Accelerating Monte Carlo methods for Bayesian inference in dynamical models

Johan Dahlin
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.
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 skenend. 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.


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 Tidfelt 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
I Background

1 Introductory overview
  1.1 Examples of applications ........................................ 4
      1.1.1 Reconstructing the temperature of prehistoric Earth .. 5
      1.1.2 Rendering photorealistic images .......................... 5
  1.2 Main contribution .................................................. 7
  1.3 Thesis outline ...................................................... 9
  1.4 Publications ....................................................... 15

2 Bayesian modelling and inference ........................................ 17
  2.1 Three examples of statistical data ................................. 18
  2.2 Parametric models .................................................. 23
      2.2.1 Linear regression and generalised linear models ...... 23
      2.2.2 Autoregressive models ..................................... 25
      2.2.3 State space models ......................................... 26
      2.2.4 Mixed effect models ....................................... 28
  2.3 Computing the posterior and making decisions ..................... 29
  2.4 Non-parametric models ............................................. 36
      2.4.1 Gaussian processes ......................................... 36
      2.4.2 Dirichlet processes ....................................... 39
  2.5 Outlook and extensions ............................................ 40

3 Monte Carlo methods ..................................................... 43
  3.1 Empirical approximations .......................................... 44
  3.2 Three sampling strategies ......................................... 45
      3.2.1 Independent Monte Carlo .................................. 45
      3.2.2 Sequential Monte Carlo ................................... 48
      3.2.3 Markov chain Monte Carlo ................................ 59
  3.3 Pseudo-marginal Metropolis-Hastings .............................. 67
  3.4 Outlook and extensions ............................................ 70
### Chapter 4: Strategies for accelerating inference

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Increasing the amount of information in the data</td>
<td>74</td>
</tr>
<tr>
<td>4.2</td>
<td>Approximating the model</td>
<td>74</td>
</tr>
<tr>
<td>4.3</td>
<td>Improving the inference algorithm</td>
<td>79</td>
</tr>
<tr>
<td>4.4</td>
<td>Outlook and extensions</td>
<td>85</td>
</tr>
</tbody>
</table>

### Chapter 5: Concluding remarks

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Summary of contributions</td>
<td>88</td>
</tr>
<tr>
<td>5.2</td>
<td>Some trends and ideas for future work</td>
<td>90</td>
</tr>
<tr>
<td>5.3</td>
<td>Source code and data</td>
<td>93</td>
</tr>
</tbody>
</table>

### Notation

- Page 95

### Bibliography

- Page 99

## II Papers

### A: Getting started with PMH for inference in non-linear models

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introductory overview</td>
<td>120</td>
</tr>
<tr>
<td>1.1</td>
<td>State space models</td>
<td>120</td>
</tr>
<tr>
<td>1.2</td>
<td>Bayesian inference</td>
<td>121</td>
</tr>
<tr>
<td>2</td>
<td>Related software</td>
<td>122</td>
</tr>
<tr>
<td>3</td>
<td>Overview of the PMH algorithm</td>
<td>123</td>
</tr>
<tr>
<td>3.1</td>
<td>Constructing the Markov chain</td>
<td>123</td>
</tr>
<tr>
<td>3.2</td>
<td>Approximating the acceptance probability</td>
<td>124</td>
</tr>
<tr>
<td>4</td>
<td>Estimating the parameters in a linear Gaussian SSM</td>
<td>126</td>
</tr>
<tr>
<td>4.1</td>
<td>Implementing the particle filter</td>
<td>128</td>
</tