be seen as the gold standard to which the Laplace approximations obtained by GPO-SMC and GPO-ABC care compare. This is done to evaluate their accuracy.

In the left part of Figure 1, we present the posterior estimates from GPO-SMC (solid curve) and QPMH2 (histogram). We begin by observing a good fit of the Laplace approximation

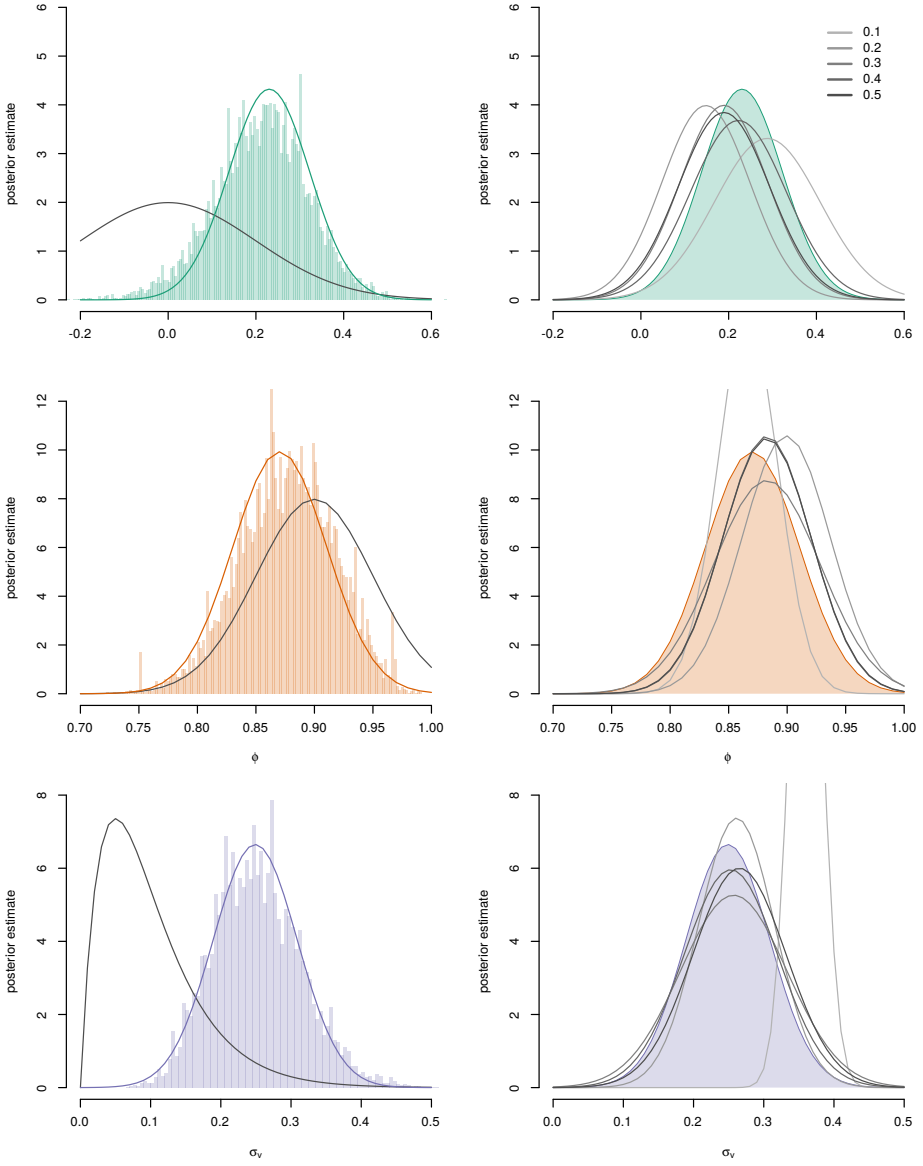


Figure 1. Marginal parameter posteriors for the synthetic data in the SVG model. Left: Solid curves indicate the Laplace approximations of the posterior using GPO-SMC for μ (green), ϕ (orange) and σ_v (purple). The histograms represent the exact posteriors estimated using QPMH2 and the dark grey (left) curves indicate the prior distributions. Right: Laplace approximations (shaded areas) from GPO-SMC for the three parameters. The grey curves (right) indicate the Laplace approximations obtained by GPO-ABC using five different values of the tolerance parameter ϵ in the ABC approximation.

from GPO-SMC to the histogram approximation obtained by QPMH2. Both the location and the spread of the posterior approximations are similar and this shows that GPO-SMC could be a more efficient alternative to QPMH2. This as GPO-SMC results in a speed-up of about $40 - 60$ times compared with QPMH2.

We now continue with analysing the Laplace approximations obtained by GPO-ABC. In the right side of Figure 1, we present the posterior approximations obtained by varying the tolerance parameter ϵ between 0.1 and 0.5 to study the bias and robustness of the approximation. Darker shades of grey indicate a larger tolerance parameter. For $\epsilon = 0.10$, we see that the approximation is rather poor with a significant bias and bad fit to the spread. However for the other choices of ϵ, the approximations from GPO-ABC converges quickly to be similar to GPO-SMC (solid curve). From these results, we conclude that we have a bias in the posterior approximation when ϵ is too small and the approximation tends to grow wider as ϵ increases. Hence, the posterior approximation experiences a bias-variance trade-off effect that is determined by the tolerance parameter ϵ.

We continue by comparing GPO-SMC to SPSA, where the latter is a gradient-free alternative with good convergence properties in many applications (Spall, 1987). SPSA operates by constructing a finite-difference approximation of the gradient at each iteration after which it takes a step in the gradient direction. We tune the algorithm using the procedure discussed by Spall (1998) to achieve good performance. Note that SPSA requires two log-posterior estimates at each iteration compared with only one sample in the GPO-SMC. Another possible drawback with SPSA is that it only provides the MAP estimate and no quantification of its uncertainty.

In Figure 2, we compare the MAP parameter estimates of two algorithms as a function of the number of log-posterior estimates. The first $L = 100$ samples of the log-posterior are used to estimate the hyperparameters of the GP prior. After this initial phase, GPO-SMC converges quickly to reasonable values of the parameters. As the log-posterior estimates are rather noisy, we require that SPSA adapts slowly which results in a slow convergence of the parameter estimates. Finally, we note that the three methods that we consider in this section make use of varying number of estimates of the log-posterior to determine the parameter estimates: GPO-SMC (350), SPSA (700) and QPMH2 (10,000). However, the estimates are comparable for GPO-SMC and QPMH2, see the left part of Figure 1.

Stochastic volatility model with α-stable log-returns

In the upper part of Figure 3, we present the log-returns of future contracts on coffee during the period between June, 2013 and December, 2014. The data seem to indicate the presence of jumps around the first half of 2014. This is common in financial data and can be modelled in a number of different ways. Here, we follow Casarin (2004) by making use of the α-stable distribution to model the jumps in the log-returns. The use of α-stable distributions is motivated and exemplified by Stoyanov et al. (2010), Rachev et al. (2005), Khindanova et al. (2001), Lombardi and Calzolari (2009) and references therein.

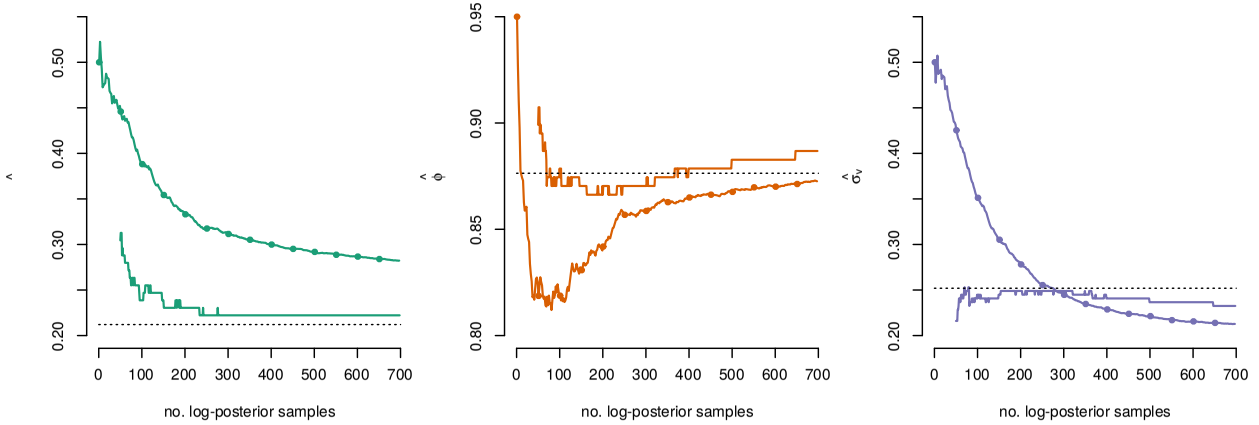


Figure 2. The trace of the MAP estimate for μ (green), ϕ (orange) and σ_v (purple) from GPO-SMC (solid) and SPSA (solid-cicle) as a function of the number of log-posterior samples. The first $L = 100$ samples of GPO-SMC are used to estimate the hyperparameters. Both algorithms are run for a total of 700 log-posterior samples. Dotted lines indicate the posterior means from QPMH2.

To model the log-returns, we consider the stochastic volatility model with α-stable log-returns (αSV) given by

$$x_0 \sim \mathcal{N}\left(x_0; \mu, \sigma_v^2/(1-\phi^2)\right), \tag{23a}$$

$$x_{t+1} \sim \mathcal{N}\left(x_{t+1}; \mu + \phi(x_t - \mu), \sigma_v^2\right), \tag{23b}$$

$$y_t \sim \mathcal{A}\left(y_t; \alpha, \exp(x_t)\right), \tag{23c}$$

with parameters $\theta = \{\mu, \phi, \sigma_v, \alpha\}$ and $\mathcal{A}(\alpha, \gamma)$ denoting a zero-mean symmetric α-stable distribution with stability parameter $\alpha \in (0, 2)$ and scale $\gamma \in \mathbb{R}_+$. The stability parameter determines the tail behaviour of the distribution, see Nolan (2003) for a discussion of the α-stable distribution and its properties. The likelihood is in general intractable for the α-stable distribution but for specific choices of α we can recover three named distributions: Gaussian ($\alpha = 2$), Cauchy ($\alpha = 1$) and Levy ($\alpha = 1/2$). As a result, we can only in general make use of QPMH2-ABC and GPO-ABC for this model, where both approaches are approximate as they rely on the ABC approximation of the model.

In the middle and lower part of Figure 3, we compare the posterior approximations obtained with GPO-ABC (solid curves) with 10 independent runs and QPMH2-ABC (histograms). We see that the mixing of QPMH2-ABC is quite poor for this model as the histograms are *peaky*. This is a common problem as Markov chains tends to get stuck if the log-posterior estimates are noisy. However, the posterior estimates overlap and seems to give reasonable parameter values for each run of GPO-ABC. We conclude that GPO-ABC can be a good alternative to PMH-ABC for inference in αSV models, as it enjoys similar accuracy but at a significant lower computational cost. Finally, we make use of the ABC version of the forward-filtering backward simulator (FFBSI; Taghavi et al., 2013) smoother using rejection sampling and early stopping to estimate the log-volatility. The resulting estimate (black) seems reasonable when compared to the log-returns.

Modelling price dependencies between oil futures

In this section, we follow Charpentier (2015) and would like to construct a copula model to capture the dependency structure between prices of oil future contracts. The data that we consider is presented in Figure 4 and consists of weekly log-returns between January 10, 1997 and June 4, 2010 of Brent (produced in the North Sea), Dubai (produced in the Persian Gulf) and Maya (produced in the Gulf of Mexico) oil. For this problem, we adopt the commonly used two-stage approach to copula modelling where marginal models are first fitted separately to each of the log-return series, and then combined using a copula to model the dependency structure (Joe, 2005). We outline this procedure in Algorithm 3.

Lines 1 and 2 We make use of the same procedure as in Section 6.2 to estimate the parameters of an αSV model to describe the log-returns for each type of oil. The resulting parameter estimates are presented in Table 1, which are similar with the exception of the mean-reverting parameter μ. From the model, we expect that the log-volatility varies slowly over time around different mean values and that the log-returns are more heavy-tailed than a Gaussian distribution. In the left part of Figure 4, we present the estimates of the log-volatility and the resulting credibility regions which confirms our expectations.

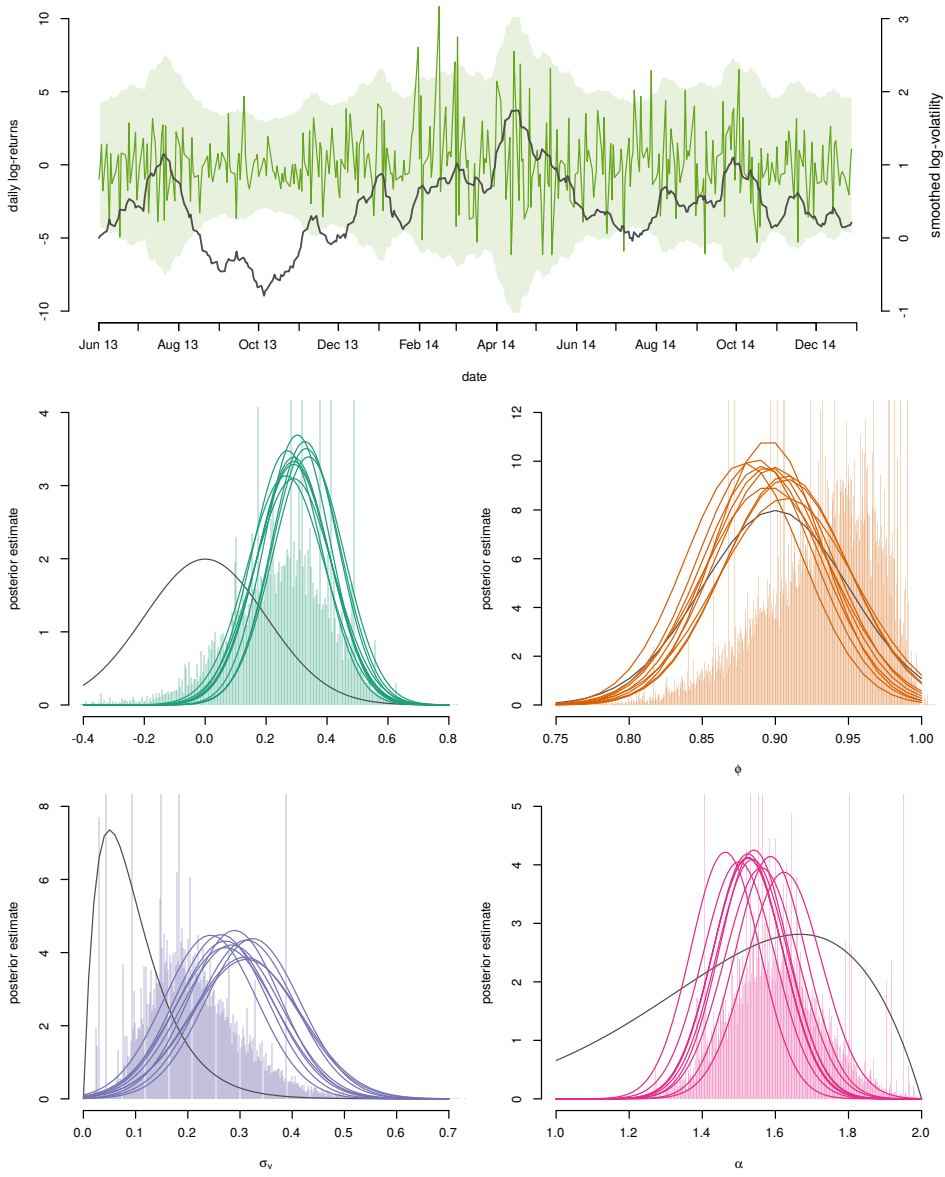


Figure 3. Upper: log-returns (green) of coffee futures and the estimate of the log-volatility (dark grey). The shaded area indicates the approximate 95% credibility region for the log-returns. Middle and lower: marginal parameter posterior in the αSV model estimated by GPO-ABC (solid curves) using 10 independent runs and QPMH2-ABC (histogram) for μ (green), ϕ (orange), σ_v (purple) and α (magenta). Dark grey curves indicate the prior distributions.

Algorithm 3 Copula modelling using αSV as the marginal models

Stage 1 (repeated for each type of oil, i.e., $i = 1, 2, 3$)

1: Estimate $\widehat{\theta}_{\text{MAP},i}$ for the αSV model (23) using GPO-ABC in analogue with Section 6.2.

2: Estimate the log-volatility $\widehat{x}_{1:T,i}$ using the ABC-FFBSi smoother and the parameter estimate $\widehat{\theta}_{\text{MAP},i}$. Compute the filtered log-returns $\widehat{e}_{t,i}$ by

$$\widehat{e}_{t,i} = \exp\Big(-\frac{1}{2}\widehat{x}_{t,i}\Big)y_t, \qquad \text{for } t = 1, \dots, T. \tag{24}$$

3: Estimate the cumulative distribution function (CDF) denoted by $\widehat{G}_i$ from the data. Compute the probability integral transformation of the residuals by

$$\widehat{u}_{t,i} = \widehat{G}_i(\widehat{e}_{t,i}), \qquad \text{for } t = 1, \dots, T. \tag{25}$$

Stage 2

4: Infer the parameters of the copula function to model the dependency between $\{\widehat{u}_{t,1}, \widehat{u}_{t,2}, \widehat{u}_{t,3}\}_{t=1}^{T}$.

Line 3 The filtered residuals given by (24) are independent and identically distributed as $\mathcal{A}(\widehat{\alpha}_i, 0, 1, 0)$ according to the αSV model (23) if $x_{1:T}$ is known. Here, we assume that this holds approximately, so that we can transform these residuals into the 3-dimensional hypercube and make use of the copula model. This can be done by the probability integral transform, i.e., $G_{\theta,i}(\widehat{u}_{t,i})$, which unfortunately is intractable for the α-stable distribution. A possible approach is instead to calculate the empirical CDF (ECDF) from the data and then compute the transformed residuals using this estimate.

A drawback of using the ECDF directly is that it suffers from poor accuracy in the tails due to the small number of samples in these regions. An alternative is instead to compute a semi-parametric estimate of the CDF. This is done by combining a parametric model known as generalised Pareto distribution (GPD) for each of the tails and the non-parametric ECDF estimator for the main bulk of the data. The density of the GPD is given by

$$p_{\eta,\mu,\sigma}(\widehat{e}_{t,i}) = \sigma^{-1}\big(1 + \eta\sigma^{-1}(\widehat{e}_{t,i} - \mu)\big)^{1+\eta^{-1}},$$

where $\mu \in \mathbb{R}$, $\sigma \in \mathbb{R}_+$ and $\eta \in \mathbb{R}$ denotes the parameters determining the location, scale and shape of the distribution, respectively.

The combination is made by non-parametric regression using the Gaussian kernel to stitch the three components of the estimated CDF together. The use of GPD for the tails is known as the *threshold excess method* and is a tool from extreme value statistics. In this approach, we model the 10% smallest and largest filtered residuals using two GPD distributions (Embrechts et al., 1997). To estimate the parameters of the GPDs, we make use of the log-likelihood given by

$$\ell\big(\widehat{e}^{\star}_{1,i}, \dots, \widehat{e}^{\star}_{K,i} \,|\, \eta, \mu, \sigma\big) = -K \log\sigma - \big(1 + \eta^{-1}\big) \sum_{k=1}^{K} \log\big(1 + \eta\sigma^{-1}(\widehat{e}^{\star}_{k,i} - \mu)\big), \tag{26}$$

where $\widehat{e}^{\star}_{i}$ denotes the upper/lower 10% of the filtered residuals. For the parameter inference, we assume uniform priors and compute the MAP estimate by optimizing (26) using a quasi-

		Parameters for marginal model				Parameters for GPD tail distributions					
	Asset	μ	ϕ	σ_v	α	η_-	μ_-	σ_-	η_+	μ_+	σ_+
αSV	Brent	0.56	0.99	0.40	1.67	0.22	0.01	0.58	0.10	0.01	0.54
		(0.18)	(0.06)	(0.09)	(0.10)	(0.11)	(0.05)	(0.03)	(0.11)	(0.04)	(0.02)
	Dubai	0.33	0.99	0.47	1.65	0.05	0.02	0.66	0.01	0.01	0.58
		(0.26)	(0.02)	(0.11)	(0.14)	(0.09)	(0.05)	(0.02)	(0.09)	(0.05)	(0.01)
	Maya	0.56	0.99	0.47	1.67	-0.04	0.02	0.68	0.17	0.02	0.56
		(0.37)	(0.01)	(0.09)	(0.23)	(0.09)	(0.05)	(0.02)	(0.11)	(0.04)	(0.03)
GSV	Brent	1.78	0.99	0.13							
		(0.13)	(0.01)	(0.02)							
	Dubai	1.63	0.99	0.15							
		(0.07)	(0.01)	(0.01)							
	Maya	1.86	0.99	0.14							
		(0.17)	(0.01)	(0.03)							

Table 1. Left: Posterior means and standard deviations (in parentheses) from GPO-ABC for the αSV model in (23) for different oil assets. Right: MAP estimates and posterior standard deviations (in parentheses) for the lower (-) and upper (+) tails of the distribution of the residuals modelled using GPDs.

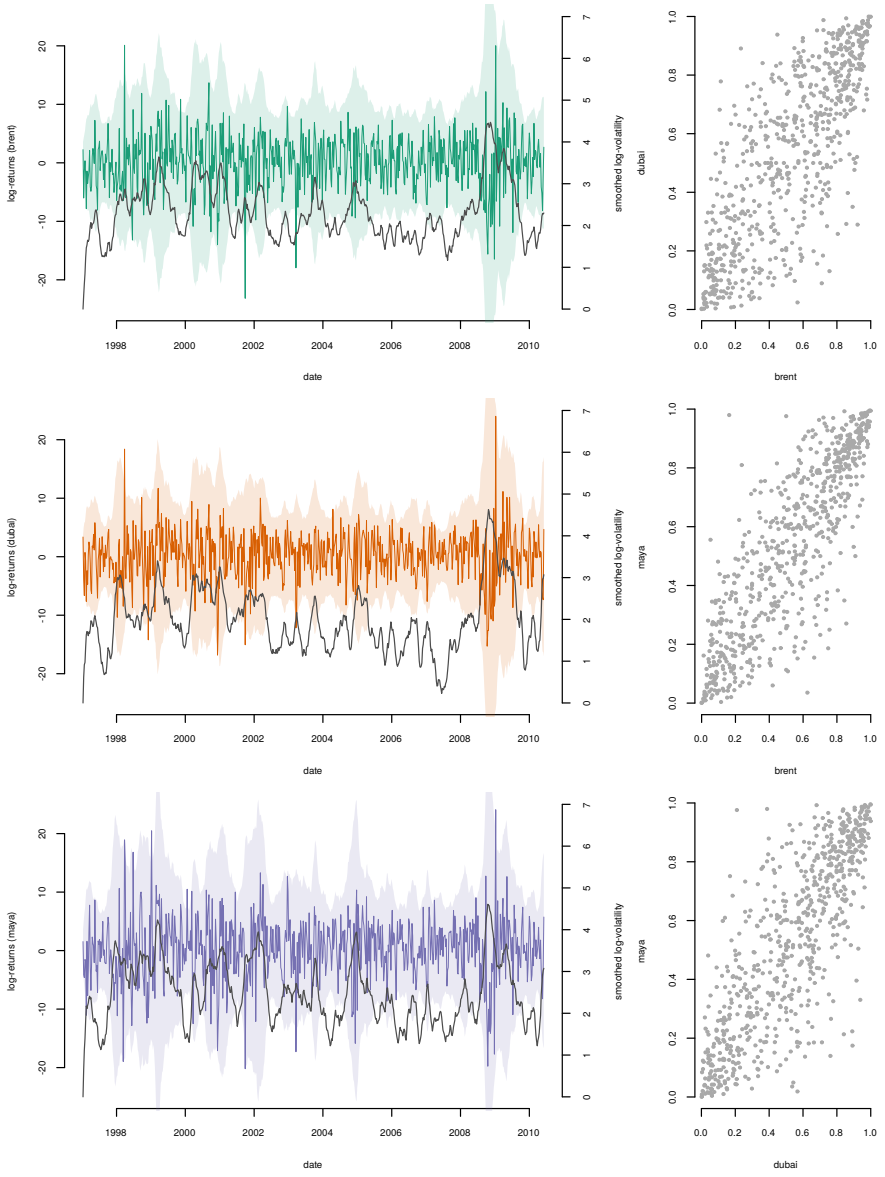


Figure 4. Left: weekly log-returns of Brent (green), Dubai (orange) and Maya (purple) oil between January 10, 1997 and June 4, 2010. The shaded area indicates the approximate 95% credibility region for the log-returns according to the αSV model. The dark grey curves indicate estimates of the log-volatility using the αSV model and the ABC-FFBSI smoother. Right: the transformed filtered residuals $\widehat{u}_{t,i}$ for the different types of oil.

Newton method. The resulting parameter estimates are presented in the right side of Table 1, where we note that the shape parameter of the lower tail differs slightly between each type of oil. The transformed residuals are calculated by using (25) where $\widehat{G}_i$ is given by the semiparametric model of the CDF. The resulting residuals are presented in the right part of Figure 4, where we note a strong correlation between the different types of oil.

Line 4 Finally, we combine the transformed residuals $\{\widehat{u}_{1:T,1}, \ldots, \widehat{u}_{1:T,d}\}$ using a copula function to find a model for the joint distribution. A *copula function* C (Nelsen, 2007; McNeil et al., 2010) is a probability distribution on a d-dimensional hypercube $[0,1]^d$. By *Sklar's theorem*, we can write any multivariate distribution function using a copula of some marginal distributions by

$$H\left(\widehat{u}_{1:T,1}, \ldots, \widehat{u}_{1:T,d}\right) = C\left(H_1(\widehat{u}_{1:T,1}), \ldots, H_d(\widehat{u}_{1:T,d})\right),$$

where $H_i(\widehat{u}_{1:T,i})$ denotes the CDF of the ith marginal distribution of H. A popular example of C is the heavy-tailed Student t-copula, which is a common choice to model financial returns, see Rachev et al. (2005). This type of copula has the form

$$C_{\nu,\mathbf{P}} = \mathbf{t}_{\nu,\mathbf{P}}\left(t_\nu^{-1}(\widehat{u}_{1:T,1}), \ldots, t_\nu^{-1}(\widehat{u}_{1:T,d})\right), \tag{27}$$

where $\mathbf{t}_{\nu,\mathbf{P}}$ denotes the multivariate t distribution with zero mean, scale matrix $\mathbf{P}$ and ν degrees of freedom. Here, t_ν^{-1} denotes the quantile function of the univariate Student's t distribution with the same degrees of freedom. The corresponding log-likelihood function is given by

$$\begin{aligned}\ell(\widehat{u}_{1:T,1}), \ldots, \widehat{u}_{1:T,d}) \,|\, \nu, \mathbf{P}) = &\sum_{t=1}^{T} \log g_{\nu,\mathcal{P}}\left(t_\nu^{-1}(\widehat{u}_{t,1}), \ldots, t_\nu^{-1}(\widehat{u}_{t,d})\right) \\ &- \sum_{t=1}^{T}\sum_{i=1}^{d} \log g_\nu\left(t_\nu^{-1}(u_{t,i})\right),\end{aligned} \tag{28}$$

where $g_{\nu,\mathcal{P}}$ and g_ν denotes the density of the multivariate and univariate Student's t distributions, respectively. Here, we assume that the correlation matrix $\mathcal{P}$ can be written

$$\mathcal{P} = \Delta_A(AA^\top)\Delta_A, \quad \text{with } \Delta_A = \mathsf{diag}(AA^\top)^{-1/2}, \quad A = \begin{bmatrix} 1 & 0 & 0 \\ \rho_1 & 1 & 0 \\ \rho_2 & \rho_3 & 1 \end{bmatrix}.$$

This is done to ensure that $\mathcal{P}$ is a valid correlation matrix, while keeping the individual correlations unrestricted, i.e., $\rho_i \in [-1, 1]$ for $i = 1, 2, 3$.

We compute the MAP estimate of the parameters in the copula function $\{\nu, \rho_1, \rho_2, \rho_3\}$ using a quasi-Newton algorithm. The resulting parameter estimates are presented in Table 2, where we also compare with using an ARMA-GARCH model as in Charpentier (2015) and the GSV model for the margins. For the latter, we present the parameters estimated using GPO-SMC in Table 1. We conclude that the copula parameters are quite similar for the three different margins but the degrees of freedom are smaller for ARMA-GARCH. This could be an indication of that GSV and αSV capture more of the variations in the log-returns.

Finally, we make use of the copula models and their margins to estimate the VAR for each

Margins	Parameters for copula			
	ν	ρ_1	ρ_2	ρ_3
ARMA-GARCH	4.29 (0.65)	-0.12 (0.03)	0.38 (0.02)	0.29 (0.02)
GSV	8.49 (1.77)	-0.08 (0.03)	0.34 (0.02)	0.28 (0.02)
αSV	9.38 (1.97)	-0.10 (0.03)	0.33 (0.02)	0.25 (0.02)

Table 2. MAP estimates and standard deviations (in parentheses) for the parameters in the Student's t copula for different models of the margins.

Margins	Spearman correlations		
	Brent-Dubai	Brent-Maya	Dubai-Maya
ARMA-GARCH	0.73	0.85	0.76
GSV	0.76	0.84	0.79
αSV	0.75	0.82	0.77

Table 3. The estimated correlations between the different types of oil in the Student's t copula for different models of the margins.

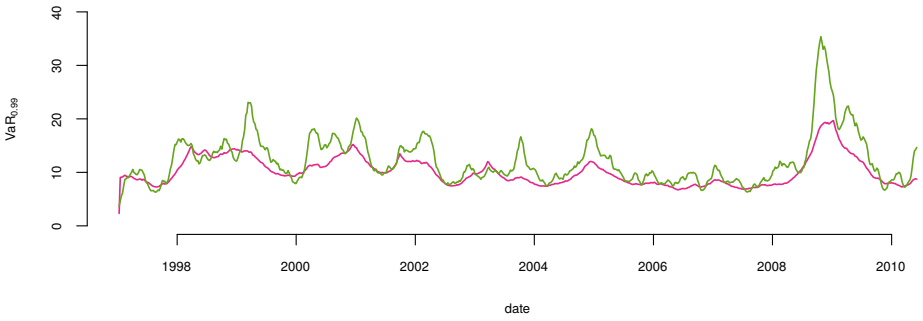


Figure 5. Estimated values of $\mathsf{VaR}_{0.99}(e_t)$ for an equally weighted portfolio of the three oils future contracts using the GSV (magenta) and αSV (green) model with the Student's t copula.

type of oil. The VAR of an asset at confidence $\alpha \in (0, 1)$ is defined by

$$\mathsf{VaR}_\alpha(e_t) = \inf\left\{ -e_t \in \mathbb{R} : G(-e_t) \geq \alpha \right\},$$

i.e., the smallest loss $-e_t$ such that probability that the loss (the negative log-return) exceeds $-e_t$ is no larger then $(1 - \alpha)$. We adopt a Monte Carlo approach to estimate $\mathsf{VaR}_\alpha(e_t)$ by: (i) simulating from the copula by the inverse of (27), (ii) making use of the inverse of the ECDF in (25) to obtain simulated filtered residuals, (iii) computing the resulting log-returns by the estimated volatility and the inverse of (24). We repeat this procedure 100,000 times for each type of oil and then compute the empirical quantile corresponding to $\mathsf{VaR}_{0.99}(e_t)$ for an equally weighted portfolio. Clearly, we can make use of more advanced portfolio weighting schemes in this approach as well. Note that we can make use of the true CDF for the GSV and do not require computing the ECDF in this case.

In Figure 5, we present the resulting estimates of $\mathsf{VaR}_{0.99}(e_t)$ resulting from the simulation for the Student's t copula with the GSV and αSV models for the margins, respectively. We note that the estimates in the two cases are quite different especially at the end of the data series. It is difficult to determine, which value of VAR is the more accurate and we leave it to the reader to decide on which model to use. However, we conclude that this application serves as an illustration that the proposed method can be useful in solving a practical problem.

Conclusions

The illustrations provided in Section 6 indicate that GPO-ABC is quite accurate and exhibit a substantial decrease in the computational cost. Specifically, compared with QPMH2 and SPSA we obtain similar posterior estimates requiring up to 60 and 3 times less computational time, respectively. GPO-ABC is also more robust to noise in the log-posterior estimates, which typically results in that the QPMH2 algorithm gets stuck at times and that the SPSA algorithm converges slowly. We also note that GPO-ABC provides estimates of the parameter uncertainty encoded in the Laplace approximation, which cannot be directly extracted from SPSA. Furthermore, we remind the reader that GPO can be applied for other inference problems, where the log-posterior estimates are noisy and computationally costly to evaluate. Another possible application is to initialise and construct proposals in PMH, which can limit the number of tedious pilot runs that are usually required when p is large.

Future work includes adopting a sparse presentation of the GP and make use of covariance functions tailored to the specific application. Some ideas for sequential and sparse representations of GPs are discussed by Csató and Opper (2002); Bijl et al. (2015); Solin and Särkkä (2014); Deisenroth and Ng (2015) and Huber (2014). An interesting approach for tailored GP priors would be to make use of non-stationary covariance functions (Paciorek and Schervish, 2004). This would capture the fact that the log-posterior often falls off rapidly in some parts of the parameter space and is almost flat in other parts. More computationally efficient alternatives to SMC-ABC are also an important area of future work. Some interesting ideas in this area are discussed in Martin et al. (2014).

Acknowledgements

The simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at Linköping University, Sweden. JD would like to thank Fredrik Lindsten, Neil Lawrence and Carl-Henrik Ek for fruitful discussions. Thanks also to Joerg M. Gablonsky, Abraham Lee, Per A. Brodtkorb and the GPy for making their Python implementations available.

Appendix

Implementation details

GPO algorithm: We make use of the the GPy package (The GPy authors, 2014) for calculating the GP predictive posterior and the GP prior hyperparameters using EB. The covariance presented in (18) is used together with a zero mean function as the GP prior for the log-posterior estimates. We make use of EI as the acquisition rule in (21) with $\zeta = 0.01$ and $\Sigma = 0.01\mathbf{I}_p$. We run the GPO algorithm for $K = 250$ iterations after the initialisation and re-estimate the hyperparameters of the GP prior every 50th iteration.

We initialise the GPO algorithm using $L = 50$ samples obtained using Latin hypercube sampling with the implementation written by Abraham Lee available at `https://pypi.python.org/pypi/pyDOE`. The optimisation problems in Lines 8 and 11 in Algorithm 2 are solved using the DIRECT implementation written by Joerg M. Gablonsky, available from `https://pypi.python.org/pypi/DIRECT/`. Finally for Line 12, we make use of the Python implementation by Per A. Brodtkorb available at `https://pypi.python.org/pypi/Numdifftools`.

Section 6.1: We use $N = 1,000$ particles in GPO-SMC and $N = 2,000$ particles with the Gaussian density with tolerance level $\epsilon = 0.20$ and $\psi(x) = x$ in GPO-ABC to produce the results in Figure 1. The search space for the GPO algorithm Θ_{GPO} is given by $\mu \in (-1, 1)$, $\phi \in (0, 1)$ and $\sigma_v \in (0, 1)$. We use the following prior densities

$$\begin{aligned} p(\mu) &\sim \mathcal{TN}_{(0,1)}(\mu; 0, 0.2^2), \\ p(\phi) &\sim \mathcal{TN}_{(-1,1)}(\phi; 0.9, 0.05^2), \\ p(\sigma_v) &\sim \mathcal{G}(\sigma_v; 0.2, 0.2), \end{aligned}$$

where $\mathcal{TN}_{(a,b)}(\cdot)$ denotes a truncated Gaussian distribution on $[a, b]$, $\mathcal{G}(a, b)$ denotes the Gamma distribution with mean a/b.

For QPMH2, we make use of the fixed-lag particle smoother with $N = 2,000$ particles, lag $\Delta = 12$, memory length $n_{\text{mem}} = 20$ and initialise the Hessian in $400\mathbf{I}_p$. We initialise QPMH2 in the MAP estimate obtained by GPO-SMC and run it for $M = 30,000$ iterations (discarding the first $5,000$ as burn-in). We use $n_{\text{hyb}} = 2,500$ samples of the Markov chain to construct the posterior estimate used in the hybrid approach. For SPSA, we make use of $N = 1,000$ particles, $a = 0.001$, $c = 0.30$, $A = 35$, $\alpha = 0.602$, $\gamma = 0.101$ and initalise the algorithm in the MAP estimate from GPO-SMC. See Spall (1998) for the details of the algorithm and its implementation.

Section 6.2: The real-world data is computed as $y_t = 100[\log(s_t) - \log(s_{t-1})]$, where s_t denotes the price of a future contract on coffee obtained from `https://www.quandl.com/CHRIS/ICE_KC2`. We use $N = 5,000$ particles with the Gaussian density with tolerance level $\epsilon = 0.10$ and $\psi(x) = \arctan(x)$ in GPO-ABC. The search space for the GPO algorithm Θ_{GPO} is given by $\mu \in (0, 1)$, $\phi \in (0, 1)$, $\sigma_v \in (0, 0.7)$ and $\alpha \in (1, 2)$. We use the same priors as for GSV with the additional $p(\alpha) \sim \mathcal{B}(\alpha/2; 6, 2)$, where $\mathcal{B}(a, b)$ denotes the Beta distribution. For QPMH2, we use the same settings as before but adjust $n_{\text{mem}} = 100$ and initialise the Hessian in $1000\mathbf{I}_p$.

Section 6.3: The real-world data is downloaded from `http://freakonometrics.free.fr/oil.xls` and the same settings for standard SMC as in Section 6.1 and SMC-ABC as in Section 6.2. We make use of uniform priors for both the parameters of both the GPDs to model the tails of the CDF and for the copula parameters. The parameter estimates are obtain as the MAP from a quasi-Newton solver.

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.

H. Bijl, J-W. van Wingerden, T. B. Schön, and M. Verhaegen. Online sparse Gaussian process regression using FITC and PITC approximations. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 703–708, Beijing, China, October 2015.

C. M. Bishop. *Pattern recognition and machine learning*. Springer Verlag, 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.

A. D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12:2879–2904, 2011.

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

A. Charpentier. Prévision avec des copules en finance. Technical report, May 2015. URL `https://hal.archives-ouvertes.fr/hal-01151233`.

Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural computation*, 14(3):641–668, 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.

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015a.

J. Dahlin, F. Lindsten, and T. B. Schön. Quasi-Newton particle Metropolis-Hastings. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, pages 981–986, Beijing, China, October 2015b.

J. Dahlin, M. Villani, and T. B. Schön. Efficient approximate Bayesian inference for models with intractable likelihoods. *Pre-print*, 2015c. arXiv:1506.06975v1.

T. A. Dean and S. S. Singh. Asymptotic behaviour of approximate Bayesian estimators. *Pre-print*, 2011. arXiv:1105.3655v1.

T. A. Dean, S. S. Singh, A. Jasra, and G. W. Peters. Parameter estimation for hidden Markov models with intractable likelihoods. *Scandinavian Journal of Statistics*, 41(4): 970–987, 2014.

M. P. Deisenroth and J. W. Ng. Distributed Gaussian Processes. *Pre-print*, 2015. arXiv:1502.02843v1.

P. Del Moral, A. Doucet, and A. Jasra. An adaptive sequential Monte Carlo method for approximate Bayesian computation. *Statistics and Computing*, 22(5):1009–1020, 2012.

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.

J. Durbin and S. J. Koopman. *Time series analysis by state space methods*. Oxford University Press, 2 edition, 2012.

E. Ehrlich, A. Jasra, and N. Kantas. Gradient free parameter estimation for hidden Markov models with intractable likelihoods. *Methodology and Computing in Applied Probability*, 17(2), 2015.

P. Embrechts, C. Klüppelberg, and T. Mikosch. *Modelling extremal events*. Springer Verlag, 1997.

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*. Springer Verlag, 2004.

R. B. Gramacy and N. G. Polson. Particle learning of Gaussian process models for sequential design and optimization. *Journal of Computational and Graphical Statistics*, 20(1):102–118, 2011.

M. U. Gutmann and J. Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Pre-print*, 2015. arXiv:1501.03291v1.

M. F. Huber. Recursive Gaussian process: On-line regression and learning. *Pattern Recognition Letters*, 45:85–91, 2014.

A. Jasra. Approximate Bayesian computation for a class of time series models. *International Statistical Review*, 83(3):405–435, 2015.

A. Jasra, S. S. Singh, J. S. Martin, and E. McCoy. Filtering via approximate Bayesian computation. *Statistics and Computing*, 22(6):1223–1237, 2012.

H. Joe. Asymptotic efficiency of the two-stage estimation method for copula-based models. *Journal of Multivariate Analysis*, 94(2):401–419, 2005.

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.

I. Khindanova, S. Rachev, and E. Schwartz. Stable modeling of value at risk. *Mathematical and Computer Modelling*, 34(9):1223–1259, 2001.

G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

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 and Data Analysis*, 53(6):2298–2308, 2009.

J-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

G. M. Martin, B. P. M. McCabe, W. Maneesoonthorn, and C. P. Robert. Approximate Bayesian computation in state space models. *Pre-print*, 2014. arXiv:1409.8363v1.

A. J. McNeil, R. Frey, and P. Embrechts. *Quantitative risk management: concepts, techniques, and tools*. Princeton University Press, 2010.

E. Meeds and M. Welling. GPS-ABC: Gaussian process surrogate approximate Bayesian computation. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)*, Quebec City, Canada, July 2014.

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, 2010.

R. B. Nelsen. *An introduction to copulas*. Springer Verlag, 2007.

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

C. J. Paciorek and M. J. Schervish. Nonstationary covariance functions for Gaussian process regression. In *Proceedings of the 2004 Conference on Neural Information Processing Systems (NIPS)*, pages 273–280, Vancouver, Canada, December 2004.

M. Panov and V. Spokoiny. Finite sample Bernstein-von-Mises theorem for semiparametric problems. *Bayesian Analysis*, 10(3):665–710, 2015.

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.

S. T. Rachev, S. V. Stoyanov, A. Biglova, and F. J. Fabozzi. An empirical examination of daily stock return distributions for US stocks. In *Data Analysis and Decision Support*, pages 269–281. Springer Verlag, 2005.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

Y. Saatçi, R. D. Turner, and C. E. Rasmussen. Gaussian process change point models. In *Proceedings of the 2010 Conference on Neural Information Processing Systems (NIPS)*, pages 927–934, Vancouver, Canada, December 2010.

C. Sherlock, A. H. Thiery, G. O. Roberts, and J. S. Rosenthal. On the efficency of pseudo-marginal random walk Metropolis algorithms. *The Annals of Statistics*, 43(1):238–275, 2015.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the 2012 Conference on Neural Information Processing Systems (NIPS)*, Lake Tahoe, USA, December 2012.

A Solin and S. Särkkä. Hilbert space methods for reduced-rank Gaussian process regression. *Pre-print*, 2014. arXiv:1401.5508v1.

J. C. Spall. A stochastic approximation technique for generating maximum likelihood parameter estimates. In *Proceedings of the 6th American Control Conference (ACC)*, pages 1161–1167, Minneapolis, 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.

S. V. Stoyanov, B. Racheva-Iotova, S. T. Rachev, and F. J. Fabozzi. Stochastic models for risk estimation in volatile markets: a survey. *Annals of Operations Research*, 176(1):293–309, 2010.

A. Svensson, J. Dahlin, and T. B. Schön. Marginalizing Gaussian process hyperparameters using sequential Monte Carlo. In *Proceedings of the 6th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Cancun, Mexico, December 2015.

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.

The GPy authors. GPy: A Gaussian process framework in Python. `http://github.com/SheffieldML/GPy`, 2014.

R. S. Tsay. *Analysis of financial time series*. John Wiley & Sons, 2 edition, 2005.

A. W. Van der Vaart. *Asymptotic statistics*. Cambridge University Press, 2000.

E. Vazquez and J. Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and inference*, 140(11):3088–3095, 2010.

R. D. Wilkinson. Approximate Bayesian computation (ABC) gives exact results under the assumption of model error. *Statistical applications in genetics and molecular biology*, 12(2): 129–141, 2013.

S. Yıldırım, S. S. Singh, T. Dean, and A. Jasra. Parameter estimation in hidden Markov models with intractable likelihoods using sequential Monte Carlo. *Journal of Computational and Graphical Statistics*, 24(3):846–865, 2014.

Paper F

Hierarchical Bayesian approaches for robust inference in ARX models

Authors: J. Dahlin, F. Lindsten, T. B. Schön and A. Wills

This paper is published by Elsevier:
http://dx.doi.org/10.3182/20120711-3-BE-2027.00318.

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@liu.se

†Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
fredrik.lindsten@it.uu.se
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.

Keywords

ARX models, Robust estimation, Bayesian methods, Markov chain Monte Carlo.

Data and source code in MATLAB

https://github.com/compops/rjmcmc-sysid2012

Financial support from

The projects *Learning of complex dynamical systems* (Contract number: 637-2014-466), *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) and CADICS, a Linnaeus Center, all funded by the Swedish Research Council.

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 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 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 Metropolis-Hastings (RJMH) algorithm introduced by Green (1995). The inference problem resulting from the use of an ARD prior is in the other hand solvable using standard Markov chain Monte Carlo (MCMC) algorithms.

The use of RJMH 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 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.

Hierarchical Bayesia 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.

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.

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

$$p(\theta^n \,|\, n, \delta) = \mathcal{N}(\theta^n; 0, \delta^{-1} I_{n_a+n_b}), \tag{8}$$

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.

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

$$p(\theta_i \,|\, \delta_i) = \mathcal{N}(\theta_i; 0, \delta_i^{-1}), \tag{11}$$

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(MacKey, 1994; Neal, 1996) for further discussion. and 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 over-parameterised model of order $n_{\max}$.

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.

The RJMH sampler (Green, 1995) is a generalisation of MH, 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 RJMH 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 RJMH sampler using the following sweep[1],

(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 Gibbs,

(II-1) ARX coefficients:

- (a) Draw $\theta^\star \mid z_{s+1:T}, \lambda, \delta_{1:m}, D_T$.

[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.

(b) Draw $\delta^{\star}_{1:m}|\theta^{\star}$.

(II-2) Innovation parameters:

(a) Draw $z^{\star}_{s+1:T}|\theta^{\star}, \lambda, \nu, D_T$.

(b) Draw $\lambda^{\star}|\theta^{\star}, z^{\star}_{s+1:T}, D_T$.

(c) Draw $\nu^{\star}|z^{\star}_{s+1:T}$.

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, which are essentially the same for both samplers.

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.

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 $q(n'|n)$. In this work, we follow Troughton and Godsill (1998) and use a constrained random walk with discretised Laplace increments with scale parameter ℓ, i.e.,

$$q(n'_a|n) \propto \exp(-\ell|n'_a - n_a|), \qquad \text{if } 1 \leq n'_a \leq n_{\max}, \tag{14}$$

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

$$\theta^{n'} \sim p(\theta^{n'}|n', z_{s+1:T}, \lambda, \delta, D_T) = \mathcal{N}(\theta^{n'}; \mu_{\theta^{n'}}, \Sigma_{\theta^{n'}}). \tag{15}$$

The expressions for the mean and the covariance are provided in the subsequent section.

Since the proposed coefficients $\theta^{n'}$ are directly connected to the model order n', we apply an MH accept/reject decision to the pair $\{\theta^{n'}, n'\}$ with the acceptance probability given by

$$\begin{aligned}\rho_{nn'} &\triangleq 1 \wedge \frac{p(n', \theta^{n'}|z_{s+1:T}, \lambda, \delta, D_T)}{p(n, \theta^{n}|z_{s+1:T}, \lambda, \delta, D_T)} \frac{q(n, \theta^{n}|n', \theta^{n'})}{q(n', \theta^{n'}|n, \theta^{n})} \\ &= 1 \wedge \frac{p(n'|z_{s+1:T}, \lambda, \delta, D_T)}{p(n|z_{s+1:T}, \lambda, \delta, D_T)} \frac{q(n|n')}{q(n'|n)},\end{aligned} \tag{16}$$

where $a \wedge b \triangleq \min(a, b)$. Since

$$p(n|z_{s+1:T}, \lambda, \delta, D_T) \propto p(y_{1:T}|n, z_{s+1:T}, \lambda, \delta, u_{1:T})p(n), \tag{17}$$

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.

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 RJMH sampler, but the sampling is completely analogous for the Gibbs 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

$$\Phi^n = \begin{bmatrix} -y_s & \cdots & -y_{s-n_a} & u_s & \cdots & u_{s-n_b+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -y_{T-1} & \cdots & -y_{T-n_a} & u_{T-1} & \cdots & u_{T-n_b} \end{bmatrix}. \tag{19}$$

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 RJMH sampler $s = \max(n_a, n'_a)$ and for the Gibbs sampler $s = n_{\max}$.

Let Δ^{-1} be the covariance matrix for the parameter prior, according to (8) or to (11), i.e.,

$$\Delta^{-1} = \begin{cases} \delta I_{n_a+n_b} & \text{for RJMH,} \\ \mathsf{diag}(\delta_1, \ldots, \delta_m) & \text{for Gibbs.} \end{cases} \tag{20}$$

Since we condition on the latent variables $z_{s+1:T}$ (and the variance 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 stated in (15), with mean and covariance given by

$$\mu_{\theta^n} = \Sigma_{\theta^n} (\Phi^n)^{\top} (\lambda z_{s+1:T} \circ y_{s+1:T}), \tag{21a}$$

$$\Sigma_{\theta^n} = \left((\Phi^n)^{\top} \mathsf{diag}(\lambda z_{s+1}, \ldots, \lambda z_T) \Phi^n + \Delta\right)^{-1}, \tag{21b}$$

respectively. Here, $\circ$ denotes elementwise multiplication.

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

$$p(\delta \,|\, \theta^n, n) = \mathcal{G}(\delta; \alpha_\delta^{\text{post}}, \beta_\delta^{\text{post}}), \tag{22}$$

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

$$p(\delta_i \,|\, \theta_i) = \mathcal{G}(\delta_i; \alpha_{\delta_i}^{\text{post}}, \beta_{\delta_i}^{\text{post}}), \tag{24}$$

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 $i = 1, \ldots, m$.

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)

$$p(y_t \,|\, z_t, \theta^n, n, \lambda, \nu, \varphi_t^n) = \mathcal{N}(y_t, (\varphi_t^n)^\top \theta^n, (\lambda z_t)^{-1}). \tag{26}$$

It follows that the posterior is given by

$$p(z_t \,|\, \theta^n, n, \lambda, \nu, D_T) = \mathcal{G}(z_t; \alpha_z^{\text{post}}, \beta_{z_t}^{\text{post}}), \tag{27}$$

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 $t = s + 1, \ldots, T$.

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).

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).

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 RJMH on real EEG data.

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

Method	Mean	CI
LS	77.51	[77.21 77.81]
RJMH	78.24	[77.95 78.83]
Gibbs	77.73	[77.47 78.06]

Table 1. The average and 95% confidence intervals (CIs) for the model fit (in percent) from experiments with 25,000 random ARX models.

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 RJMH method in comparison with the standard LS technique. Also, the RJMH appear to perform better than the simpler Gibbs method (for this model class). Therefore, we will focus primarily on the former method in the remainder of the numerical illustrations.

In the upper part of Figure 1, the differences in model fit between RJMH and LS for all 25,000 systems are shown. We note that there are no cases with large negative values, indicating that the RJMH 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 RJMH. 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 RJMH sampler (nor for the Gibbs 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

[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.

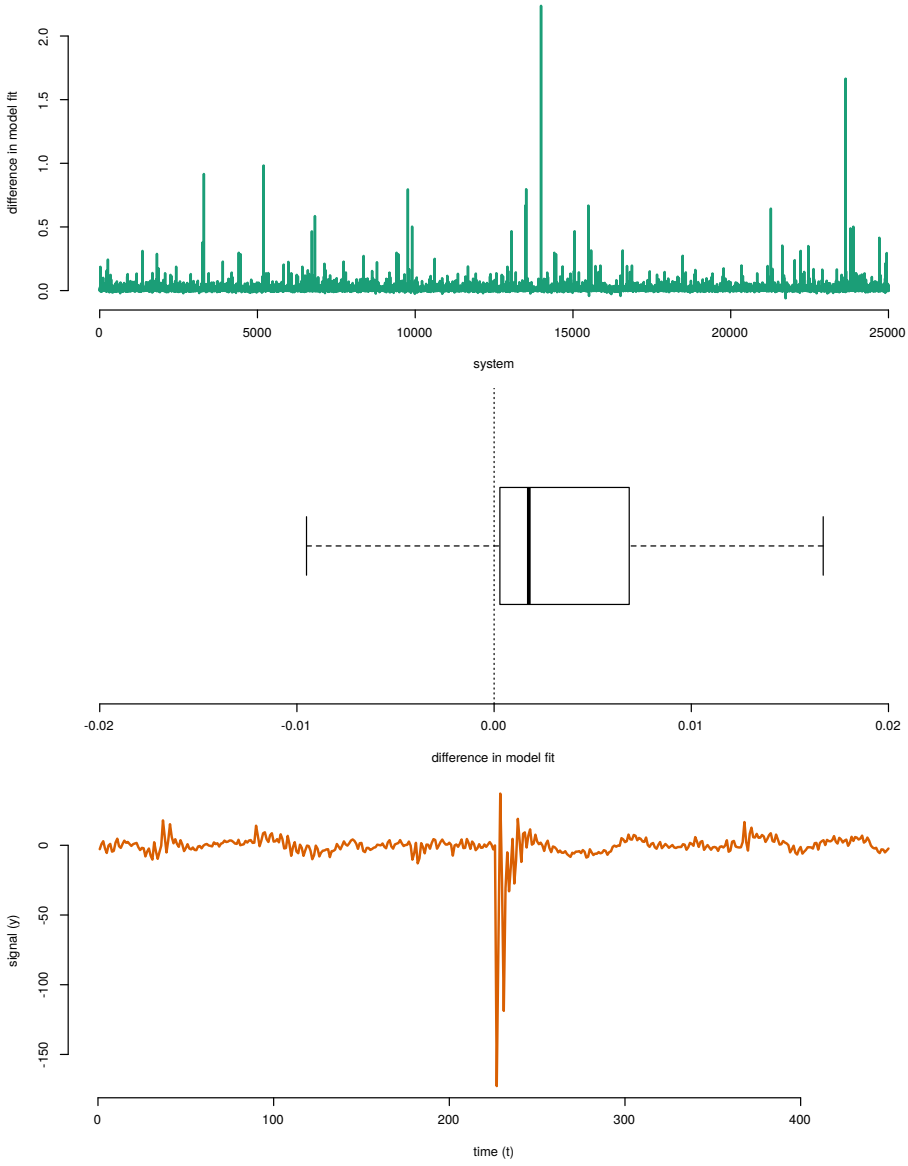


Figure 1. Upper: The difference in model fit between the RJMH and LS methods. Middle: A boxplot of the difference in model fit with the outliers removed. Lower: One particular randomly generated ARX model with a large innovation outlier that affects the system output.

better than the model with Gaussian noise. The model fit for this system was 46.15% for the RJMH 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.

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 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 RJMH method (green stars) and the LS approach (orange 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 RJMH method.

[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.

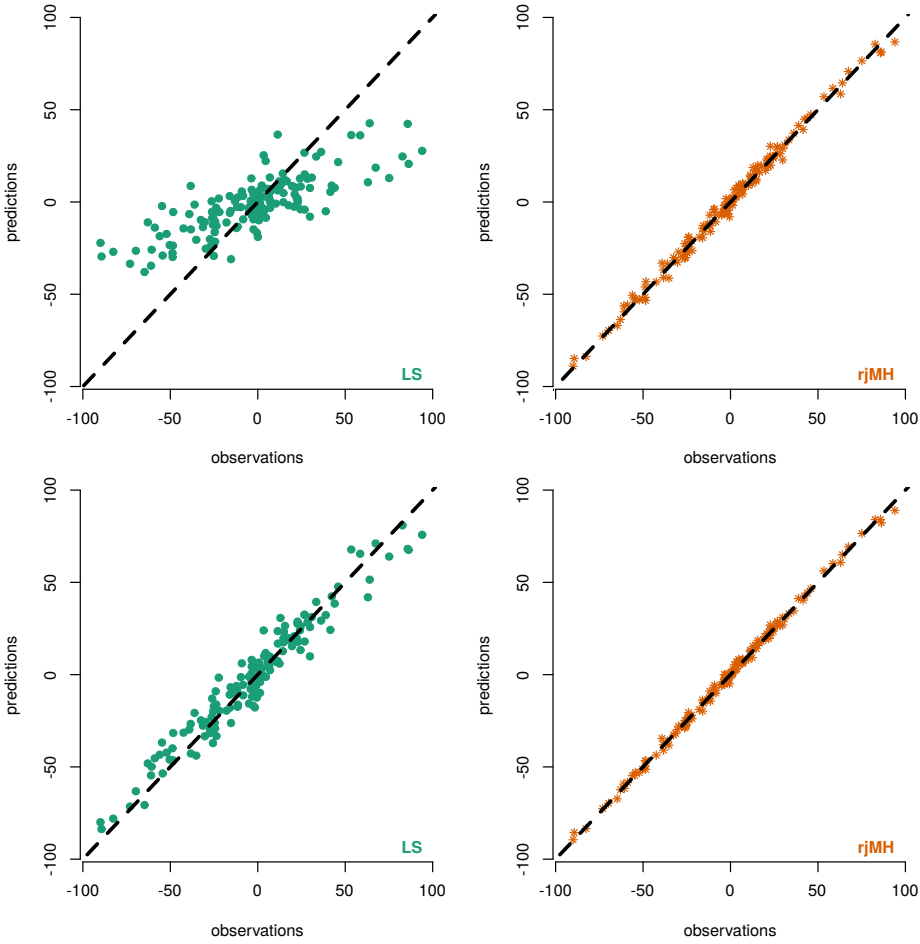


Figure 2. Predictions versus observations for data with outliers (upper) and data with missing observations (lower). The model fit values for the outlier data example are 91.6% for the RJMH (orange stars) and 40.2% for LS (green dots). The corresponding values for the missing data example are 94.4% and 75.7%.

	Outliers		Missing data	
Method	Mean	CI	Mean	CI
LS	39.13	[37.86 40.41]	75.20	[74.00 76.40]
RJMH	70.54	[69.03 72.04]	80.18	[78.74 81.62]
Gibbs	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.

Real EGG 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 distributions by plotting their quantiles against each other (Wilk and Gnanadesikan, 1968).

The RJMH 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.

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 RJMH 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 (Gibbs) or better (RJMH) 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.

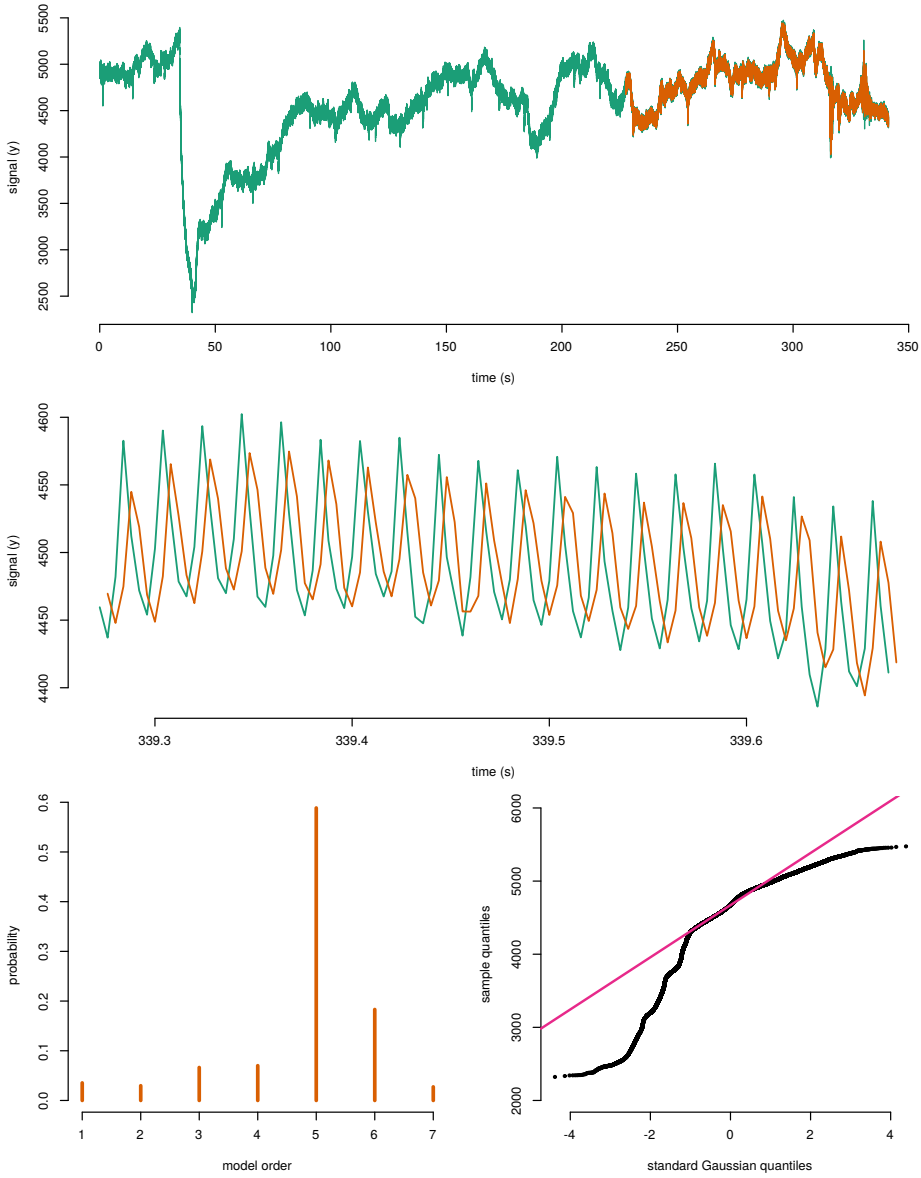


Figure 3. Upper: the EEG signal (green) collected on one specific channel and patient with the one-step-ahead predictions (orange). Middle: the last 100 samples from the upper graph. Lower left: The estimated posterior model order density from the RJMH method. Lower right: The QQ-plot for the data set. The model fit for the results in this figure is 85.6%.

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 non-linear 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 Verlag, 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 Verlag, 1996.

C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Verlag, 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 23rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Seattle, USA, May 1998.

M. B. Wilk and R. Gnanadesikan. Probability plotting methods for the analysis of data. *Biometrika*, 55(1):1–17, March 1968.

Paper G

Bayesian inference for mixed effects models with heterogeneity

Authors: J. Dahlin, R. Kohn and T. B. Schön

Preliminary version:

Technical Report LiTH-ISY-R-3091, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Bayesian inference for mixed effects models with heterogeneity

J. Dahlin*, R. Kohn† and T. B. Schön‡

*Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden.
johan.dahlin@isy.liu.se

†UNSW Business School,
University of New South Wales,
Kensington NSW 2033, Sydney, Australia.
r.kohn@unsw.edu.au

‡Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
thomas.schon@it.uu.se

Abstract

We are interested in Bayesian modelling of panel data using a mixed effects model with heterogeneity in the individual random effects. We compare two different approaches for modelling the heterogeneity using a mixture of Gaussians. In the first model, we assume an infinite mixture model with a Dirichlet process prior, which is a non-parametric Bayesian model. In the second model, we assume an over-parametrised finite mixture model with a sparseness prior. Recent work indicates that the second model can be seen as an approximation of the former. In this paper, we investigate this claim and compare the estimates of the posteriors and the mixing obtained by Gibbs sampling in these two models. The results from using both synthetic and real-world data supports the claim that the estimates of the posterior from both models agree even when the data record is finite.

Keywords

Bayesian inference, mixed effects model, panel/longitudinal data, Dirichlet process mixture, finite mixture, sparseness prior.

Data and source code in R

`https://github.com/compops/panel-dpm2016`

Financial support from

The projects *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) and CADICS, a Linnaeus Center, both funded by the Swedish Research Council.

Introduction

In many fields, we are interested in modelling the dependence of an observation y_{it} of individual i at time t given some regressors/covariates x_{it}. This type of data is known as panel data in economics (Baltagi, 2008) and longitudinal data in statistics (Verbeke and Molenberghs, 2009). That is, when we obtain multiple observations T of a number of N individuals over time. A common application is health surveys in which annual questionnaires are sent out to a group of individuals. The focus is then to isolate different factors that are correlated with disease or visits to the doctor. However, the importance of these factors can vary between different sub-groups in the population and this is referred to as heterogeneity. For prediction purposes, it is therefore important to captures these variations and be able to identify to which sub-group a specific individual belongs.

Another popular application is recommendation systems for online retailing or streaming sites such as Amazon, Netflix and Spotify. The underlying algorithms vary between different sites and are often proprietary information unknown to the public. However, academic work in recommendation system by Condliff et al. (1999) and Ansari et al. (2000) have made used of panel data models. The common theme for both applications are that the number of observations T for each individual is typically much smaller than the size N of the population. It is therefore essential to pool information together from similar individuals to construct a model from data.

For this end, we consider a linear mixed effects model (Verbeke and Lesaffre, 1996) with observation $y_{it} \in \mathbb{R}$ for individual $i = 1, \ldots, N$ at time $t = 1, \ldots, T$. This model can be expressed as

$$y_{it} \,|\, \alpha, \beta_i^s, \sigma_e^2, x_{it} \sim \mathcal{N}\Big(y_{it}, \alpha x_{it} + \beta_i^s z_{it}, \sigma_e^2/\omega_i\Big), \tag{1}$$

where $\alpha \in \mathbb{R}^d$ denotes the fixed effects and $\beta_i^s = \{\beta_{ij}^s\}_{j=1}^p \in \mathbb{R}^p$ denotes the random effects for individual i. Here, x_{it} and z_{it} denote the design matrices connected to the fixed and random effects, respectively. Furthermore, $\sigma_e > 0$ denotes the standard deviation of noise affecting the observations and $\omega_i > 0$ denotes the individual scaling of the variance to allow for variance heterogeneity. We denote a Gaussian distribution with mean μ and standard deviation $\sigma > 0$ by $\mathcal{N}(\mu, \sigma^2)$.

In this paper, we assume that β_i^s can be modelled as an infinite mixture of Gaussians. This means that the random effects can for example be distributed according to a multi-modal distribution. Each mode would then potentially correspond to a certain sub-group of the population with similar behaviour. This information could be important in marketing, economics, medicine and other applications as discussed by Allenby et al. (1998), Canova (2004) and Lopes et al. (2003). A potential benefit of having an infinite mixture is that the data determines the number of components to include in the model.

An infinite mixture of Gaussians can be expressed by

$$\beta_i^s \sim \sum_{k=1}^{\infty} \eta_k \mathcal{N}(\beta_i^s; \beta_k, Q_k), \qquad \text{s.t.} \sum_{k=1}^{\infty} \eta_k = 1, \tag{2}$$

for some weights $\eta_k > 0$, mean vector $\beta_k \in \mathbb{R}^p$ and covariance matrix $Q_k \in \mathbb{R}^{p\times p}$. We proceed to model this mixture in a Bayesian setting in Section 2, where a Dirichlet process (DP; Ferguson, 1973, 1974) prior is employed. This is a Bayesian non-parametric method, which can act as a prior for probability distributions such as (2). However, the inference often relies on Markov chain Monte Carlo (MCMC; Robert and Casella, 2004), which can be challenging to implement and sometimes mixes poorly. The latter is discussed by Hastie et al. (2015) and could be a problem when applying this kind of model for big data.

The main contribution of this paper is to compare two different models for the heterogenity described by (2). In the first approach, we make use of a DP model for (2). In the second approach, we truncate the infinite mixture to obtain an overparametrised finite mixture (FM) for which we make use of a sparseness prior. The latter model is often simpler to implement and can also enjoy better mixing properties. Ishwaran and Zarepour (2002) and Rousseau and Mengersen (2011) have analysed the properties of this approximation. In short, the results are that the approximation is asymptotically consistent (in NT) and converges to the solution obtained when using a DP. The FM is over-parametrised but the sparseness prior empties the superfluous components. Our aim is to compare these two approaches for mixed effects models in terms of the estimate of the posterior and the mixing in the Markov chain constructed by MCMC algorithms.

The comparisons are made on both synthetic and real-world data. The results indicate that the posterior estimates are similar in most cases and that the similarity increases as N and T tend to infinity. The mixing is compared using a Monte Carlo simulation with synthetic data. The resulting mixing seems to be similar for the finite and infinite mixture models. Therefore, there is nothing lost or gained by using either model compared with the other. More work is needed to see if this result holds for even larger sets of data and alternative sampling schemes.

Bayesian mixture modelling

In this section, we discuss the details of the two approaches for modelling the mixture of the random effects (2). The first approach uses the DP prior of the infinite mixture of Gaussians and the second approach approximates the infinite mixture by a FM. We return to the problem of sampling from the posterior of the parameters in the mixed effects model and the mixture for the random effects in Section 3.

Infinite mixture model using a Dirichlet process

In the first approach, we model the random effects β_i^s using the infinite mixture model in (2) with the DP (Ferguson, 1973, 1974) as a prior. The mixture model can be seen as a hierarchical Bayesian model, which we develop step by step in this section.

The DP is an example of a Bayesian non-parametric model, where the number of parameters grow with the number of observations and can be viewed as infinite. A realisation G from a DP is a random discrete probability distribution in the form of an empirical distribution.

Hence, we can express G (Gelman et al., 2013) by

$$G = \sum_{k=1}^{\infty} \eta_k \delta_{\vartheta_k}, \tag{3}$$

where the weights $\{\eta_k\}_{k=1}^{\infty}$ and locations $\{\vartheta_k\}_{k=1}^{\infty}$ are random variables. Here, $\delta_{\vartheta'}$ denotes a Dirac measure placed at ϑ', where ϑ denotes the parameters of the mixture component. Furthermore, we have that $\sum_{k=1}^{\infty} \eta_k = 1$ with probability 1, which means that G can be interpreted as a probability measure.

Let $\mathcal{DP}(\eta_0, G_0)$ denote a DP with the *concentration parameter* $\eta_0 > 0$ and the *base measure* G_0. We say that G is distributed according to a DP if all of its marginal distributions are Dirichlet distributed. Let $\mathcal{D}(\eta)$ denote the Dirichlet distribution with concentration parameter $\eta = \{\eta_1, \ldots, \eta_R\}$. Hence, if G_0 is a probability measure on the space $(\Omega, \mathcal{F})$, we have that

$$\Big(G(A_1), G(A_2), \ldots, G(A_N)\Big) \sim \mathcal{D}\Big(\eta_0 G_0(A_1), \eta_0 G_0(A_2), \ldots, \eta_0 G_0(A_N)\Big), \tag{4}$$

for any finite (measurable) partition $A_{1:N}$ of Ω. Note that the expected value of G is the base measure and therefore G has the same support as G_0. Moreover, G is discrete with probability one even if the base measure is continuous, which is useful in mixture models as discussed below.

Assume that we obtain some data generated from the model given by

$$G \sim \mathcal{DP}(\eta_0, G_0), \qquad \vartheta_i \,|\, G \sim G, \qquad i = 1, 2, \ldots.$$

In many applications, we would like to compute the predictive distribution for some new parameter $\vartheta_\star$ given the observations $\vartheta_{1:N}$. The predictive distribution can be computed as a marginalisation given by

$$p(\vartheta_\star \,|\, \vartheta_{1:N}) = \int G(\vartheta_\star) p(G|\vartheta_{1:N}) \mathrm{d}G,$$

which is possible to carry out in closed-form. The result is a so-called Pólya urn scheme discussed by Blackwell and MacQueen (1973). This scheme can be expressed mathematically

$$\vartheta_\star \,|\, \vartheta_{1:N} \sim \frac{\eta_0}{\eta_0 + N} G_0 + \frac{1}{\eta_0 + N} \sum_{i=1}^{N} n_i \delta_{\vartheta_i}. \tag{5}$$

Here, n_i denotes the number of parameters that are identical to ϑ_i, i.e.,

$$n_i = \sum_{j=1}^{N} \mathbb{I}[\vartheta_j = \vartheta_i],$$

where $\mathbb{I}[A]$ denotes the indicator function which assumes the value one if A is true and zero otherwise.

From (5), we note that the DP is a discrete process with a non-zero probability of ties, i.e., that two or more realisations are identical to each other. From the Pólya urn scheme, we either draw a new parameter from the base measure or an existing parameter from the Dirac mixture. The probability of sampling a new parameter from the base measure is determined

by concentration parameter. We are more likely to sample from the base measure if $\eta_0 \gg N$, which means that the predictive posterior concentrates to the base measure. If the concentration parameter is small, we often sample from the existing parameters, which gives many ties and a strong clustering behaviour.

Hence, we can make use of this clustering effect to reformulate (2) as a Dirichlet process mixture (DPM; Antoniak, 1974) given by

$$G \sim \mathcal{DP}(\alpha, G_0), \qquad \beta, Q \,|\, G \sim G, \qquad \beta_i^s \sim \mathcal{N}(\beta, Q), \tag{6}$$

where we choose $\vartheta = \{\beta, Q\}$ for this particular model. The presence of ties means that several β_i^s will be drawn from a Gaussian distribution with the same parameters. Hence, this is an alternative presentation of (2).

We have now introduced the DPM and discussed the properties of the underlying DP as well as how to compute the predictive posterior distribution. What remains is to discuss how to simulate from a DPM and how to compute the prior-posterior update given some data. In this paper, we make use of the stick-breaking by Sethuraman (1994) to simulate from the mixture. The procedure iterates

$$\beta_k, Q_k \sim G_0, \qquad \eta_k = V_k \prod_{l<k}(1 - V_l), \qquad V_k \sim \mathcal{B}(1, \eta_0), \tag{7}$$

for $k = 1, \ldots, R$, where R is an upper bound on the number of components in the mixture selected by the user. Here, $\mathcal{B}(a, b)$ denotes a Beta distribution with shape $a > 0$ and scale $b > 0$. Moreover, $\prod_{l<k}(1 - V_l)$ denotes the length remaining of a unit stick after breaking off $k - 1$ pieces, where each V_k denotes the fraction of the remaining stick to break off.

Note that the truncation implied by the stick-breaking corresponds a mixture given by

$$\beta_i^s \,|\, \eta_{1:R}, \beta_{1:R}, Q_{1:R} \sim \sum_{r=1}^{R} \eta_r \mathcal{N}(\beta_i^s; \beta_r, Q_r).$$

In many cases, we can choose R to be larger than the number of individuals N, so this is not a problem when n is rather small. To see the connection between the sick-breaking and (5), we note that the expected value of the Beta distribution sampled from in (7) is $(1 + \eta_0)^{-1}$. Hence, the expected part of the stick to break off tends to zero when $\eta_0 \to \infty$ and tends to one when $\eta_0 \to 0$. In the latter case, we usually obtain a few components, which is in agreement with the previous discussion of the concentration parameter. We return to computing the posterior of a DPM in Section 3 using Gibbs sampling.

Finite mixture model

The second approach that we consider is the FM, which can be recovered as a special case of the DPM model (6) given by

$$\beta_i^s \,|\, S_i, \beta_{1:R}, Q_{1:R} \sim \mathcal{N}(\beta_{S_i}, Q_{S_i}), \qquad S_i \,|\, \eta \sim \mathcal{M}(1, \eta_1, \ldots, \eta_R), \tag{8a}$$

$$\beta_k, Q_k \sim G_0, \qquad \eta \sim \mathcal{D}(\eta_{0,1}, \ldots, \eta_{0,R}), \tag{8b}$$

where the latent variable S_i denotes which component that individual i belongs to. Here, $\mathcal{M}(1, p)$ denotes the multinomial distribution with one try and event probabilities given by

p. The parameters of the mixture components are still realisations from the base measure. However, the DP is now a Dirichlet distribution, which follows from the property of the DP discussed in connection with (4). The Dirichlet distribution determines the probability of a certain cluster membership S_i. Hence, we get a similar clustering effect as for the DPM if we select η_0 to be small. This is essentially the same result as we had for the Pólya urn scheme in (5).

An interesting property discussed by Ishwaran and Zarepour (2002) and Rousseau and Mengersen (2011) is that the FM can approximate the DPM. This means that the sparsity will empty unnecessary components and that the estimate of the posterior tend to the true one. However, we need to be careful when setting the hyperparameters in the Dirichlet prior distribution for this approximation to work. On choice advocated by Ishwaran and Zarepour (2002) is to use the concentration parameter $\eta_{0,1} = \ldots = \eta_{0,R} = \eta_0/R$ with $\eta_0 = 1$. Here, the number of components R can be selected as n if the amount of data is small. Here, the small value of η_0 results in that a few samples from the Dirichlet distribution are allocated most of the probability mass. Therefore, we get a sparsity effect as only a few components are occupied a priori when $\eta_0 \leq 1$.

We discuss how to sample from the posterior of the FM (8) in Section 3

Sampling from the posterior

In this section, we make use of MCMC sampling to estimate the posterior of the parameters in the mixed effects model (1) and the mixture modelling the random effects (2) given the data $\{y, x, z\} = \{\{y_{it}, x_{it}, z_{it}\}_{i=1}^{n}\}_{t=1}^{T}\}$. The parameter vector is given by $\theta = \{\alpha, \beta_{1:R}, Q_{1:R}, \beta^s_{1:N}, \omega_{1:N}, \sigma^2_e, S_{1:N}, \eta_{1:R}\}$, which includes the parameters of the DPM (6) and the FM (8). To decrease the notational burden, we have used the same notation for the parameters in both mixture models.

We make use of conjugate priors to obtain closed-form conditional posteriors, which allows for Gibbs sampling as discussed by Gelman et al. (2013), Frühwirth-Schnatter (2006) and Neal (2000). Note that there are many other interesting alternatives to Gibbs sampling for estimating the posterior of a DPM. Sequential Monte Carlo algorithms (Del Moral et al., 2006) discussed by Fearnhead (2004) and Ulker et al. (2010) do not have problems with mixing but can be challenging to implement.

Slice samplers are also an interesting alternative discussed by Walker (2007), which are easy to implement and give moderate or good mixing. Finally, split-merge algorithms can give good mixing as discussed by Jain and Neal (2004) and Bouchard-Côté et al. (2015) but can be challenging to implement. For the FM, a simple Gibbs sampling approach often gives good mixing and is easy to implement.

Prior distributions

To compute the posterior, we need to assign prior distributions for each of the elements in θ. Here, we make use of the prior distributions from Frühwirth-Schnatter (2006, p. 265)

for all the parameters except for the mixture weights $\eta_{1:R}$. For the fixed effects and the noise variance with heterogeneity, we choose

$$\alpha \sim \mathcal{N}(\alpha; a_0, A_0), \qquad \sigma_e^2 \sim \mathcal{G}^{-1}(\sigma_e^2; c_0^e, C_0^e), \qquad \omega_i \sim \mathcal{G}\Big(\frac{\nu}{2}, \frac{\nu}{2}\Big), \tag{9}$$

for individuals $i = 1, \ldots, n$. Here, we introduce the notation $\mathcal{G}(a, b)$ for the Gamma distribution with shape $a > 0$ and rate $b > 0$, which means that the expected value is ab^{-1}. The inverse Gamma distribution is denoted by $\mathcal{G}^{-1}(a, b)$ with the expected value $b(a-1)^{-1}$. Hence, we have the hyperparameters $\{a_0, A_0, c_0^e, C_0^e, \nu\}$ for the user to choose.

We also need to choose a number of priors for the mixture model (2) describing the distribution of the random effects. Here, we make use of the priors

$$\beta_k \sim \mathcal{N}(\beta_k; b_0, B_0), \qquad Q_k^{-1} \sim \mathcal{W}(Q_k^{-1}; c_0^Q, C_0^Q), \tag{10}$$

where $\mathcal{W}(n, V)$ denotes the Wishart distribution with $n > 0$ degrees of freedom and scale matrix $V > 0$, respectively. Hence, we have $\{b_0, B_0, c_0^Q, C_0^Q\}$ as additional hyperparameters for the user to choose.

Furthermore, we assign a prior for the concentration parameter in the DPM given by

$$\eta_0 \sim \mathcal{G}(c_0^\eta, C_0^\eta), \tag{11}$$

for some hyperparameters c_0^η and C_0^η.

Gibbs sampling

To sample from the posterior, we make use of blocked Gibbs sampling algorithms. For the DPM model, we make use of the algorithm proposed by Gelman et al. (2013, p. 552), which is a truncated approximation using at most R clusters. The stick-breaking procedure is used to sample from the DPM in this formulation.

For the FM, we make use of the Gibbs sampler proposed by Frühwirth-Schnatter (2006, p. 368), which samples from the mixture using the conjugacy between the Drichlet prior distribution and the multinomial data. The remaining steps of the Gibbs sampler are identical and the full procedure performed during one iteration of the sampler is presented in Algorithm 1. Note that the differences between the two alternatives for modelling appears in Steps 1(i) and 5.

For the FM, we sample the mixture weights from the posterior given by the conjugacy property as previously discussed. This results in the sampling scheme

$$\eta'_{1:R} \sim \mathcal{D}(\eta_0/R + n_1, \eta_0/R + n_2, \ldots, \eta_0/R + n_R), \tag{12}$$

where n_r denotes the number of individuals in component r. The details are discussed by Gelman et al. (2013, p. 534). For the DPM model, we make use of the stick-breaking construction in (7) to generate the weights $\eta'_{1:R}$ for the truncated model, i.e.,

$$\eta'_r = V_r \prod_{l<r}(1 - V_l), \qquad V_r \sim \mathcal{B}\left(1 + n_r, \eta_0 + N - \sum_{j=1}^{r} n_j\right), \tag{13}$$

Algorithm 1 Gibbs sampling for mixture models

INPUTS: $\{\{y_{it}, x_{it}, z_{it}\}_{i=1}^{N}\}_{t=1}^{T}$ (data), θ_0 (hyperparameters).
OUTPUTS: θ' (approximate sample from the parameter posterior).

During one iteration of the Gibbs sampler:

1: Update parameters given the cluster allocation.
(i: FM) Sample $\eta_r' | S_{1:N}$ using (12) for $r = 1, 2, \ldots, R$.
(i: DPM) Sample $\eta_r' | S_{1:N}$ using (13) for $r = 1, 2, \ldots, R$.
(ii) Sample $\alpha^{\star,\prime} | y, x, z, Q_{1:R}, \omega_{1:N}, S_{1:N}$ using (14) with $\alpha^\star = \{\alpha, \beta_{1:R}\}$.
(iii) Sample $Q_r' | \beta_{1:N}^s, \beta_{1:R}', S_{1:N}$ using (15) for $r = 1, 2, \ldots, R$.
(iv) Sample $\sigma_e^{2,\prime} | y, x, z, \alpha', \beta_{1:N}^s, \omega_{1:N}$ using (16).
2: Sample allocations $S_i' | y, x, z, \alpha', \beta_i^{s,\prime}, \beta_{1:R}', Q_{1:R}', \omega_{1:N}$ using (17) for $i = 1, 2, \ldots, N$.
3: Sample the random effects $\beta_i^{s,\prime} | y_i, x_i, z_i, \alpha', \beta_{S_i'}', Q_{S_i'}', \omega_i$ using (18) for $i = 1, 2, \ldots, N$.
4: Sample the variance heterogeneity $\omega_i' | y, x, z, \alpha', \beta_i^{s,\prime}, \sigma_e^{2,\prime}$ using (19) for $i = 1, 2, \ldots, N$.
5: [DPM] Sample the concentration parameter η_0' using (20).

for $r = 1, \ldots, R$ and starting with a stick of unit length. The remaining parameters are sampled from the conditional posteriors derived in the subsequent section.

Conditional posteriors

In this section, we outline the details for sampling from each of the conditional posteriors in Algorithm 1. In Step 1(ii), we are required to sample from the conditional of $\alpha^\star \triangleq \{\alpha, \beta_1, \ldots, \beta_R\}$, which are the fixed effects and the mean of each component in the mixture (2). This essentially requires us to solve a regression problem where the regressors are given by $z_i = (x_i, z_i D_{i1}, \ldots, z_i D_{iK})$ with $D_{ik} = 1$ if $S_i = k$ and zero otherwise. Hence, we rewrite the regression model to obtain

$$y_i = z_i \alpha^\star + \widetilde{e}_i,$$

where $\widetilde{e}_i \sim \mathcal{N}(0, V_i)$ with $V_i = z_i Q_{S_i}(z_i)^\top + \sigma_e^2 / \omega_i \mathbf{I}_T$. We can therefore compute the posterior using a standard Bayesian linear regression with known covariance and Gaussian prior for the regression coefficients. The conditional posterior is given by

$$\alpha^\star | y_{1:N}, Q_{1:R}, \sigma_e^2, \omega_{1:N}, S_{1:N} \sim \mathcal{N}_{d+Kp}(\alpha^\star; a_N^\star, A_N^\star), \tag{14}$$

where the mean and the covariance are given by

$$(A_N^\star)^{-1} = \sum_{i=1}^{N} (z_i^\star)^\top V_i^{-1} z_i^\star + (A_0^\star)^{-1}, \qquad a_N^\star = A_N^\star \left(\sum_{i=1}^{N} (z_i^\star)^\top V_i^{-1} y_i + (A_0^\star)^{-1} a_0^\star \right).$$

In Step 1(iii), we sample the covariance of the mixture components. The posterior is given by the conjugacy of the Wishart prior for the covariance matrix and a Gaussian likelihood.

We can compute the conditional posterior for $r = 1, \ldots, R$ by

$$Q_r^{-1} \,|\, \beta_{1:R}, \beta^s_{1:N}, S_{1:N} \sim \mathcal{W}\left(Q_r^{-1}; c_r^Q, C_r^Q\right), \tag{15}$$

where the sufficient statistics (assuming independence between B_0 and C_r^Q) are given by

$$c_r^Q = c_0^Q + \frac{n_r}{2}, \qquad C_r^Q = C_0^Q + \frac{1}{2} \sum_{\{i:S_i=r\}} (\beta_i^s - \beta_r)(\beta_i^s - \beta_r)^\top,$$

where again n_r denotes the number of individuals in component r, i.e.,

$$n_r = \sum_{i=1}^{N} \mathbb{I}(S_i = r).$$

In Step 1(iv), we sample the variance of the noise σ_e^2, which we assume to have an inverse-Gamma prior distribution. As the likelihood is Gaussian, we obtain the conjugate posterior given by

$$\sigma_e^2 \,|\, y_{1:N}, \alpha, \beta^s_{1:N}, \omega_{1:N} \sim \mathcal{G}^{-1}\left(\sigma_e^2; c_k^e, C_k^e\right), \tag{16}$$

where the sufficient statistics are given by

$$c_k^e = c_0^e + \frac{NT}{2}, \qquad C_k^e = C_0^e + \frac{1}{2} \sum_{i=1}^{N} \omega_i \big(y_i - \alpha x_i - \beta_i^s z_i\big)^2.$$

Note that the data in this update is given by the scaled residuals for each of the individuals.

In Step 2, we update the allocation of each individual to a component by sampling S_i for $i = 1, \ldots, N$. The latent variable S_i is sampled from a multinomial distribution, where the probabilities reflect the likelihood that individual i belongs to a certain component. That is, we sample

$$S_i \,|\, \alpha, \beta_{1:R}, Q_{1:R}, \omega_i, \sigma_e^2, y_i \sim \mathcal{M}(1, p_{1:R}), \tag{17}$$

where the probabilities are given by

$$p_r = \frac{\eta_r \mathcal{N}\big(y_i; \alpha x_i + \beta_k z_i, W_{ik}\big)}{\sum_{l=1}^{R} \eta_l \mathcal{N}\big(y_i; \alpha x_i + \beta_l z_i, W_{il}\big)}, \qquad W_{il} = z_i Q_l (z_i)^\top + \frac{\sigma_e^2}{\omega_i} \mathbf{I}_T.$$

Note that the probabilities follow from the mixture model (2) directly. The intuition is that we are more likely to select the component that best explains the data in terms of its contribution to the likelihood.

In Step 3, we sample the random effects for each individual from the component to which the individual is assigned. The prior for each individual is given by $p(\beta_i) = \mathcal{N}(\beta_{S_i'}, Q_{S_i'})$ and the likelihood is Gaussian and given by

$$y_i - \alpha x_i = z_i \beta_i^s + \frac{\sigma_e}{\sqrt{w_i}} e_i,$$

where e_i is a standard Gaussian random variable. Note that this again is a Bayesian linear regression with a Gaussian prior for the regression coefficient β_i^s, where the observations

are given by $y_i - \alpha x_i$ and the regressors are x_i. In this case, the variance is known and the conditional posterior for $i = 1, \ldots, N$ is

$$\beta_i^s \,|\, y_i, \alpha, \beta_{S_i}, Q_{S_i}, \omega_i \sim \mathcal{N}(\beta_i^s; b_i^s, B_i^s), \tag{18}$$

where the sufficient statistics are given by

$$B_i^s = \left(Q_{S_i}^{-1} + (z_i)^\top z_i \frac{\omega_i}{\sigma_e^2}\right)^{-1}, \qquad b_i^s = B_i^s \left(Q_{S_i}^{-1}\beta_{S_i} + (z_i)^\top (y_i - \alpha x_i)\frac{\omega_i}{\sigma_e^2}\right),$$

In Step 4, we sample the individual scaling of the noise of the observations. The derivation is similar to for Step 1(iv), but here we have a Gamma distributed prior for ω_i and a Gaussian likelihood. The conditional posterior is given by

$$\omega_i \,|\, y_{1:N}, \alpha, \beta_{1:N}^s, \sigma_e^2 \sim \mathcal{G}\left(\omega_i; c_k^\omega, C_k^\omega\right), \tag{19}$$

where the sufficient statistics are given by

$$c_k^\omega = \frac{\nu}{2} + \frac{T}{2}, \qquad C_k^\omega = \frac{\nu}{2} + \frac{1}{2\sigma_e^2}(y_i - \alpha x_i - \beta_i^s z_i)^2.$$

In Step 5, we sample the concentration parameter η_0 for the DPM model using the Gamma prior in (11). In this case, the conditional posterior (Gelman et al., 2013, p. 553) is given by

$$\eta_0' \sim \mathcal{G}\left(c_0^\eta + N - 1, C_0^\eta - \sum_{r=1}^{R-1} \log(1 - V_r)\right), \tag{20}$$

where V_r is the fraction broken off during the stick-breaking in Step 1(i) of the procedure.

Numerical illustrations

In this section we present three numerical experiments to compare the difference of modelling heterogeneity in the random effects using the FM and DPM model. The aim is to compare the posterior estimates and the mixing in the Markov chain created by the Gibbs sampler. The latter is important as it determines the inefficiency of the sampling and the asymptotic variance of the posterior estimates.

Mixture of Gaussians

We begin with a simple mixture of Gaussians model (Escobar and West, 1995) to validate our implementation and present the setup that we use for the comparisons in this section. In this model, we simplify the mixed effects model (1) such that $y_i = \beta_i^s$, where the heterogeneity of the random effects is modelled by (2). We generate $T = 1$ observations for $N = 100$ individuals using $R = 3$ components and $\eta_{1:3} = \{0.3, 0.5, 0.2\}$, $\beta_{1:3} = \{-1, 0, 2\}$ and $Q_{1:3} = \{0.2^2, 1, 0.05^2\}$. We carry out the inference using Algorithm 1 with the settings presented in Appendix A.

The results are presented in Figure 1. In the upper part, we present the estimates of the posterior of β_i^s using three different models: (i) a FM with the true number of components

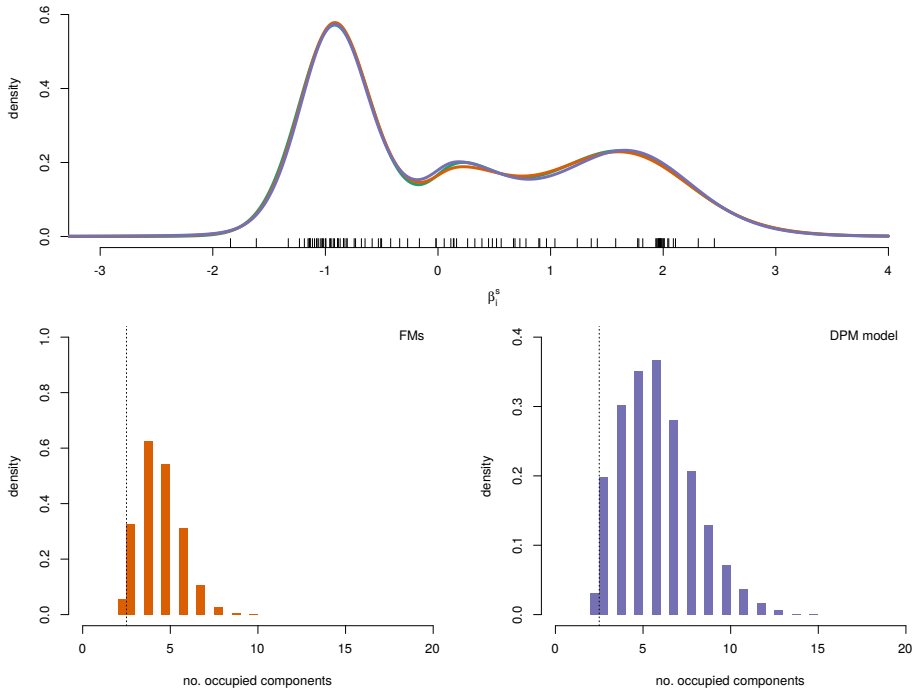


Figure 1. Upper: the posterior estimate from a finite mixture with 3 components (green), the FM (orange) and DPM model (purple). The *rug* indicate the observed data. Lower: the number of active clusters, where vertical dotted lines indicate the true number of clusters.

$R = 3$, (ii) a FM with $R = 20$ components and (iii) a DPM model. We note that the three models give almost the same estimate of the posterior, which indicates that the sparsity effect of the FM works satisfactory and empties the extra components. This can be seen in the lower part of the same figure as less than 20 components are active at any time in the FM. Furthermore, the DPM model use on average more components than the FM. However, both models often make use of more components than in the model from which we generated the data.

Mixed effects model with synthetic data

We now consider the complete mixed effects model (1) with heterogenity in the individual random effects (2). We simulate 40 independent data sets using $T = 100$ observations in each while varying the number of individuals N in $\{10, 20, 50, 100, 200, 500\}$. We make use of a mixture for the random effects with $R = 3$ components and parameters

$$\eta_{1:3} = \{0.4, 0.3, 0.3\}, \qquad \beta_{1:3} = \{2, 3, -2\}, \qquad Q_{1:3} = \{1, 0.2, 0.2\}.$$

Moreover, we make use of $\sigma_e^2 = 1$, $\alpha = 2$ and $\omega_{1:N} \sim |\mathcal{N}(0, 1)|$ for the remaining parameters of the model. We carry out the inference using Algorithm 1 with the settings presented in Appendix A.

We proceed by comparing the difference in the posterior estimates of the random effects. This is done by computing the mean squared error (MSE) between the kernel density estimates (KDEs) of the sought posterior using the FM and DPM model. Hence for each data set, we compute the KDEs of the posterior estimates and compute the squared distance between them. In Table 1, we present the results which indicate that the MSE (after an initial increase) tends to decrease when N grows. We therefore conclude that the posterior estimates tend to become similar as the amount of data increases. Furthermore, we present some graphical comparisons in Figure 2 for four Monte Carlo runs. We see that the posterior estimates are similar most of the time, which is promising and validates the conclusion from the MSE comparison.

We compare the mixing in the two models using the integrated autocorrelation time (IACT) also known as the inefficiency factor for the Gibbs sampler applied to the two models. The IACT is estimated by

$$\widehat{\mathsf{IACT}}\big(\varphi(\theta_{K_b:K})\big) = 1 + 2\sum_{l=1}^{L} \widehat{\rho}_l\big(\varphi(\theta_{K_b:K})\big), \tag{21}$$

where K_b denotes the number of iterations in the *burn-in* and K denotes the number of iterations of the Gibbs sampler. Here $\widehat{\rho}_l(\varphi(\theta_{K_b:K}))$ denotes the empirical autocorrelation at lag l of some test function φ that depends on θ. Here, we make use of the log-likelihood and the number of occupied components to compute the IACT. The log-likelihood for the mixed effects model (1) is given by

$$\psi_1(\theta) = \sum_{i=1}^{N}\sum_{t=1}^{T} \log \mathcal{N}\left(y_{it}; \alpha x_{it} - \beta_i^s z_{it}, \frac{\sigma_e^2}{\omega_i}\right).$$

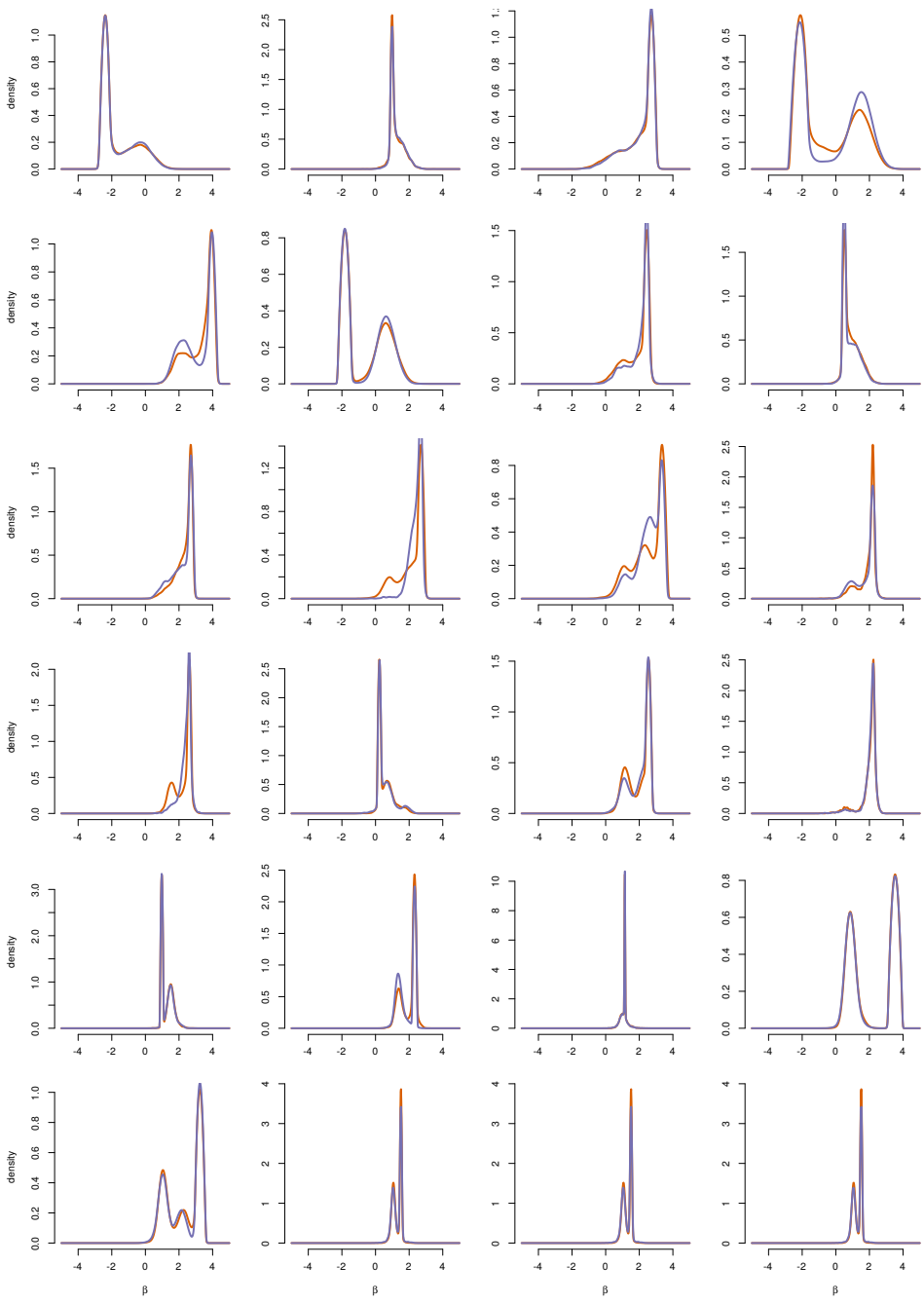


Figure 2. Estimates of $p(\beta_i^s|y)$ from a FM (orange) and DPM model (purple) for $n = \{10, 20, 50, 100, 200, 500\}$ (rows) and 4 independent Monte Carlo runs (columns).

		$N10$	$N = 20$	$N = 50$	$N = 100$	$N = 200$	$N = 500$
Log-MSE		-6.1	-5.5	-5.1	-3.7	-4.1	-4.9
$\widehat{\text{IACT}}(\psi_1)$	FM	16 (27)	22 (48)	36 (75)	19 (61)	2 (5)	1 (1)
	DPM	11 (25)	24 (34)	29 (45)	10 (46)	2 (4)	1 (1)
$\widehat{\text{IACT}}(\psi_2)$	FM	4 (1)	7 (3)	8 (4)	8 (3)	7 (1)	11 (5)
	DPM	12 (10)	12 (10)	13 (9)	11 (10)	8 (5)	8 (4)

Table 1. The logarithm of the MSE of the estimates and the IACT of the log-likelihood and the number of occupied components obtained in the FM and in a DPM model. The IACT values are the median with the interquartile range in parenthesise from 40 Monte Carlo simulations.

The number of occupied clusters is given by

$$\psi_2(\theta) = \sum_{k=1}^{R} \mathbb{I}(n_r > 0),$$

where n_r denotes the number of individuals in component r. A small value of the IACT for both quantities indicates that we obtain many uncorrelated samples from the sought posterior. This implies that the chain is mixing well and that the asymptotic variance of the parameter estimates is rather small. We choose $L = \max\{3, \lfloor 0.5(M - M_b)\rfloor\}$ and make use of the `IAT` command in the R-package `LaplacesDemon` (Hall, 2012) to compute the estimate of the IACT.

Table 1 also presents the median IACT with the interquartile range (the length between the first and third quartiles) from 40 Monte Carlo runs over different data sets. We compare the IACT for the FM and the DPM model for both the log-likelihood and the number of occupied components. We note that there are no significant differences between the two models. Also, the IACT in the log-likelihood seems to decrease with increasing N and the opposite holds for the mixing in the number of occupied components.

Mixed effects model with sleep deprivation data

Finally, we consider the data set presented by Belenky et al. (2003) of the reaction times in a sleep deprivation trial of $N = 18$ individuals. We present the data in Figure 3 from the test during $T = 10$ consecutive days during which the subjects are only allowed to sleep three hours every night. Moreover, we present the linear regression for each individual indicated by the green lines. It seems that there are two different types of individuals in the data. In the first sub-group, the reaction time significantly increases for each day of sleep deprivation, e.g., individuals 337, 308 and 350. In the second sub-group, the lack of sleep does not seem to have a large impact on the reaction times, e.g., individual 351, 335 and 309. Hence, we can assume that the distribution of the random effects is possibly multi-modal, where the modes represent these sub-groups.

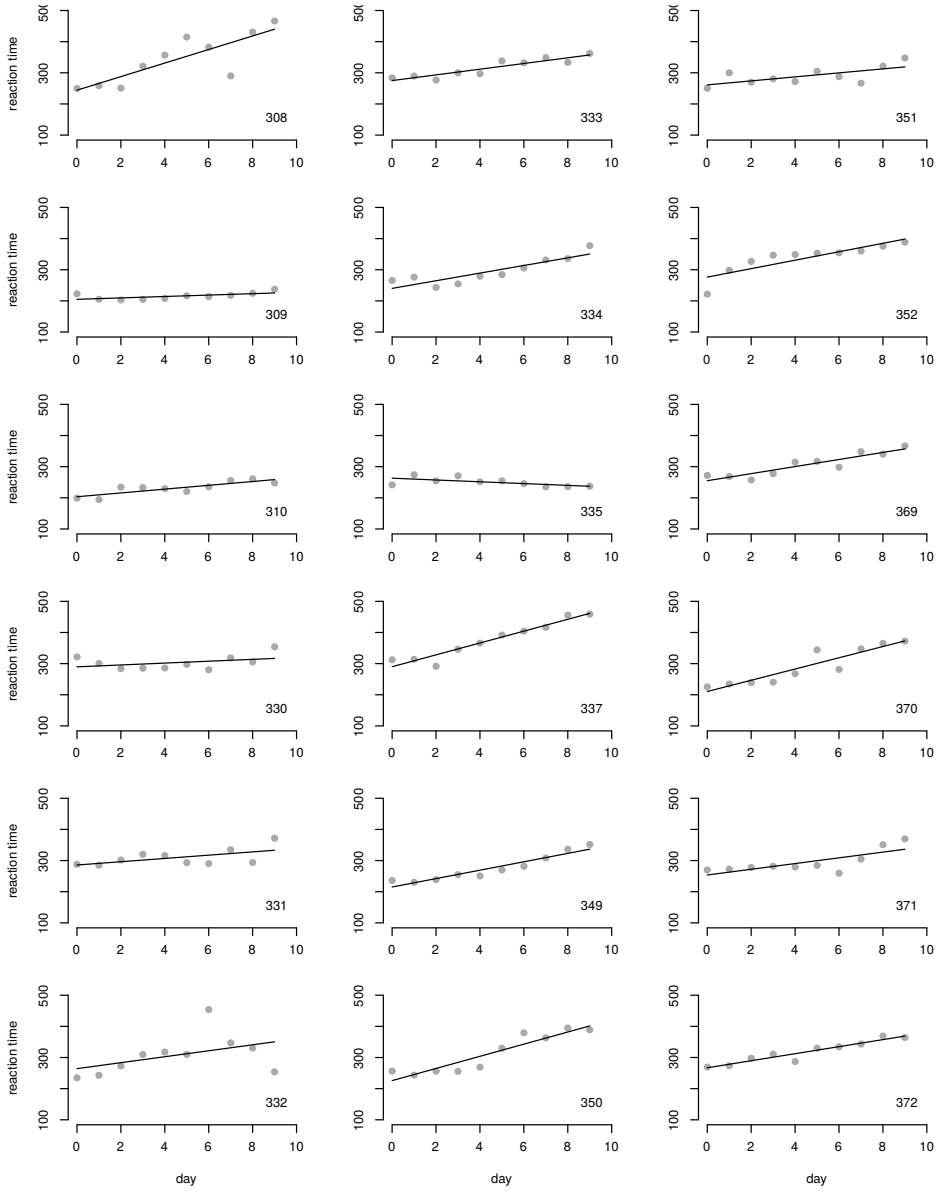


Figure 3. The reaction time in milliseconds for $N = 18$ individuals during $T = 10$ consecutive days with only three hours of sleep per night. The solid lines indicate the best linear fit for each individual and the dots indicate the data points.

We make use of the model discussed by Bates et al. (2015) to try to capture this effect. Let $\{\{y_{it}\}_{i=1}^{N}\}_{t=1}^{T}$ denote the reaction time in millisecond for individual i on day t. We assume that the observations can be modelled by a mixed effects model given by

$$y_{it} \,|\, \alpha, \beta_i^s, \sigma_e^2 \sim \mathcal{N}\left(y_{it}; (\alpha_0 + \beta_{i0}^s) + (\alpha_1 + \beta_{i1}^s)(t-1), \frac{\sigma_e^2}{\omega_i}\right). \tag{22}$$

Furthermore, we assume that the individual random effects can be modelled using a mixture of Gaussians (2). The model parameters are $\theta = \{\alpha_0, \alpha_1, \beta_{1:R,0}, \beta_{1:R,1}, \sigma_e^2, \omega_{1:N}\}$. The interpretation of this model is that the mean reaction time and change in reaction time for every day of sleep deprivation are given by α_0 and α_1. The individual variations of these two quantities are captured by β_{i0}^s and β_{i1}^s, respectively.

Figure 4 presents the estimates of the posterior of the fixed and random effects using the FM (orange) and the DPM model (purple). From the upper part, we see the estimates of the mean reaction time at the first day when the participants are well-rested together with the average increase for each day. The estimates of the posterior means $\{\widehat{\alpha}_0, \widehat{\alpha}_1\}$ are $\{237, 1.15\}$ and $\{237, 1.07\}$ for the FM and the DPM model, respectively. This corresponds to that the mean reaction time the first day of the test is 237 milliseconds, which then increases by 1.15 or 1.07 milliseconds for every day of sleep deprivation.

From the lower part, we note that posterior estimate for β_{i0}^s is uni-modal with some skewness to the right. This indicates that most individuals have the same or an increased reaction time on the first day compared with the corresponding fixed effect α_0. However, from the posterior regarding β_{i1}^s, we note that there seems to be three different sub-groups with: a small decrease, small increase and large increase in reaction times for each day of sleep deprivation. This validates some of our initial findings that some individuals have a small (or even negative) change in reaction time when sleep deprived. Finally, the estimates of the random effects posterior means $\{\widehat{\beta^s}_{i0}, \widehat{\beta^s}_{i1}\}$ are $\{9.50, 8.94\}$ and $\{9.50, 9.09\}$ for the FM and the DPM model, respectively.

Conclusions

We have compared two different models for describing heterogeneity of the random effects in a mixed effects model. The numerical illustrations show that the two approaches give similar estimates of the posteriors using both synthetic and real-world data. This provides us with some promising indications that the results from Rousseau and Mengersen (2011) holds for this type of models. However, more theoretical work is required to generalise the convergence results to this type of model and priors.

Another important future work is to apply this approach for more realistic real-world models as discussed by e.g., Burda and Harding (2013). It would also be useful to conduct more extensive simulations to compare the mixing in the Markov chain to see if the approximate FM can help mitigating the problems with poor mixing as discussed by Hastie et al. (2015).

The source code and data for the numerical illustrations are available from `https://github.com/compops/panel-dpm2016/`.

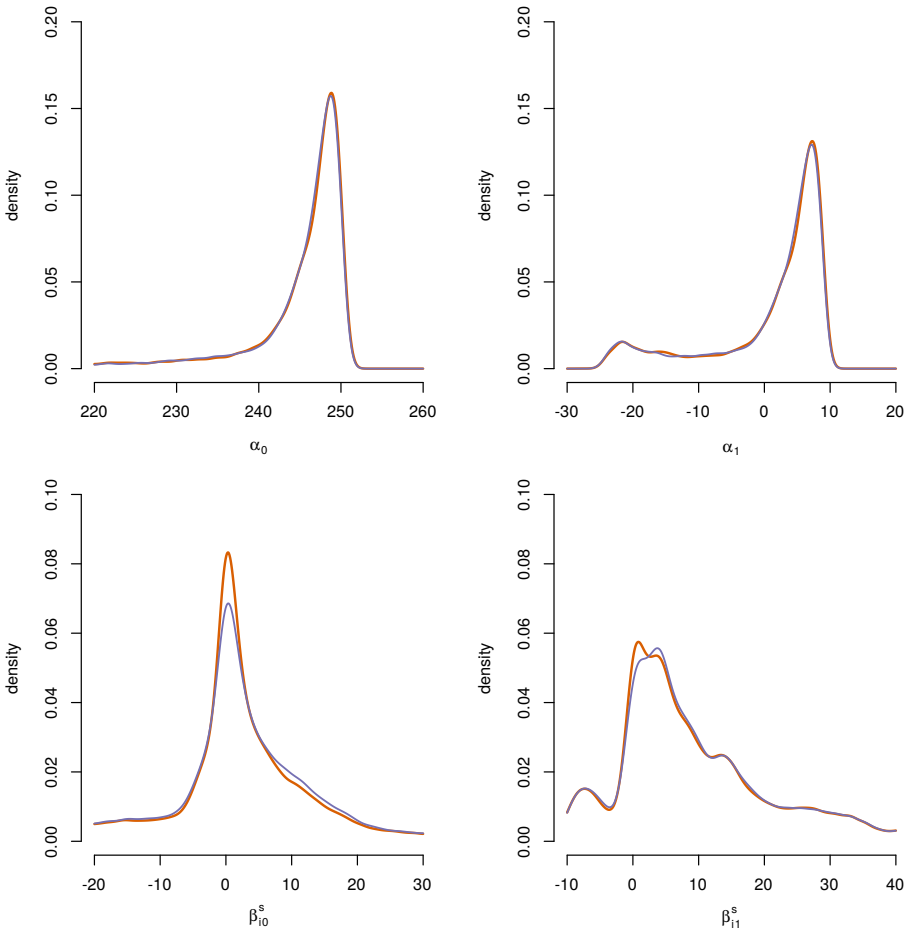


Figure 4. Posterior estimates of α_0 (upper left), α (upper right), β^s_{0i} (lower left) and β^s_{1i} (lower right) for the FM (orange) and the DPM model (purple).

Acknowledgements

The simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at Linköping University, Sweden.

Appendix

Implementation details

In all illustrations, we use $K = 10,000$ iterations in the Gibbs sampler and discard the first $K_b = 2,500$ iterations as burn-in.

Mixture of Gaussians

We use $R = 20$ components in the FM with $\eta_{0,r} = 1/R$ for $r = 1, \ldots, R$ as the concentration parameter to promote sparsity as suggested by Ishwaran and Zarepour (2002). Furthermore, we use $\eta = 1$ for the finite mixture with the true number of components $R = 3$ so that all components are occupied. For the prior of η_0 in the DPM model, we use $c_o^\eta = 1$ and $C_o^\eta = 0.5$ as the hyperparameters. The remaining hyperparameters for the priors are selected as $b_0 = 0$, $B_0 = 0.2^2$, $c_0^Q = 1$ and $c_0^Q = 1$.

Mixed effects model with synthetic data

We use $R = 20$ components in the FM with $\eta_{0,r} = 1/R$ for $r = 1, \ldots, R$ to promote sparsity as in the previous example. For the prior of η_0 in the DPM model, we use the hyperparameters $c_0^\eta = 0.01$ and $C_0^\eta = 0.01$. We make use of the mean estimate from the linear regressions of each individual to select $a_0^\star$. Hence, the first element corresponds to the mean intercept and the remaining R elements correspond to the mean slope from the N linear regression estimates. We make use of $A_0^\star = \mathcal{I}_{d+Kp}$ as the prior covariance of $\alpha^\star$. Furthermore, we select $c_0^Q = 10$, $C_0^Q = 1$, $c_0^e = 0$, $C_0^e = 0$ and $\nu = 5$.

Mixed effects model with sleep deprivation data

We use $R = 20$ components in the FM with $\eta_{0,r} = 1/R$ for $r = 1, \ldots, R$ to promote sparsity as in the previous example. For the prior of η_0 in the DPM model, we use the hyperparameters $c_0^\eta = 0.01$ and $C_0^\eta = 0.01$. We make use of the same approach as in the previous illustration to choose $a_0^\star = \{251.4, 10.5, 0, \ldots, 0\}$ and select $A_0^\star = \mathsf{diag}\{0.01, 0.005, 0.02, \ldots, 0.02\}$. Furthermore, we select $c_0^Q = 10$, $C_0^Q = 0.05\mathbf{I}_2$, $c_0^e = 1$, $C_0^e = 2$ and $\nu = 5$.

Bibliography

G. M. Allenby, N. Arora, and J. L. Ginter. On the heterogeneity of demand. *Journal of Marketing Research*, 35(3):384–389, 1998.

A. Ansari, S. Essegaier, and R. Kohli. Internet recommendation systems. *Journal of Marketing research*, 37(3):363–375, 2000.

C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

B. H. Baltagi. *Econometric analysis of panel data*. John Wiley & Sons, 2008.

D. Bates, M. Mächler, B. Bolker, and S. Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015.

G. Belenky, N. J. Wesensten, D. R. Thorne, M. L. Thomas, H. C. Sing, D. P. Redmond, M. B. Russo, and T. J. Balkin. Patterns of performance degradation and restoration during sleep restriction and subsequent recovery: A sleep dose-response study. *Journal of sleep research*, 12(1):1–12, 2003.

D. Blackwell and J. B. MacQueen. Ferguson distributions via Pólya urn schemes. *The Annals of Statistics*, 1(2):353–355, 1973.

A. Bouchard-Côté, A. Doucet, and A. Roth. Particle Gibbs split-merge sampling for Bayesian inference in mixture models. *Pre-print*, 2015. arXiv:1508.02663v1.

M. Burda and M. Harding. Panel probit with flexible correlated effects: quantifying technology spillovers in the presence of latent heterogeneity. *Journal of Applied Econometrics*, 28(6):956–981, 2013.

F. Canova. Testing for convergence clubs in income per capita: a predictive density approach. *International Economic Review*, 45(1):49–77, 2004.

M. K. Condliff, D. D. Lewis, D. Madigan, and C. Posse. Bayesian mixed-effects models for recommender systems. In *Proceedings of ACM SIGIR'99 Workshop on Recommender Systems*, Berkeley, USA, August 1999.

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.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995.

P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21, 2004.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

T. S. Ferguson. Prior distributions on spaces of probability measures. *The Annals of Statistics*, 2(4):615–629, 1974.

S. Frühwirth-Schnatter. *Finite mixture and Markov switching models*. Springer Verlag, 2006.

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.

B. Hall. *LaplacesDemon: software for Bayesian inference*, 2012. URL `http://cran.r-project.org/web/packages/LaplacesDemon/index.html`. R package version 12.05.07.

D. I. Hastie, S. Liverani, and S. Richardson. Sampling from Dirichlet process mixture models with unknown concentration parameter: mixing issues in large data implementations. *Statistics and Computing*, 25(5):1023–1037, 2015.

H. Ishwaran and M. Zarepour. Dirichlet prior sieves in finite normal mixtures. *Statistica Sinica*, 12(3):941–963, 2002.

S. Jain and R. M. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1): 158–182, 2004.

H. F. Lopes, P. Müller, and G. L. Rosner. Bayesian meta-analysis for longitudinal data models using multivariate mixture priors. *Biometrics*, 59(1):66–75, 2003.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Verlag, 2 edition, 2004.

J. Rousseau and K. Mengersen. Asymptotic behaviour of the posterior distribution in overfitted mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(5):689–710, 2011.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

Y. Ulker, B. Gunsel, and T. A. Cemgil. Sequential Monte Carlo samplers for Dirichlet process mixtures. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 876–883, Sardinia, Italy, May 2010.

F. Verbeke and E. Lesaffre. A linear mixed-effects model with heterogeneity in the random-effects population. *Journal of the American Statistical Association*, 91(433):217–221, 1996.

G. Verbeke and G. Molenberghs. *Linear mixed models for longitudinal data*. Springer Verlag, 2009.

S. G. Walker. Sampling the Dirichlet mixture model with slices. *Communications in Statistics - Simulation and Computation*, 36(1):45–54, 2007.

Paper H

Robust Input design for non-linear dynamical models

Authors: P. E. Valenzuela, J. Dahlin, C. R. Rojas and T. B. Schön

Edited version of the paper:

P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. On robust input design for nonlinear dynamical models. *Automatica*, 2016a. (provisionally accepted).

Parts of the theory presented in this paper have also been presented in:

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.

Robust Input design for non-linear dynamical models

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@liu.se

[‡]Dept. of Information Technology,
Uppsala University,
SE-751 05 Uppsala, Sweden.
thomas.schon@it.uu.se

Abstract

We present a method for robust input design for non-linear state-space models. The method optimizes a scalar cost function of the Fisher information matrix over a set of marginal distributions of stationary processes. By using elements from graph theory we characterize such a set. Since the true system is unknown, the resulting optimization problem considers a measure of the uncertainty over the set of model parameters. In addition, the required estimates of the information matrix are computed using particle methods, and the resulting problem is convex in the decision variables. Numerical examples illustrate the proposed technique by identifying models using the expectation maximization.

Keywords

System identification, input design, particle filter, non-linear systems.

Financial support from

The projects *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) and CADICS, a Linnaeus Center, both funded by the Swedish Research Council.

Introduction

Input design is concerned with generating an input signal that maximizes the information retrieved from an experiment, quantified in terms of a cost function related to the intended model application. Some of the initial contributions are discussed in Cox (1958) and Goodwin and Payne (1977). Since then, many contributions to the subject have been presented; see e.g. Fedorov (1972); Whittle (1973); Hildebrand and Gevers (2003); Gevers (2005) and the references therein.

In the case of dynamical systems, the existing results on input design are mostly focused on linear models. The assumption of a linear model structure can reduce the complexity of the problem, leading to formulations that are convex in the decision variables Ljung (1999). In this case, the convexity of the problem is achieved by designing the power spectrum of the input signal.

Several approaches to input design for linear models have been proposed in the literature involving, e.g., linear matrix inequalities (LMIs; Jansson and Hjalmarsson, 2005; Lindqvist and Hjalmarsson, 2000), Markov chains (Brighenti et al., 2009), and time domain techniques (Suzuki and Sugie, 2007). With the exception of the methods by Jansson and Hjalmarsson (2005) and Lindqvist and Hjalmarsson (2000) that rely on convexification of the problem, the previous formulations are non-convex, which illustrates the difficulty of solving the input design problem.

In recent years, there has been an interest to extend the input design methods to non-linear model structures. The main issue here is that the frequency domain methods cannot be applied to solve the input design problem, which restricts the applicability of convex formulations (Jansson and Hjalmarsson, 2005; Lindqvist and Hjalmarsson, 2000). The input design problem for non-linear models using the knowledge of linear systems is discussed in Hjalmarsson and Mårtensson (2007), where the main challenges of designing inputs for non-linear systems are presented by employing a non-linear FIR model structure. The non-linear FIR model structure is also considered in Larsson et al. (2010), where the input design problem is solved over a set of marginal distributions of stationary processes.

An extension of the input design problem to structured non-linear models is presented in Vincent et al. (2009) and Vincent et al. (2010). The model structures in Vincent et al. (2009) and Vincent et al. (2010) assume the interconnection of linear models and static non-linearities, and a realization of the optimal excitation is obtained by running a Markov chain with a prescribed stationary distribution. The class of non-linear model structures covered in input design is generalized in Forgione et al. (2014), where the input signal is optimized over an alphabet with finite cardinality.

A multilevel excitation is also discussed in De Cock et al. (2013) for identification of Wiener models, where the input is optimized using the D-criterion. The restriction to a finite alphabet is relaxed in Gopaluni et al. (2011), where the design of an input sequence for the identification of non-linear state-space models is discussed. In Gopaluni et al. (2011), the input is constrained to the class of stationary ARX processes.

A methodology to design inputs for identification of non-linear output-error models is developed in Valenzuela et al. (2013) and Valenzuela et al. (2015), which is extended in Valenzuela

et al. (2014a) to non-linear state space models. The technique proposed in Valenzuela et al. (2013), Valenzuela et al. (2015) and Valenzuela et al. (2014a) relies on graph-theoretical tools to describe a set of marginal distributions of stationary processes with finite alphabet.

The existing results on input design allow to optimize input signals when the system contains non-linear 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 multilevel excitation (Forgione et al., 2014; Larsson et al., 2010), and stationary processes (Brighenti et al., 2009; Valenzuela et al., 2013, 2014a, 2015), most part of the proposed methods cannot handle amplitude limitations on the input signal, which could arise due to physical and/or safety reasons.

The input design methods previously mentioned assume that a prior estimate of the model parameters is available for optimization. The requirement of such knowledge is a common issue in input design and different solutions to this difficulty have been proposed in the literature (Welsh and Rojas, 2009). A first approach is to consider an adaptive scheme, where the input sequence is designed as more information is collected from the system (Rojas et al., 2011; Gerencsér et al., 2009). The second option is to consider a robust input design (Rojas et al., 2007, 2012), where the input signal is designed by optimizing a measure of the uncertainty over the parameter space. In this article we address the uncertainty in the model parameters following the second option.

The main contribution of this article is to present a robust input design method for the identification of non-linear state-space models (SSM) with input constraints, which extends the model structure considered in Valenzuela et al. (2013) and Valenzuela et al. (2015), and the nominal input design presented in Valenzuela et al. (2014a). The optimal input signal is considered to be a realization of a stationary process, which optimizes a scalar cost function of the Fisher information matrix.

To pose a tractable problem, we restrict the optimization to a set of marginal distributions of stationary processes with finite alphabet. This set is characterized by a finite set of linear inequalities, and hence it can be described by a convex combination of its vertices. The vertices are cumulative distribution functions that can be found by using de Bruijn graphs, as discussed in Valenzuela et al. (2013) and Valenzuela et al. (2014a).

Since the vertices of the set are known, we can draw an input realization and compute an estimate of the Fisher information matrix for each vertex using particle methods (Doucet and Johansen, 2011; Del Moral et al., 2006). The estimates of the information matrices are computed using the method introduced in Segal and Weinstein (1989), which only needs one realization of the input-output data, and thus reducing the computational effort when estimating the Fisher information matrix compared to Valenzuela et al. (2014a). Moreover, the resulting optimization problem is convex in the decision variables.

To make the input design robust against model uncertainty, the optimization problem considers a measure of the uncertainty over the space of model parameters, which relaxes the requirements on the knowledge of the system assumed in Valenzuela et al. (2013), Valenzuela et al. (2015) and Valenzuela et al. (2014a). The method is illustrated through numerical examples, where the designed input is employed to identify an SSM using the expectation-maximization (EM; Schön et al., 2011; McLachlan and Krishnan, 2008) algorithm.

The rest of this article is organized as follows. Section 2 states the problem and the main challenges when designing inputs for identification of non-linear SSM. Section 3 describes the graph theoretical approach to input design. Section 4 discusses the estimation of the Fisher information matrix using particle methods. A summary of the proposed robust input design method is presented in Section 5. The input signal generation once the optimization is solved is presented in Section 6. To illustrate the correctness and utility of the method, two numerical examples are discussed in Section 7. Finally, Section 8 concludes this work and presents future research directions.

Notation: Throughout this article, $\mathbb{N}$ denotes the set of natural numbers, $\mathbb{R}^p$ denotes the set of p-dimensional vectors with real entries, $\mathbb{R}^{p\times r}$ is the set of $p \times r$ matrices with real entries, and $\mathbb{R}_+$ the set of positive real numbers. $\mathbf{P}$, $\mathbf{E}$, and $\mathrm{Var}\{\cdot\}$ stand for a probability measure, the expected value, and the variance, respectively. Sometimes a subscript is added to $\mathbf{P}$ and $\mathbf{E}$ to clarify the stochastic process considered by these operators. Finally, for a set $\mathcal{A}$ with finite number of elements, $\#\mathcal{A}$ denotes its cardinality.

Problem formulation

Consider a non-linear SSM that can be described for all $t \geq 1$ by

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

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

$$x_0 \sim \mu_\theta(x_0), \tag{1c}$$

where f_θ, g_θ, and μ_θ denote probability density functions (pdf) parameterized by $\theta \in \Theta \subset \mathbb{R}^{n_\theta}$ (where Θ is an open set). Here, $u_t \in \mathbb{R}^{n_u}$ denotes the input signal, $x_t \in \mathbb{R}^{n_x}$ are the (unobserved/latent) internal states, and $y_t \in \mathbb{R}^{n_y}$ are the measured outputs.

The objective in this article is to design an input signal $u_{1:n_{\mathrm{seq}}} := (u_1, \ldots, u_{n_{\mathrm{seq}}})$, as a realization of a stationary process, such that the non-linear SSM (1) can be identified with maximum accuracy as defined by a scalar function of the Fisher information matrix (Ljung, 1999). In the sequel, we will assume that there exists at least one parameter $\theta_0 \in \Theta$ such that the model (1) exactly describes the pdfs of the system, i.e., there is no undermodelling.

Given $u_{1:n_{\mathrm{seq}}}$, the Fisher information matrix is defined as

$$\mathcal{I}_F^{n_{\mathrm{seq}}}(\theta_0) := \mathbf{E}\left\{\mathcal{S}(\theta_0)\mathcal{S}^\top(\theta_0) |\, u_{1:n_{\mathrm{seq}}}\right\}, \tag{2}$$

where $\mathcal{S}(\theta_0)$ denotes the score function, i.e.,

$$\mathcal{S}(\theta_0) := \nabla_\theta\, \ell_\theta(y_{1:n_{\mathrm{seq}}})\Big|_{\theta=\theta_0}. \tag{3}$$

Here, $\ell_\theta(y_{1:n_{\mathrm{seq}}})$ denotes the log-likelihood function

$$\ell_\theta(y_{1:n_{\mathrm{seq}}}) := \log p_\theta(y_{1:n_{\mathrm{seq}}} | u_{1:n_{\mathrm{seq}}}). \tag{4}$$

We note that the expected value in (2) is with respect to the stochastic processes in (1). Since we consider $u_{1:n_{\mathrm{seq}}}$ to be a realization of a stationary process, here we are interested in the

per-sample Fisher information matrix, defined as

$$\begin{aligned}\mathcal{I}_F(\theta_0) &:= \frac{1}{n_{\text{seq}}}\mathbf{E}_u\left\{\mathcal{I}_F^{n_{\text{seq}}}(\theta_0)\right\}\\ &= \frac{1}{n_{\text{seq}}}\mathbf{E}\left\{\mathcal{S}(\theta_0)\mathcal{S}^\top(\theta_0)\right\},\end{aligned} \tag{5}$$

where the expected value in (5) is over both the stochastic processes in (1), and the stochastic vector $u_{1:n_{\text{seq}}}$.

We note that (5) depends on the cumulative distribution function (CDF) of $u_{1:n_{\text{seq}}}$, denoted by $P_u(u_{1:n_{\text{seq}}})$. Therefore, the input design problem is to find a CDF $P_u^{\text{opt}}(u_{1:n_{\text{seq}}})$ which maximizes a scalar function of (5). We define this scalar function as $\mathcal{H} : \mathbb{R}^{n_\theta \times n_\theta} \times \Theta \to \mathbb{R}$, where $\mathcal{H}$ is a matrix concave function (Boyd and Vandenberghe, 2004, p. 108). Different choices of $\mathcal{H}$ have been proposed in the literature, see e.g. Rojas et al. (2007); some examples are $\mathcal{H}(A, \theta) = \log\det(A)$, and $\mathcal{H}(A, \theta) = -\operatorname{tr}\{A^{-1}\}$ for all invertible matrices $A \in \mathbb{R}^{n_\theta \times n_\theta}$.

Since $P_u^{\text{opt}}(u_{1:n_{\text{seq}}})$ has to be a marginal CDF of a stationary process, the optimization problem must be constrained to the set

$$\begin{aligned}\mathcal{P} := \Bigg\{ P_u : \mathbb{R}^{n_{\text{seq}}} \to \mathbb{R} \Bigg|\ & P_u(\mathbf{x}) \geq 0,\ \forall \mathbf{x} \in \mathbb{R}^{n_{\text{seq}}};\\ & P_u \text{ is monotone non-decreasing};\\ & \lim_{\substack{x_i \to \infty \\ i=\{1,\ldots,n_{\text{seq}}\} \\ \mathbf{x}=(x_1,\ldots,x_{n_{\text{seq}}})}} P_u(\mathbf{x}) = 1;\\ & \int_{v\in\mathbb{R}} dP_u(v, \mathbf{z}) = \int_{v\in\mathbb{R}} dP_u(\mathbf{z}, v), \forall \mathbf{z} \in \mathbb{R}^{n_{\text{seq}}-1} \Bigg\}.\end{aligned} \tag{6}$$

The last condition in (6) (with slight abuse of notation) guarantees that $P_u \in \mathcal{P}$ is a marginal CDF associated with a stationary process (Zaman, 1983). Indeed, in order to have P_u as a valid marginal CDF of a stationary process, it must satisfy the *shift-invariant* property, namely that the marginal CDF for the first $n_{\text{seq}} - 1$ arguments in P_u must be equal to that given by the last $n_{\text{seq}} - 1$ arguments in P_u, which is translated to the last condition in (6). Hence, the set $\mathcal{P}$ considers CDFs whose marginal distributions are time invariant.

To simplify our problem, we will assume that u_t can only adopt a finite number c_{seq} of values. We denote this set of values as $\mathcal{C}$. With the previous assumption, we can define the following set, which is derived from a proper subset of $\mathcal{P}$:

$$\begin{aligned}\mathcal{P}_{\mathcal{C}} := \Bigg\{ p_u : \mathcal{C}^{n_{\text{seq}}} \to \mathbb{R} \Bigg|\ & p_u(\mathbf{x}) \geq 0,\ \forall \mathbf{x} \in \mathcal{C}^{n_{\text{seq}}};\\ & \sum_{\mathbf{x}\in\mathcal{C}^{n_{\text{seq}}}} p_u(\mathbf{x}) = 1;\\ & \sum_{v\in\mathcal{C}} p_u(v, \mathbf{z}) = \sum_{v\in\mathcal{C}} p_u(\mathbf{z}, v), \forall \mathbf{z} \in \mathcal{C}^{n_{\text{seq}}-1} \Bigg\}.\end{aligned} \tag{7}$$

The set introduced in (7) will constrain the probability mass function (PMF) p_u to the set of marginal PMFs associated with stationary processes.

So far we have been concerned with the formulation of the problem in terms of the stationary process describing $u_{1:n_{\text{seq}}}$. However, we still need to consider a remaining issue: Equation (5) depends on the parameter θ_0 describing the true system (1). Therefore, the optimal input sequence $u_{1:n_{\text{seq}}}$ depends on the parameter we want to estimate, which limits the applicability of the previous formulation in practical situations. To overcome this issue, we consider the function $\mathcal{R} : \Theta \to \mathbb{R}$ that measures the uncertainty over Θ. In the following, two definitions of $\mathcal{R}$ are considered: $\mathcal{R} = \mathbf{E}_\theta\{\cdot\}$ (where we assume that Θ is a measurable space with known CDF P_θ), and $\mathcal{R} = \min_{\theta\in\Theta}\{\cdot\}$.

To summarize, the problem we are interested in solving can be written as

Problem 1. 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

$$p_u^{\text{opt}} := \arg\max_{p_u \in \mathcal{P}_\mathcal{C}} \mathcal{R}\{\mathcal{H}(\mathcal{I}_F(\theta), \theta)\}, \tag{8}$$

with $\mathcal{H} : \mathbb{R}^{n_\theta \times n_\theta} \times \Theta \to \mathbb{R}$ a matrix concave function, $\mathcal{R} : \Theta \to \mathbb{R}$, and $\mathcal{I}_F(\theta) \in \mathbb{R}^{n_\theta \times n_\theta}$ defined as in (5). ■

Problem 1 is difficult to solve. The main challenges are:

(A) The set Θ may be uncountably infinite, which implies that the computation of $\mathcal{R}\{\mathcal{H}(\mathcal{I}_F(\theta), \theta)\}$ can be intractable.

(B) The set $\mathcal{P}_\mathcal{C}$ may be very large.

(C) The model structure (1) is very general, which implies that a closed form expression for the Fisher information matrix (5) might not be available, with the exception of special cases (e.g. when (1) is a linear Gaussian SSM).

To address issue (A), we consider a procedure that depends on $\mathcal{R}$. For $\mathcal{R} = \mathbf{E}_\theta\{\cdot\}$, we solve a Monte-Carlo approximation of Problem 1 by sampling N_s points from the set Θ according to the CDF P_θ, and replacing the expected value by its sample mean estimate. In the case that $\mathcal{R} = \min_{\theta\in\Theta}\{\cdot\}$, we employ the scenario approach (Calafiore and Campi, 2005; Welsh and Rojas, 2009). By sampling N_s points from the set Θ according to a given CDF P_s, we can rewrite Problem 1 as an optimization problem over a finite number of points in Θ. In this case we will obtain a sub-optimal solution to Problem 1 which, however, can be made close to the optimal solution by increasing N_s; we refer to Calafiore and Campi (2005); Campi and Garatti (2008) for more details.

Issue (B) will be addressed in the next section using a graph theoretical perspective, while issue (C) will be addressed in Section 4 using particle methods to approximate the Fisher information matrix.

Describing the set of stationary processes

The difficulties associated with issue (B) are:

(B.i) the PMF p_u is of dimension n_{seq}, where n_{seq} can potentially be very large, and

(B.ii) we need to represent the elements in $\mathcal{P}_{\mathcal{C}}$.

The points mentioned above make Problem 1 computationally intractable. To overcome these difficulties, we will use the graph theoretical approach introduced in Valenzuela et al. (2013), which is briefly described in this section.

To solve issue (B.i), we restrict p_u to the set of PMFs defined over $\mathcal{C}^{n_m}$, where $n_m < n_{\text{seq}}$, and corresponding to marginal PMFs of stationary processes. This assumption allows to solve an approximation of Problem 1 in the sense that the difference between the optimal cost for the solution considering $p_u(u_{1:n_{\text{seq}}})$, and the optimal cost for $p_u(u_{1:n_m})$ can be made arbitrarily small by defining n_m sufficiently close to n_{seq}.

To address issue (B.ii), we notice that all the elements in $\mathcal{P}_{\mathcal{C}}$ can be represented as a convex combination of its extreme points, since $\mathcal{P}_{\mathcal{C}}$ is described by a finite number of linear inequalities (Rockafellar, 1970, Chapter 17). 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 defined by the possible 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 $k > 0$. The resulting graph is referred to as a *de Bruijn* graph. 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 $t + 1$. In the following, $\mathcal{G}_{\mathcal{X}} = \{\mathcal{X}, \mathcal{E}\}$ will denote a de Bruijn graph with set of nodes $\mathcal{X}$ and set of edges $\mathcal{E}$.

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[1] whose set of nodes does not have a proper subset which is an elementary cycle (Zaman, 1983, pp. 678). It is known 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 de Bruijn graph $\mathcal{G}_{\mathcal{C}^{n_m}}$.

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[2]. 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 $((0, 1), (1, 0), (0, 1))$.

[1] An *elementary cycle* is a cycle where all nodes are different, except for the first and the last (Johnson, 1975, p. 77).

[2] For the examples in Section 7, we have used the algorithm presented in Johnson (1975, p. 79–80) complemented with the one proposed in (Tarjan, 1972, pp. 157).

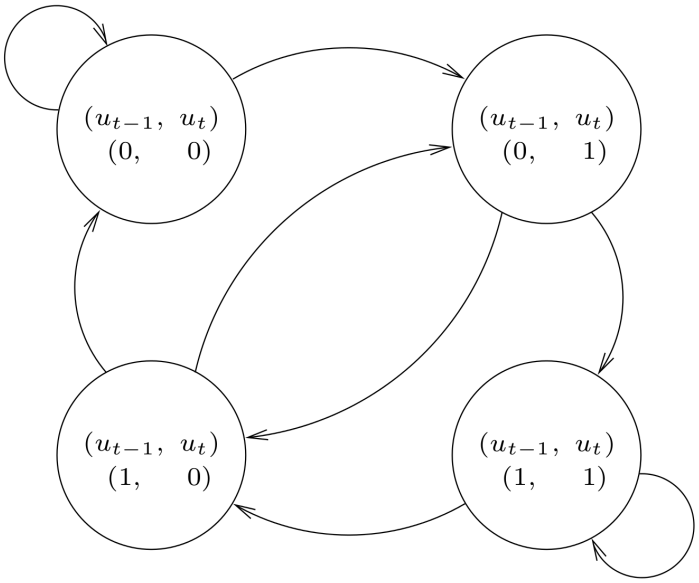


Figure 1. Example of graph derived from $\mathcal{C}^{n_m}$, with $n_m = 2$, and $\mathcal{C} := \{0, 1\}$.

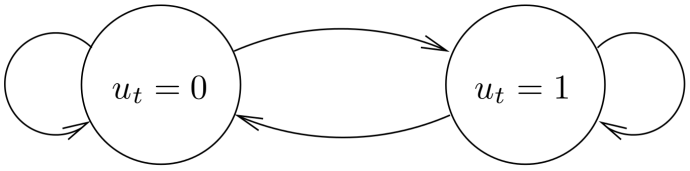


Figure 2. Example of graph derived from $\mathcal{C}^{n_m}$, with $n_m = 1$, and $\mathcal{C} := \{0, 1\}$.

Once we find the prime cycles, it is possible to generate an input sequence $u^j_{1:T}$ from v_j, which will be referred to as the *basis inputs*. As an example, we use the graph in Figure 1. One prime cycle for this graph is given by ((0, 1), (1, 0), (0, 1)). Therefore, the sequence $u^j_{1:T}$ is given by taking the value of u_t in each node, i.e., $u^j_{1:T} = \{1, 0, 1, 0, \ldots, ((-1)^{T-1} + 1)/2\}$.

Given $u^j_{1:T}$, we can use it to obtain the corresponding information matrix for $v_j \in \mathcal{V}_{\mathcal{P}_C}$, denoted by $\mathcal{I}^{(j)}_F$. However, in general the matrix $\mathcal{I}^{(j)}_F$ cannot be computed explicitly, as mentioned in issue (C) in Section 2. To overcome this problem, we use particle methods to approximate $\mathcal{I}^{(j)}_F$, as discussed in the next section.

Remark 1. For simplicity, we considered in this section a scalar input. However, the proposed approach can also be used when the model has multiple inputs. Indeed, for a multiple input formulation the states in the de Bruijn graph are associated with the possible states of an $n_u \times n_m$ matrix. The rest of the procedure for the multiple input case follows exactly the same lines as for the scalar case described here. ■

Estimation of the Fisher information matrix

The proposed method relies upon accurate estimates of the Fisher information matrix, which unfortunately is analytically intractable for general SSMs (1). Instead, we make use of statistical simulation methods known as particle smoothers to compute estimates. In this section, we provide the reader with an overview of the approach. More extensive treatments are found in Doucet and Johansen (2011) and Lindsten and Schön (2013).

Estimating the score function

From (2), we know that we can compute the Fisher information matrix by the use of the score function. However, the score function is also intractable for a general SSM but it can be estimated using particle smoothers. The key ingredient for this is the *Fisher identity* (Cappé et al., 2007) that we present in Lemma 2. For the remainder of this section, we write $\mathbf{v} := v_{1:T}$ for any vector $v_{1:T}$ to ease the notation. In the Fisher identity, $p_\theta(\mathbf{x}, \mathbf{y} \mid \mathbf{u})$ denotes the complete data log-likelihood for (1) given by

$$\log p_\theta(\mathbf{x}, \mathbf{y} \mid \mathbf{u}) = \log \mu_\theta(x_0) + \sum_{t=1}^{T} \xi_\theta(x_{t-1:t}), \tag{9}$$

$$\xi_\theta(x_{t-1:t}) := \log f_\theta(x_t \mid x_{t-1}, u_{t-1}) + \log g_\theta(y_t \mid x_t, u_t),$$

where we make use of the notation $x_{t-1:t} = \{x_{t-1}, x_t\}$.

Lemma 2 (Fisher identity (Cappé et al., 2005)). *Assume that the following hold:*

(i) For any $\theta \in \Theta$, $p_\theta(\mathbf{y} \mid \mathbf{u})$ is positive and finite.

(ii) For any $(\theta, \theta') \in \Theta \times \Theta$,

$$\int \left| \log p_\theta(\mathbf{x} \mid \mathbf{y}, \mathbf{u}) \right| p_{\theta'}(\mathbf{x} \mid \mathbf{y}, \mathbf{u})\, d\mathbf{x},$$

is finite.

(iii) $\ell_\theta(\mathbf{y}|\mathbf{u})$ *is continuously differentiable on* Θ.

(iv) For any $\theta' \in \Theta$, *the function* $\mathcal{L}_{\theta'} : \Theta \to \mathbb{R}$ *defined as*

$$\mathcal{L}_{\theta'}(\theta) := -\int \log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{u}) p_{\theta'}(\mathbf{x}|\mathbf{y},\mathbf{u})\, d\mathbf{x},$$

is continuously differentiable on Θ. *In addition,* $\mathcal{L}_{\theta'}(\theta)$ *is finite for any* $(\theta,\ \theta') \in \Theta \times \Theta$, *and*

$$\nabla_\theta \int \log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{u}) p_{\theta'}(\mathbf{x}|\mathbf{y},\mathbf{u})\, d\mathbf{x} = \int \nabla_\theta \log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{u}) p_{\theta'}(\mathbf{x}|\mathbf{y},\mathbf{u})\, d\mathbf{x}.$$

Then,

$$\mathcal{S}(\theta') = \int \nabla_\theta \log p_\theta(\mathbf{x},\mathbf{y}|\mathbf{u})\big|_{\theta=\theta'} p_{\theta'}(\mathbf{x}|\mathbf{y},\mathbf{u})\, d\mathbf{x}\,. \tag{10}$$

Proof: We refer to Cappé et al. (2005) for a proof of this lemma. □

Using (9) and (10), we arrive at the estimator

$$\mathcal{S}(\theta') = \sum_{t=1}^{T} \underbrace{\int \nabla_\theta\, \xi_\theta(x_{t-1:t})|_{\theta=\theta'} p_{\theta'}(x_{t-1:t}|\mathbf{y},\mathbf{u})\, \mathrm{d}x_{t-1:t}}_{:=\mathcal{S}_t(\theta')}\,. \tag{11}$$

To make use of this estimator, we require the *two-step smoothing distribution* $p_\theta(x_{t-1:t}|\mathbf{y},\mathbf{u})$, which is not analytically available for a general SSM. Instead, we approximate it using an empirical distribution

$$\widehat{p}_\theta(\mathrm{d}x_{t-1:t}|\mathbf{y}) := \sum_{i=1}^{N} w_T^{(i)} \delta_{x_{t-1:t}^{(i)}}(\mathrm{d}x_{t-1:t}), \tag{12}$$

where $x_t^{(i)}$ and $w_t^{(i)}$ denote particle i and its normalized weight at time t. Here, $\{x_t^{(i)}, w_t^{(i)}\}_{t=1}^T$ denotes the *particle system* generated by a particle filter and $\delta_{x'}$ denotes the Dirac measure located at $x = x'$.

In this paper, we propose to generate the particle system using the bootstrap particle filter (BPF). We outline the procedure in Algorithm 1. However, the estimator in (12) based on the BPF often suffers from poor accuracy. This is due to problems with *particle degeneracy*, see e.g., Doucet and Johansen (2011).

To mitigate this problem, we make use of a particle smoother that introduces a backward sweep after the forward run of the particle filter. Here, we use the forward-filtering backwards simulator (FFBSI) with rejection sampling and early stopping as discussed by Douc et al. (2011); Taghavi et al. (2013). There are many other interesting alternatives, we refer to Lindsten and Schön (2013) for a recent survey.

Algorithm 1 Bootstrap particle filter (BPF)

Inputs: An SSM (1), $y_{1:T}$ (observations), $u_{1:T}$ (inputs), $N \in \mathbb{N}$ (number of particles).
Output: $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$, $t = 1, \ldots, T$.
All operations are carried out over $i, j = 1, \ldots, N$.

1: Sample $x_0^{(i)} \sim \mu_\theta(x_0)$ and set $w_0^{(i)} = 1/N$.
2: **for** $t = 1$ to T **do**
3: (**Resampling**) Sample $a_t^{(i)}$ from a multinomial distribution with $\mathbf{P}\left(a_t^{(i)} = j\right) = w_{t-1}^{(j)}$.
4: (**Propagation**) Sample $x_t^{(i)} \sim f_\theta\left(x_t^{(i)} \middle| x_{t-1}^{a_t^{(i)}}, u_t\right)$.
5: Set $x_{0:t}^{(i)} = \left\{x_{0:t-1}^{a_t^{(i)}}, x_t^{(i)}\right\}$.
6: (**Weighting**) Calculate $\widetilde{w}_t^{(i)} = g_\theta\left(y_t \middle| x_t^{(i)}, u_t\right)$.
7: Normalize $\widetilde{w}_t^{(i)}$ (over i) to obtain $w_t^{(i)}$.
8: **end for**

The complete procedure for running the FFBSI is presented in Algorithm 2 and it makes use of the BPF in Algorithm 1 to create the particle system. In Algorithm 2, **Multi**$(\{p^{(i)}\}_{i=1}^N)$ denotes the multinomial distribution over N elements, with $p^{(i)}$ being the probability of choosing the i-th element. In addition, we note that the parameter ρ required by Algorithm 2 is chosen such that $f_\theta(x_{t+1}|x_t) \leq \rho$ for all $t \in \{1, \ldots, T\}$.

The computational complexity of FFBSI is of order $\mathcal{O}(\text{NMT})$, where N and M denote the number of filter and smoother particles, respectively. In general, N and M determine the accuracy of the estimates and are highly dependent on the model and on T. However, often we can have $M < N$ and still obtain reasonable accuracy but the estimates are only unbiased in the limit when N and M tend to infinity. The theoretical properties of the FFBSI are further analysed and discussed in Douc et al. (2011). We return to numerically examine the properties of the smoother when N and M are finite in Section 7.1.

The resulting estimator

From (11), we note that $\mathcal{S}(\theta')$ can be written as a sum of conditional scores $\mathcal{S}_t(\theta')$. This can be seen as a Martingale representation as each conditional score is conditionally independent given the past. Using results presented in Segal and Weinstein (1989) and Meilijson (1989), we can compute an estimate of the Fisher information matrix by

$$\widehat{\mathcal{I}}_F(\theta) = \frac{1}{T}\left\{\sum_{t=1}^{T}\left[\widehat{\mathcal{S}}_t(\theta)\right]\left[\widehat{\mathcal{S}}_t(\theta)\right]^\top - \frac{1}{T}\left[\widehat{\mathcal{S}}(\theta)\right]\left[\widehat{\mathcal{S}}(\theta)\right]^\top\right\}, \tag{13}$$

where

$$\widehat{\mathcal{S}}(\theta) := \sum_{t=1}^{T}\widehat{\mathcal{S}}_t(\theta)\,. \tag{14}$$

Algorithm 2 Fast forward-filtering backward-smoothing with early stopping (FFFBSI-ES)

Inputs: Inputs to Algorithm 1, $M \in \mathbb{N}$ (No. backward trajectories), $N_{\text{limit}} \in \mathbb{N}$ (Limit for when to stop using rejection sampling), $\rho > 0$.
Output: $\widehat{\mathcal{I}}_F(\theta)$ (estimate of the Fisher information matrix).

Run Algorithm 1 to obtain the particle system $\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^N$ for $t = 1, \ldots, T$.
Sample $\{b_T(j)\}_{j=1}^M \sim \textbf{Multi}(\{w_T^{(i)}\}_{i=1}^N)$.
Set $\tilde{x}_T^{(j)} = x_T^{b_T(j)}$ for $j = 1, \ldots, N$.
for $t = T - 1$ **to** 1 **do**
  $L \leftarrow 1, \ldots, M$.
  {Rejection sampling until N_{limit} trajectories remain.}
  while $|L| \geq N_{\text{limit}}$ **do**
    $n \leftarrow \textbf{Multi}(L)$.
    $\delta \leftarrow \emptyset$.
    Sample $\{I(k)\}_{k=1}^n \sim \textbf{Multi}(\{w_t^{(i)}\}_{i=1}^N)$.
    Sample $\{U(k)\}_{k=1}^n \sim \textbf{Uniform}([0, 1])$.
    for $k = 1$ **to** n **do**
      if $U(k) \leq f\left(\tilde{x}_{t+1}^{L(k)} | x_t^{I(k)}\right)/\rho$ **then**
        $b_t(L(k)) \leftarrow I(k)$.
        $\delta \leftarrow \delta \cup \{L(k)\}$.
      end if
    end for
    $L \leftarrow L \setminus \delta$.
  end while
  {Use standard FFBSI for the remaining trajectories.}
  for $j \in L$ **do**
    Compute $\tilde{w}_{t|T}^{(i,j)} \propto w_t^{(i)} f\left(\tilde{x}_{t+1}^{(j)} | x_t^{(i)}\right)$ for $i = 1, \ldots, N$.
    Normalize the smoothing weights $\{\tilde{w}_{t|T}^{(i,j)}\}_{i=1}^N$.
    Draw $b_t(j) \sim \textbf{Multi}\left(\{\tilde{w}_{t|T}^{(i,j)}\}_{i=1}^N\right)$.
  end for
  Set $\tilde{x}_{t:T}^{(j)} = \left\{x_t^{b_t(j)}, \tilde{x}_{t+1:T}^{(j)}\right\}$ for $j = 1, \ldots, M$.
  Calculate the estimate of the score function at time t using (11) by

$$\widehat{\mathcal{S}}_t(\theta) = \frac{1}{M} \sum_{j=1}^{M} \xi_\theta\left(\tilde{x}_{t:t+1}^{(j)}\right).$$

end for
Compute $\widehat{\mathcal{I}}_F(\theta)$ using (13).

Here we make use of Algorithm 2 to compute $\widehat{\mathcal{I}}_F(\theta)$. We can analyze the variance of this estimator using the martingale difference property. Let $\widehat{\mathcal{S}}_\ell(\theta)$ and $\widehat{\mathcal{I}}_{F,\ell m}(\theta)$ denote the ℓ-th and (ℓ, m) entry of $\widehat{\mathcal{S}}(\theta)$ and $\widehat{\mathcal{I}}_F(\theta)$, respectively. Then, for the estimator (13) we have

$$\begin{aligned} T^2 \operatorname{Var}\left\{\widehat{\mathcal{I}}_{F,\ell m}(\theta)\right\} &= \frac{(T-1)^2}{T^2} \operatorname{Var}\left\{ \sum_{t=1}^{T} \widehat{\mathcal{S}}_{t,\ell}(\theta)\widehat{\mathcal{S}}_{t,m}(\theta)\right\} \\ &\quad + \frac{2}{T}\left(\mathbf{E}\left\{\Big[\sum_{t=1}^{T} \widehat{\mathcal{S}}_{t,\ell}(\theta)\widehat{\mathcal{S}}_{t,m}(\theta)\Big]^2\right\} - \mathbf{E}\left\{\sum_{t=1}^{T}\Big[\widehat{\mathcal{S}}_{t,\ell}(\theta)\widehat{\mathcal{S}}_{t,m}(\theta)\Big]^2\right\}\right) \\ &\quad + \frac{1}{T^2}\left(\mathbf{E}\left\{\Big[\widehat{\mathcal{S}}_{\ell}(\theta)\widehat{\mathcal{S}}_{m}(\theta)\Big]^2\right\} - \mathbf{E}\left\{\Big[\sum_{t=1}^{T} \widehat{\mathcal{S}}_{t,\ell}(\theta)\widehat{\mathcal{S}}_{t,m}(\theta)\Big]^2\right\}\right). \end{aligned} \tag{15}$$

We note that the variance is bounded for a fixed T, provided that the variances of each element in $\sum_{t=1}^{T}\left[\widehat{\mathcal{S}}_t(\theta)\right]\left[\widehat{\mathcal{S}}_t(\theta)\right]^\top$ and $\left[\widehat{\mathcal{S}}(\theta)\right]\left[\widehat{\mathcal{S}}(\theta)\right]^\top$ together with their second order moments are bounded. The latter is satisfied by FFBSi under some regularity assumptions as discussed in Douc et al. (2011). Hence, we conclude that this estimator is consistent as $T \to \infty$ when $\widehat{\mathcal{S}}(\theta)$ is unbiased, which means that both N, and M should also tend to infinity. Clearly, this is not satisfied in any practical application and we investigate the accuracy of the estimate in Section 7.1 for the finite sample case.

An alternative for estimating $\mathcal{I}_F(\theta)$ is to use the sample covariance matrix of the score function as discussed in Valenzuela et al. (2014a). In this approach, we estimate the score function $\widehat{\mathcal{S}}^{(k)}$ using particle methods over $k \in \{1, \ldots, K\}$ different realizations of the SSMs based on a single realization of the input. Hence, we obtain the estimate by

$$\widehat{\mathcal{I}}_F(\theta) = \frac{1}{KT}\sum_{k=1}^{K}\left[\widehat{\mathcal{S}}^{(k)}(\theta)\right]\left[\widehat{\mathcal{S}}^{(k)}(\theta)\right]^\top. \tag{16}$$

The variance of this estimator is given by

$$T^2 \operatorname{Var}\left\{\widehat{\mathcal{I}}_{F,\ell m}(\theta)\right\} = \frac{1}{K}\operatorname{Var}\left\{\widehat{\mathcal{S}}_\ell^{(k)}(\theta)\widehat{\mathcal{S}}_m^{(k)}(\theta)\right\}, \tag{17}$$

for some $k \in \{1, \ldots, K\}$. This implies that the accuracy of (16) increases with the number of realizations K, provided that the variance of each element in $\left[\widehat{\mathcal{S}}^{(k)}(\theta)\right]\left[\widehat{\mathcal{S}}^{(k)}(\theta)\right]^\top$ is bounded, which again is satisfied by FFBSi. Therefore, we conclude that the variance of the estimator (16) is bounded as $T \to \infty$ if N and M also increases sufficiently fast, and it goes to zero as $K \to \infty$.

The main benefit of using the estimator in (13) instead of (16) is a smaller computational cost. In the former, we only require estimating the score function for a single realization of the SSM, whereas we require K such estimates in the latter case. It is difficult to establish theoretically which of the two estimators has better accuracy. However, we have observed numerically that the new estimator outperforms the latter in terms of both accuracy and computational cost. A third alternative discussed by Poyiadjis et al. (2011) and Dahlin et al. (2015) is to make use of the Louis identity to estimate the Fisher information matrix, which often results in negative definite estimate, which is a problem in our application.

Final input design method

The proposed method to design input signals in $\mathcal{C}^{n_m}$ for the identification of non-linear SSMs is summarized in Algorithm 3. The method solves an approximation of Problem 1, which can be written as

$$\gamma^\star = \underset{\gamma=\{\alpha_j\}_{j=1}^{n_\mathcal{V}}}{\operatorname{argmax}} \widehat{\mathcal{R}}\left(\mathcal{H}(\mathcal{I}_F^{\text{app}}(\gamma,\,\theta),\,\theta)\right) \tag{18}$$

$$\text{subject to} \quad \mathcal{I}_F^{\text{app}}(\gamma,\,\theta_i) = \sum_{j=1}^{n_\mathcal{V}} \alpha_j\, \widehat{\mathcal{I}}_F^{(j)}(\theta_i) \qquad \sum_{j=1}^{n_\mathcal{V}} \alpha_j = 1\,,$$

for $i \in \{1,\,\ldots,\,N_s\}$ and $j \in \{1,\,\ldots,\,n_\mathcal{V}\}$. Here, $\alpha_j \geq 0$ and $\widehat{\mathcal{R}}$ denotes the approximation of the function $\mathcal{R}$ when the set Θ is approximated by $\{\theta_i\}_{i=1}^{N_s}$. The implementation of $\widehat{\mathcal{R}}$ for the cost functions considered in this article follows the solution presented for issue (A) in Section 2. Thus, for $\mathcal{R} = \mathbf{E}_\theta\{\,\cdot\,\}$ we have

$$\widehat{\mathcal{R}}(\mathcal{H}(\mathcal{I}_F^{\text{app}}(\gamma,\,\theta),\,\theta)) = \frac{1}{N_s}\sum_{i=1}^{N_s} \mathcal{H}(\mathcal{I}_F^{\text{app}}(\gamma,\,\theta_i),\,\theta_i)\,, \tag{19}$$

and for $\mathcal{R} = \min_{\theta\in\Theta}\{\,\cdot\,\}$ we obtain

$$\widehat{\mathcal{R}}(\mathcal{H}(\mathcal{I}_F^{\text{app}}(\gamma,\,\theta),\,\theta)) = \min_{\theta\in\{\theta_i\}_{i=1}^{N_s}} \mathcal{H}(\mathcal{I}_F^{\text{app}}(\gamma,\,\theta),\,\theta)\,. \tag{20}$$

Algorithm 3 computes the vector $\gamma^\star = \{\alpha_j^\star\}_{j=1}^{n_\mathcal{V}}$ 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 $\mathcal{V}_{\mathcal{P}_\mathcal{C}}$, according to

$$p_u^{\text{opt}} = \sum_{j=1}^{n_\mathcal{V}} \alpha_j^\star\, v_j\,. \tag{21}$$

Input signal generation

In Section 3 we discussed a parametrization of $\mathcal{P}_\mathcal{C}$ to obtain a computationally tractable description of the PMFs in this set. However, a remaining issue must be addressed: given a PMF in $\mathcal{P}_\mathcal{C}$, say p, we want to obtain an input vector $u_{1:n_{\text{seq}}}$, where each u_t is sampled from p, for $t \in \{1,\,\ldots,\,n_{\text{seq}}\}$. In this section we present a procedure to generate an input sequence $u_{1:n_{\text{seq}}}$ from a given PMF $p(u_{1:n_m}) \in \mathcal{P}_\mathcal{C}$, which has been introduced in Valenzuela et al. (2015). To this end, we note that it is possible to associate $\mathcal{G}_{\mathcal{C}^{n_m}}$ with the discrete-time Markov chain (Doob, 1953)

$$\pi_{k+1} = A\,\pi_k\,, \tag{24}$$

where $A \in \mathbb{R}^{\mathcal{C}^{n_m}\times\mathcal{C}^{n_m}}$ is a transition probability matrix[3], and $\pi_k \in \mathbb{R}^{\mathcal{C}^{n_m}}$ is the state vector. In this case, there is a one-to-one correspondence between each entry of $\pi_k \in \mathbb{R}^{\mathcal{C}^{n_m}}$ and an element of $\mathcal{C}^{n_m}$.

[3]Given a set of finite cardinality X, we denote by $\mathbb{R}^{X\times X}$ the matrices with real entries, with dimensions given by the cardinality of X.

Algorithm 3 New input design method

Inputs: $\mathcal{C}$ (input values), n_m, T (number of input samples), $\mathbf{P}$ (probability measure over Θ), and N_s (number of samples over Θ).
Output: $\gamma^\star$ (optimal weighting of the basis inputs).

1: Sample $\{\theta_i\}_{i=1}^{N_s}$ from Θ according to $\mathbf{P}$.
2: Compute all the elementary cycles of $\mathcal{G}_{\mathcal{C}^{n_m-1}}$ using, e.g., Johnson (1975, p. 79–80), Tarjan (1972, p. 157).
3: 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 in Section 3 (cf. Zaman (1983, Lemma 4)).
4: Generate the input signals $u^j_{1:T}$ from the prime cycles of $\mathcal{G}_{\mathcal{C}^{n_m}}$, for each $j \in \{1, \ldots, n_{\mathcal{V}}\}$.
5: **for** $i = 1$ to N_s **do**
6: Execute Algorithm 2 based on θ_i and $\{u^j_{1:T}\}_{j=1}^{n_{\mathcal{V}}}$ to obtain $\{\widehat{\mathcal{I}}_F^{(j)}(\theta_i)\}_{j=1}^{n_{\mathcal{V}}}$.
7: **end for**
8: Solve the optimization problem (18) to obtain $\gamma^\star$.

Algorithm 4 Design of a transition probability matrix

Inputs: A PMF $p \in \mathcal{P}_{\mathcal{C}}$, and n_m.
Output: A transition probability matrix A with stationary distribution p.

1: For each $r \in \mathcal{C}^{n_m}$, define

$$\mathcal{N}_r := \{l \in \mathcal{C}^{n_m} : (l, r) \in \mathcal{E}\}. \tag{22}$$

In other words, $\mathcal{N}_r$ is the set of ancestors of r.
2: For each r, $l \in \mathcal{C}^{n_m}$, let

$$A_{rl} = \begin{cases} \frac{\mathbf{P}\{r\}}{\sum_{k\in\mathcal{N}_r}\mathbf{P}\{k\}}, & \text{if } l \in \mathcal{N}_r \text{ and } \sum_{k\in\mathcal{N}_r}\mathbf{P}\{k\} \neq 0, \\ \frac{1}{\#\mathcal{N}_r}, & \text{if } l \in \mathcal{N}_r \text{ and } \sum_{k\in\mathcal{N}_r}\mathbf{P}\{k\} = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

Based on this association, $p(u_{1:n_m})$ corresponds to $\Pi^s \in \mathbb{R}^{\mathcal{C}^{n_m}}$, the stationary distribution of a Markov chain of the form (24). Therefore, in order to generate an input sequence $u_{1:n_{\text{seq}}}$ from $p(u_{1:n_m})$, we can design a Markov chain having $p(u_{1:n_m})$ as its stationary distribution, and simulate this Markov chain to generate $u_{1:n_{\text{seq}}}$ from its samples.

If $A_{rl} \in \mathbb{R}$ denotes the (r, l)-entry in A, then a valid A for the Markov chain (24) must satisfy the following conditions

$$A_{rl} \geq 0, \text{ for all } r, l \in \mathcal{C}^{n_m}, \tag{25a}$$

$$\sum_{r \in \mathcal{C}^{n_m}} A_{rl} = 1, \text{ for all } l \in \mathcal{C}^{n_m}, \tag{25b}$$

$$A_{rl} = 0, \text{ if } (l, r) \notin \mathcal{E}. \tag{25c}$$

Note that the indices of A are not numerical, but belong to $\mathcal{C}^{n_m}$. Algorithm 4 presents a method to design a transition probability matrix A for $\mathcal{G}_{\mathcal{C}^{n_m}}$ with Π^s as stationary distribution. Here we will only make use of this result to generate $u_{1:n_{\text{seq}}}$ with distribution given by the optimal PMF $p_u^{\text{opt}}(u_{1:n_m})$. We refer to Valenzuela et al. (2015) for more details about Algorithm 4, where its validity is established.

Numerical illustrations

In this section, we discuss numerical simulations to illustrate some aspects of the proposed input design method in two SSMs. In the first illustration, we make use of a linear Gaussian state space (LGSS) model to evaluate the accuracy of the estimator for the Fisher information matrix and the impact of the design parameters in Algorithm 2. We also compare the solution obtained for the input design problem to some standard input signals as a sanity check of that the method works. In the second illustration, we consider a non-linear state space model, which is more challenging from an input design perspective.

Accuracy of information matrix estimation

Consider an LGSS model given by

$$x_{t+1} \,|\, x_t \sim \mathcal{N}(x_{t+1}; \phi x_t + u_t, \sigma_v^2), \quad y_t \,|\, x_t \sim \mathcal{N}(y_t; x_t, 0.1^2), \tag{26}$$

where the parameter vector is $\theta = \{\phi, \sigma_v\}$ with the state persistence $\phi \in (-1, 1)$ and the standard deviation of the innovations $\sigma_v \in \mathbb{R}_+$.

Kalman methods

In our application, we require accurate estimates of the Fisher information matrix. In general, this accuracy depends on both the state inference problem and on the model. For the LGSS model, we can solve the former exactly using Kalman methods. Hence, we can investigate the properties of the estimator in (13) without the influence of the variance induced by the particle filter. More specifically, we are interested in the accuracy of the estimator as T grows large as discussed in Section 4.2.

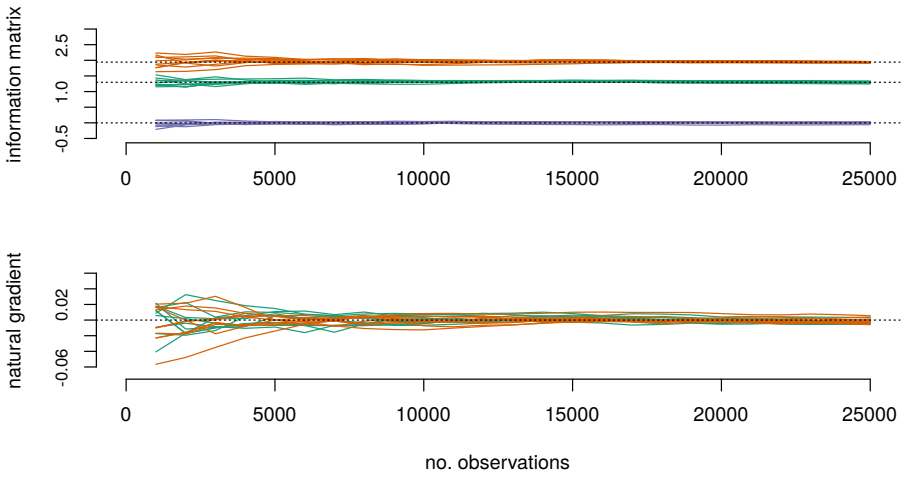


Figure 3. Top: the information matrix element for ϕ (orange), σ_v (green) and cross-term between ϕ and σ_v (blue). Bottom: the natural gradient for different number of observations T. The plots show the results from 25 Monte Carlo simulations.

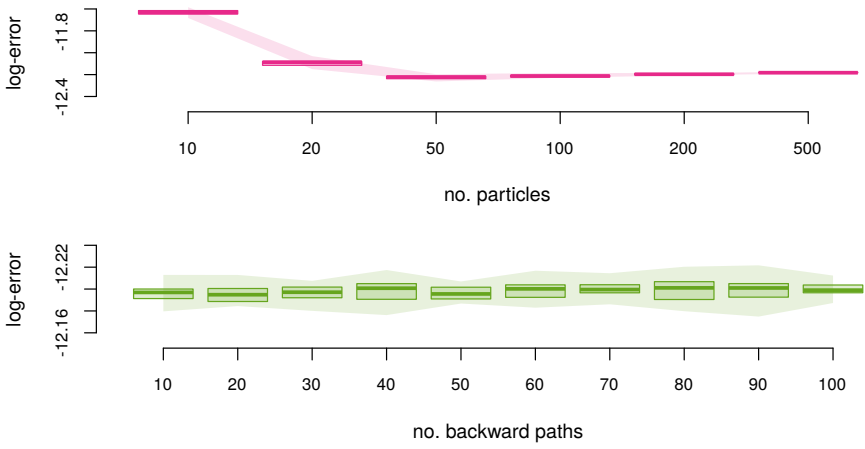


Figure 4. The logarithm of the error in Frobenius norm of the estimate of the information matrix when varying N (top) and M (bottom). The plots show the results from 25 Monte Carlo simulations and the shaded areas indicate the span of the errors.

To this end, we simulate 25 data sets with $T = 25 \cdot 10^3$ observations using the parameter $\theta_0 = \{0.5, 1.0\}$ without any input ($u_t \equiv 0$). We solve the state inference problem using the Kalman smoother (Rauch et al., 1965) and estimate the information matrix using (13) while varying the amount of observations. In Figure 3, we present the elements of the information matrix and the resulting natural gradient (the gradient scaled by the inverse Hessian) when varying T. We see that the estimator stabilizes after about $T \approx 15 \cdot 10^3$ observations. Hence, we conclude that this is the number of observations required to obtain accurate estimates of the Fisher information matrix for this model. Furthermore, we observe that the natural gradients are almost zero at this value of T, indicating that the maximum likelihood estimate of parameter vector is indeed θ_0.

Particle methods

For a general SSM, we cannot solve the state inference problem exactly and we need to make use of approximations obtained by e.g., particle methods. Here, we are interested in observing how the accuracy of the estimates of the Fisher information matrix depends on the choice of the number of particles N and the number of backwards trajectories M in Algorithm 2. We fix the number of observations to $T = 15 \cdot 10^3$ based on the results for the Kalman smoother in the previous section. Furthermore, we use of the fully adapted particle filter (FAPF; Pitt and Shephard, 1999), which reduces the computational cost and increases the accuracy of the state estimates.

In the uppper part of Figure 4, we present the log-error in the Frobenius norm of the information matrix when varying N and M. In the first case, we fix $M = \lfloor N/4 \rfloor$ (the closest integer to $N/4$ from below) and vary N between 10 and 500 particles. For this model, we conclude that 200 particles are enough to obtain good accuracy of the estimate. Note that the increase of the log-error of the box plots in Figure 4 for $N \geq 50$ is probably due to the randomness of the Monte Carlo simulations.

In the second case, we fix $N = 200$ and vary the number of backward paths in the set $M \in \{10, 20, \ldots, 100\}$. We present the results in the lower part of Figure 4. Here, we note that the accuracy of the estimates is quite robust to the choice of M and increasing the number of backward trajectories above 10 seems to be a waste of computational resources.

Input design for the LGSS model

We now return to the problem of designing an input sequence $u_{1:T}$ for the LGSS model (26) such that the parameters $\theta = \{\phi, \sigma_v\}$ can be estimated with a high precision. As discussed before, we can make use of both Kalman methods and the FAPF algorithm to solve the state estimating problem and estimate the Fisher information matrix. Hence, we make use of both methods together with the procedure outlined in Algorithm 3 to compare the approximate particle implementation to the optimal Kalman implementation. Furthermore, we make use of the same parameters and settings as in Section 7.1.

The robust input design problem considers the set of model parameters given by $\Theta = \{(\phi, \sigma_v) : \phi \in [0.4, 0.6], \sigma_v \in [0.8, 1.2]\}$. For Algorithm 3, we use $n_m = 2$, and $c_{\text{seq}} = 3$ values (-1, 0 and 1), resulting in 8 different basis inputs. Also, we use $\mathcal{H}(\mathcal{I}_F(\theta), \theta) =$

	Kalman (KM)		Particle (PM)	
Input	$\widehat{\phi}$	$\widehat{\sigma_v}$	$\widehat{\phi}$	$\widehat{\sigma_v}$
none	-9.04	-9.04	-2.84	-2.84
constant	-15.33	**-13.47**	**-6.29**	-5.41
uniform	-12.88	-13.44	-5.79	**-5.43**
mean case (KM)	-14.13	-13.46	-5.91	-5.42
worst case (KM)	-15.33	-13.45	-6.29	-5.41
mean case (PM)	**-15.37**	**-13.47**	**-6.29**	-5.41
worst case (PM)	-15.33	**-13.47**	-6.29	-5.41

Table 1. Log-MSE of the parameter estimates in the LGSS model obtained using EM with different inputs. Bold face marks the best values.

Input	Log-MSE		Bootstrapped 95% CI	
	$\widehat{\gamma}$	$\widehat{\beta}$	$\widehat{\gamma}$	$\widehat{\beta}$
none	-2.54	-3.48	0.011	0.067
constant	3.45	-2.76	0.704	0.020
uniform	-1.67	-4.65	0.167	0.038
mean case	-1.11	-4.99	0.187	**0.025**
worst case	**-1.71**	**-5.14**	**0.157**	0.032

Table 2. Left: log-MSE of the parameter estimates in the NL model obtained using EM with different inputs. Right: length of the bootstrapped 95% confidence intervals for the estimated parameters. Bold face marks the best values.

$\log\det(\mathcal{I}_F(\theta))$ as the scalar cost function of the Fisher information matrix. By using $N_s = 100$ realizations uniformly sampled from the parameter space Θ, we can estimate the optimal weighting of the basis inputs using both Kalman and particle methods.

Parameter inference

With the optimal input generated, we would like to evaluate the effect of applying the input to the system in terms of the uncertainty in the estimated parameters. Here, we make use of the expectation maximization (EM) algorithm from Schön et al. (2011). We consider $N = 200$ particles, $M = 20$ backward trajectories and 150 iterations of the EM algorithm to identify the parameters.

In Table 1, we present the logarithm of the mean squared error (MSE) of the parameter estimates for different inputs: none ($u_t \equiv 0$), constant ($u_t \equiv 1$), uniform random input and the optimal inputs constructed using the proposed technique with the Kalman method (KM) and the particle method (PM) when $\mathcal{R} = \mathbf{E}_\theta\{\cdot\}$ (mean case), and $\mathcal{R} = \min_{\theta\in\Theta}\{\cdot\}$ (worst case). From these results, we conclude that the input design technique using particle methods attains similar log-MSE values than those obtained with a constant signal ($u_t = 1$). Hence, the input design approach works for the simple LGSS model.

Input design for a non-linear model

As a second example, we consider a non-linear (NL) SSM of the form

$$x_{t+1} \mid x_t \sim \mathcal{N}\Big(x_{t+1}; \frac{1}{\gamma + x_t^2} + u_t, 0.1^2\Big), \tag{27a}$$

$$y_t \mid x_t \sim \mathcal{N}\Big(y_t; \beta x_t^2, 1^2\Big), \tag{27b}$$

where the parameters are $\theta = \{\gamma, \beta\}$. We generate $T = 10^3$ observations from the model with $\theta_0 = \{2, 0.8\}$.

For Algorithm 3, we use $n_m = 2$, and $c_{\text{seq}} = 4$ values (-1, $-1/3$, $1/3$ and 1), resulting in 24 different basis inputs. Otherwise, we apply the same procedure as before to determine the optimal input and to carry out the parameter inference. Note that for this model, we cannot make use of the Kalman smoother or the FAPF as the model is not linear in the parameters and state. Instead, we make use of the BPF using $N = 2.5 \cdot 10^3$ particles and $M = 100$ backward trajectories in the FFBSI smoother. In addition, we consider the parameter set described as $\Theta = \{(\beta, \gamma) : \gamma \in [1.6, 2.4], \beta \in [0.6, 1]\}$. The cost functions $\mathcal{R}$ are computed by approximating Θ using $\{\theta_i\}_{i=1}^{N_s}$, where $N_s = 30$ and each θ_i is sampled from a uniform distribution over Θ.

In Figure 5, we present 40 realizations of the estimated parameter vector at each iteration of the EM algorithm for different inputs. We note that the parameter β in the model (27) is easier to estimate than the parameter γ. This can be explained by noticing that the measurement equation in (27) is linear in β, which is not the case for γ in the state equation. Moreover, the expression x_t^2 in the measurement equation implies that two different values for the state can equally represent a value for y_t. Therefore, the parameter estimation problem for the model (27) is difficult to solve.

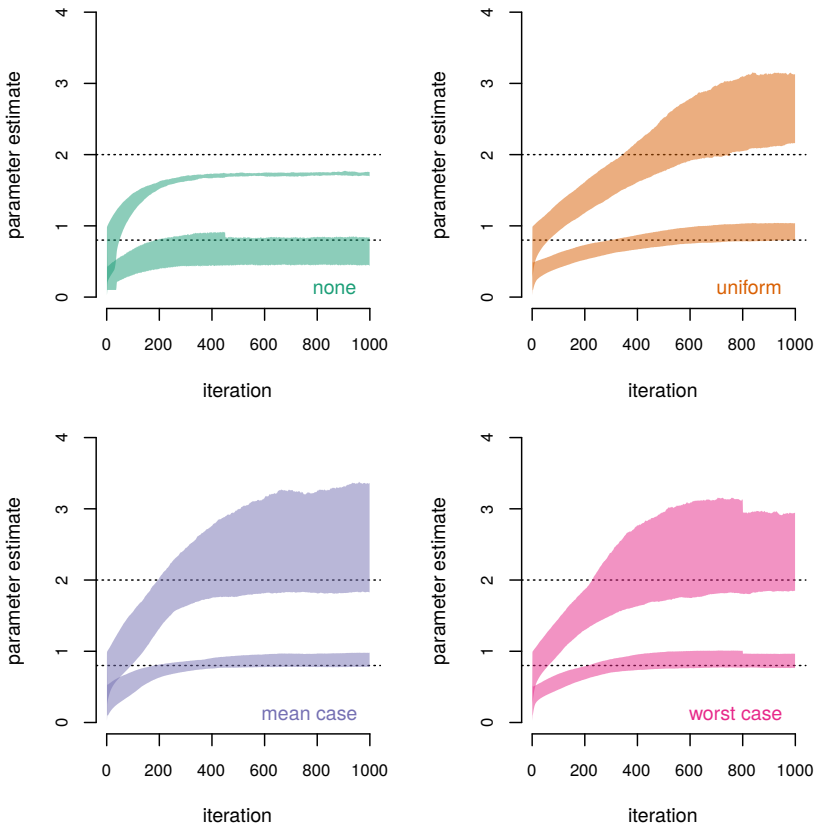


Figure 5. The evolution of the estimated parameters in the NL model obtained from 40 runs of the EM algorithm with random initializations and different inputs (none, uniform, mean optimal input, and worst case optimal input). For each plot, the upper curves represent the value of $\widehat{\gamma}$ at each iteration, while the lower ones show the value of $\widehat{\beta}$ at each iteration. The shaded area indicates the span between the maximum and minimum values.

An interesting result derived from Figure 5 is that the optimal input accelerates the convergence of the model parameters when compared to a uniformly distributed input. Indeed, for the optimal inputs the parameter estimates stabilize after approximately 500 iterations, and the same is achieved after approximately 800 iterations for the uniform input.

Table 2 presents the log-MSE of the parameter estimates using different input signals. It seems that applying a zero input would lead to more accurate results for the parameter γ. However, from Figure 5 we see that the parameter γ is not correctly identified, and thus the log-MSE value is affected by a non-zero bias. The same conclusion can be drawn for the estimated parameters using a constant input. On the other hand, we note that a nonconstant input helps to improve the accuracy of the estimated parameters. In particular, the designed input for the worst case achieves better accuracy than the uniform excitation.

To confirm this, Table 2 also shows the bootstrapped 95% confidence interval (CI) for the estimated parameters, which is obtained after 1,000 resamplings and using the adjusted bootstrap percentile (Davison and Hinkley, 1997). We note that the length of the confidence interval for the worst case input is smaller than the one obtained with the uniform input. The optimal input obtained in the mean case also reduces the length of the confidence interval for the parameter $\widehat{\beta}$. Hence, the input design method can be employed to obtain better parameter estimates in this challenging model.

Conclusions

In this article a robust input design method for the identification of non-linear SSMs is presented. The problem considers the design of an input sequence as a realization of a stationary process maximizing a scalar cost function of the Fisher information matrix. Since the model parameters are unknown a priori, the optimization of the experiment considers a measure of the uncertainty over the space of model parameters.

Numerical examples show that the method helps to improve the accuracy of the estimated parameters when the EM algorithm is employed to identify the model.

Future work on the subject will be focused on reducing the computational effort required to obtain a tractable parametrization of the optimization set, and extensions of the proposed method to more general input classes.

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, Shanghai, China, December 2009.

G. Calafiore and M.C. Campi. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102(1):25–46, 2005.

M. C. Campi and S. Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230, 2008.

O. Cappé, E. Moulines, and T. Rydén. *Inference in hidden Markov models*. Springer Verlag, 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.

D. R. Cox. *Planning of experiments*. New York: Wiley, 1958.

J. Dahlin, F. Lindsten, and T. B. Schön. Particle Metropolis-Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015.

A.C. Davison and D.V. Hinkley. *Bootstrap methods and their application*. Cambridge University Press, 1997.

A. De Cock, M. Gevers, and J. Schoukens. A preliminary study on optimal input design for nonlinear systems. In *Proceedings of the 52nd Conference on Decision and Control (CDC)*, pages 4931–4936, Florence, Italy, December 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.

J. L. Doob. *Stochastic processes*. New York Wiley, 1953.

R. Douc, A. Garivier, E. Moulines, and J. Olsson. Sequential Monte Carlo smoothing for general state space hidden Markov models. *Annals of Applied Probability*, 21(6):2109–2145, 2011.

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.

M. Forgione, X. Bombois, P. M. J. Van den Hof, and H. Hjalmarsson. Experiment design for parameter estimation in nonlinear systems based on multilevel excitation. In *Proceedings of the 13th European Control Conference (ECC)*, Strasbourg, France, June 2014.

L. Gerencsér, H. Hjalmarsson, and J. Mårtensson. Identification of ARX systems with non–stationary inputs–asymptotic analysis with application to adaptive input design. *Automatica*, 45(3):623–633, 2009.

M. Gevers. Identification for control: from the early achievements to the revival of experiment design. *European Journal of Control*, 11:1–18, 2005.

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 dynamic 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*, pages 1572–1576, New York, United States, 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, 2005.

D. B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, mar 1975.

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*, Santa Barbara, California, USA, July 2000.

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.

L. Ljung. *System Identification. Theory for the User, 2nd ed.* Upper Saddle River, NJ: Prentice-Hall, 1999.

G. McLachlan and T. Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2008.

I. Meilijson. A fast improvement to the EM algorithm on its own terms. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 51(1):127–138, 1989.

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

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.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

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.

C. R. Rojas, J. C. Aguero, J. S. Welsh, G. C. Goodwin, and A. Feuer. Robustness in experiment design. *IEEE Transactions on Automatic Control*, 57(4):860–874, 2012.

T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, January 2011.

M. Segal and E. Weinstein. A new method for evaluating the log-likelihood gradient, the Hessian, and the Fisher information matrix for linear dynamic systems. *IEEE Transactions on Information Theory*, 35(3):682–687, 1989.

H. Suzuki and T. Sugie. On input design for system identification in time domain. In *Proceedings of the European Control Conference (ECC)*, Kos, Greece, July 2007.

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.

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)*, pages 5740–5745, Florence, Italy, December 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* 19*th IFAC World Congress*, Cape Town, South Africa, August 2014a.

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 2014b.

P. E. Valenzuela, C. R. Rojas, and H. Hjalmarsson. A graph theoretical approach to input design for identification of nonlinear dynamical models. *Automatica*, 51:233–242, 2015.

P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. On robust input design for nonlinear dynamical models. *Automatica*, 2016. (provisionally accepted).

T. L. Vincent, C. Novara, K. Hsu, and K. Poolla. Input design for structured nonlinear system identification. In *Proceedings of the 15th IFAC Symposium on System Identification*, pages 174–179, Saint-Malo, France, July 2009.

T. L. Vincent, C. Novara, K. Hsu, and K. Poolla. Input design for structured nonlinear system identification. *Automatica*, 46(6):990–998, 2010.

J.S. Welsh and C.R. Rojas. A scenario based approach to robust experiment design. In *Proceedings of the 15th IFAC Symposium on System Identification*, 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.

PhD Dissertations
Division of Automatic Control
Linköping University

M. Millnert: Identification and control of systems subject to abrupt changes. Thesis No. 82, 1982. ISBN 91-7372-542-0.

A. J. M. van Overbeek: On-line structure selection for the identification of multivariable systems. Thesis No. 86, 1982. ISBN 91-7372-586-2.

B. Bengtsson: On some control problems for queues. Thesis No. 87, 1982. ISBN 91-7372-593-5.

S. Ljung: Fast algorithms for integral equations and least squares identification problems. Thesis No. 93, 1983. ISBN 91-7372-641-9.

H. Jonson: A Newton method for solving non-linear optimal control problems with general constraints. Thesis No. 104, 1983. ISBN 91-7372-718-0.

E. Trulsson: Adaptive control based on explicit criterion minimization. Thesis No. 106, 1983. ISBN 91-7372-728-8.

K. Nordström: Uncertainty, robustness and sensitivity reduction in the design of single input control systems. Thesis No. 162, 1987. ISBN 91-7870-170-8.

B. Wahlberg: On the identification and approximation of linear systems. Thesis No. 163, 1987. ISBN 91-7870-175-9.

S. Gunnarsson: Frequency domain aspects of modeling and control in adaptive systems. Thesis No. 194, 1988. ISBN 91-7870-380-8.

A. Isaksson: On system identification in one and two dimensions with signal processing applications. Thesis No. 196, 1988. ISBN 91-7870-383-2.

M. Viberg: Subspace fitting concepts in sensor array processing. Thesis No. 217, 1989. ISBN 91-7870-529-0.

K. Forsman: Constructive commutative algebra in nonlinear control theory. Thesis No. 261, 1991. ISBN 91-7870-827-3.

F. Gustafsson: Estimation of discrete parameters in linear systems. Thesis No. 271, 1992. ISBN 91-7870-876-1.

P. Nagy: Tools for knowledge-based signal processing with applications to system identification. Thesis No. 280, 1992. ISBN 91-7870-962-8.

T. Svensson: Mathematical tools and software for analysis and design of nonlinear control systems. Thesis No. 285, 1992. ISBN 91-7870-989-X.

S. Andersson: On dimension reduction in sensor array signal processing. Thesis No. 290, 1992. ISBN 91-7871-015-4.

H. Hjalmarsson: Aspects on incomplete modeling in system identification. Thesis No. 298, 1993. ISBN 91-7871-070-7.

I. Klein: Automatic synthesis of sequential control schemes. Thesis No. 305, 1993. ISBN 91-7871-090-1.

J.-E. Strömberg: A mode switching modelling philosophy. Thesis No. 353, 1994. ISBN 91-7871-430-3.

K. Wang Chen: Transformation and symbolic calculations in filtering and control. Thesis No. 361, 1994. ISBN 91-7871-467-2.

T. McKelvey: Identification of state-space models from time and frequency data. Thesis No. 380, 1995. ISBN 91-7871-531-8.

J. Sjöberg: Non-linear system identification with neural networks. Thesis No. 381, 1995. ISBN 91-7871-534-2.

R. Germundsson: Symbolic systems – theory, computation and applications. Thesis No. 389, 1995. ISBN 91-7871-578-4.

P. Pucar: Modeling and segmentation using multiple models. Thesis No. 405, 1995. ISBN 91-7871-627-6.

H. Fortell: Algebraic approaches to normal forms and zero dynamics. Thesis No. 407, 1995. ISBN 91-7871-629-2.

A. Helmersson: Methods for robust gain scheduling. Thesis No. 406, 1995. ISBN 91-7871-628-4.

P. Lindskog: Methods, algorithms and tools for system identification based on prior knowledge. Thesis No. 436, 1996. ISBN 91-7871-424-8.

J. Gunnarsson: Symbolic methods and tools for discrete event dynamic systems. Thesis No. 477, 1997. ISBN 91-7871-917-8.

M. Jirstrand: Constructive methods for inequality constraints in control. Thesis No. 527, 1998. ISBN 91-7219-187-2.

U. Forssell: Closed-loop identification: Methods, theory, and applications. Thesis No. 566, 1999. ISBN 91-7219-432-4.

A. Stenman: Model on demand: Algorithms, analysis and applications. Thesis No. 571, 1999. ISBN 91-7219-450-2.

N. Bergman: Recursive Bayesian estimation: Navigation and tracking applications. Thesis No. 579, 1999. ISBN 91-7219-473-1.

K. Edström: Switched bond graphs: Simulation and analysis. Thesis No. 586, 1999. ISBN 91-7219-493-6.

M. Larsson: Behavioral and structural model based approaches to discrete diagnosis. Thesis No. 608, 1999. ISBN 91-7219-615-5.

F. Gunnarsson: Power control in cellular radio systems: Analysis, design and estimation. Thesis No. 623, 2000. ISBN 91-7219-689-0.

V. Einarsson: Model checking methods for mode switching systems. Thesis No. 652, 2000. ISBN 91-7219-836-2.

M. Norrlöf: Iterative learning control: Analysis, design, and experiments. Thesis No. 653, 2000. ISBN 91-7219-837-0.

F. Tjärnström: Variance expressions and model reduction in system identification. Thesis No. 730, 2002. ISBN 91-7373-253-2.

J. Löfberg: Minimax approaches to robust model predictive control. Thesis No. 812, 2003. ISBN 91-7373-622-8.

J. Roll: Local and piecewise affine approaches to system identification. Thesis No. 802, 2003. ISBN 91-7373-608-2.

J. Elbornsson: Analysis, estimation and compensation of mismatch effects in A/D converters. Thesis No. 811, 2003. ISBN 91-7373-621-X.

O. Härkegård: Backstepping and control allocation with applications to flight control. Thesis No. 820, 2003. ISBN 91-7373-647-3.

R. Wallin: Optimization algorithms for system analysis and identification. Thesis No. 919, 2004. ISBN 91-85297-19-4.

D. Lindgren: Projection methods for classification and identification. Thesis No. 915, 2005. ISBN 91-85297-06-2.

R. Karlsson: Particle Filtering for Positioning and Tracking Applications. Thesis No. 924, 2005. ISBN 91-85297-34-8.

J. Jansson: Collision Avoidance Theory with Applications to Automotive Collision Mitigation. Thesis No. 950, 2005. ISBN 91-85299-45-6.

E. Geijer Lundin: Uplink Load in CDMA Cellular Radio Systems. Thesis No. 977, 2005. ISBN 91-85457-49-3.

M. Enqvist: Linear Models of Nonlinear Systems. Thesis No. 985, 2005. ISBN 91-85457-64-7.

T. B. Schön: Estimation of Nonlinear Dynamic Systems — Theory and Applications. Thesis No. 998, 2006. ISBN 91-85497-03-7.

I. Lind: Regressor and Structure Selection — Uses of ANOVA in System Identification. Thesis No. 1012, 2006. ISBN 91-85523-98-4.

J. Gillberg: Frequency Domain Identification of Continuous-Time Systems Reconstruction and Robustness. Thesis No. 1031, 2006. ISBN 91-85523-34-8.

M. Gerdin: Identification and Estimation for Models Described by Differential-Algebraic Equations. Thesis No. 1046, 2006. ISBN 91-85643-87-4.

C. Grönwall: Ground Object Recognition using Laser Radar Data – Geometric Fitting, Performance Analysis, and Applications. Thesis No. 1055, 2006. ISBN 91-85643-53-X.

A. Eidehall: Tracking and threat assessment for automotive collision avoidance. Thesis No. 1066, 2007. ISBN 91-85643-10-6.

F. Eng: Non-Uniform Sampling in Statistical Signal Processing. Thesis No. 1082, 2007. ISBN 978-91-85715-49-7.

E. Wernholt: Multivariable Frequency-Domain Identification of Industrial Robots. Thesis No. 1138, 2007. ISBN 978-91-85895-72-4.

D. Axehill: Integer Quadratic Programming for Control and Communication. Thesis No. 1158, 2008. ISBN 978-91-85523-03-0.

G. Hendeby: Performance and Implementation Aspects of Nonlinear Filtering. Thesis No. 1161, 2008. ISBN 978-91-7393-979-9.

J. Sjöberg: Optimal Control and Model Reduction of Nonlinear DAE Models. Thesis No. 1166, 2008. ISBN 978-91-7393-964-5.

D. Törnqvist: Estimation and Detection with Applications to Navigation. Thesis No. 1216, 2008. ISBN 978-91-7393-785-6.

P-J. Nordlund: Efficient Estimation and Detection Methods for Airborne Applications. Thesis No. 1231, 2008. ISBN 978-91-7393-720-7.

H. Tidefelt: Differential-algebraic equations and matrix-valued singular perturbation. Thesis No. 1292, 2009. ISBN 978-91-7393-479-4.

H. Ohlsson: Regularization for Sparseness and Smoothness — Applications in System Identification and Signal Processing. Thesis No. 1351, 2010. ISBN 978-91-7393-287-5.

S. Moberg: Modeling and Control of Flexible Manipulators. Thesis No. 1349, 2010. ISBN 978-91-7393-289-9.

J. Wallén: Estimation-based iterative learning control. Thesis No. 1358, 2011. ISBN 978-91-7393-255-4.

J. Hol: Sensor Fusion and Calibration of Inertial Sensors, Vision, Ultra-Wideband and GPS. Thesis No. 1368, 2011. ISBN 978-91-7393-197-7.

D. Ankelhed: On the Design of Low Order H-infinity Controllers. Thesis No. 1371, 2011. ISBN 978-91-7393-157-1.

C. Lundquist: Sensor Fusion for Automotive Applications. Thesis No. 1409, 2011. ISBN 978-91-7393-023-9.

P. Skoglar: Tracking and Planning for Surveillance Applications. Thesis No. 1432, 2012. ISBN 978-91-7519-941-2.

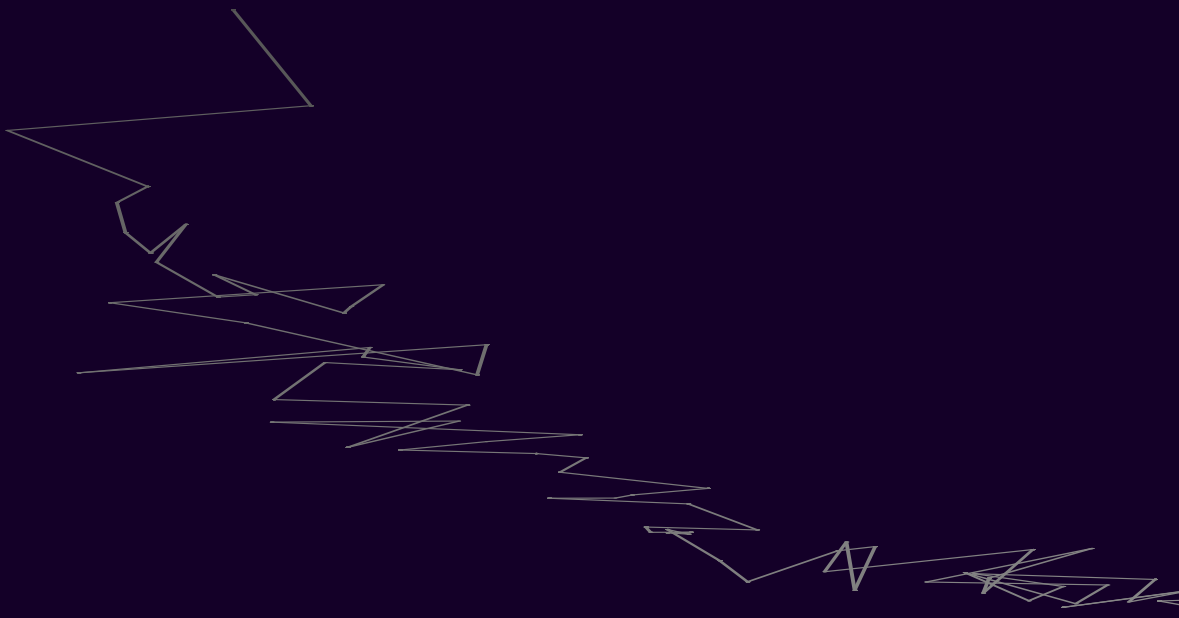
K. Granström: Extended target tracking using PHD filters. Thesis No. 1476, 2012. ISBN 978-91-7519-796-8.

C. Lyzell: Structural Reformulations in System Identification. Thesis No. 1475, 2012. ISBN 978-91-7519-800-2.

J. Callmer: Autonomous Localization in Unknown Environments. Thesis No. 1520, 2013. ISBN 978-91-7519-620-6.

D. Petersson: A Nonlinear Optimization Approach to H2-Optimal Modeling and Control. Thesis No. 1528, 2013. ISBN 978-91-7519-567-4.

Z. Sjanic: Navigation and Mapping for Aerial Vehicles Based on Inertial and Imaging Sensors. Thesis No. 1533, 2013. ISBN 978-91-7519-553-7.
F. Lindsten: Particle Filters and Markov Chains for Learning of Dynamical Systems. Thesis No. 1530, 2013. ISBN 978-91-7519-559-9.
P. Axelsson: Sensor Fusion and Control Applied to Industrial Manipulators. Thesis No. 1585, 2014. ISBN 978-91-7519-368-7.
A. Carvalho Bittencourt: Modeling and Diagnosis of Friction and Wear in Industrial Robots. Thesis No. 1617, 2014. ISBN 978-91-7519-251-2.
M. Skoglund: Inertial Navigation and Mapping for Autonomous Vehicles. Thesis No. 1623, 2014. ISBN 978-91-7519-233-8.
S. Khoshfetrat Pakazad: Divide and Conquer: Distributed Optimization and Robustness Analysis. Thesis No. 1676, 2015. ISBN 978-91-7519-050-1.
T. Ardeshiri: Analytical Approximations for Bayesian Inference. Thesis No. 1710, 2015. ISBN 978-91-7685-930-8.
N. Wahlström: Modeling of Magnetic Fields and Extended Objects for Localization Applications. Thesis No. 1723, 2015. ISBN 978-91-7685-903-2.



FACULTY OF SCIENCE AND ENGINEERING

Linköping studies in science and technology. Dissertations. No. 1754
Department of Electrical Engineering, Linköping University.

Linköping University
SE-581 83 Linköping, Sweden

www.liu.se

LINKÖPING UNIVERSITY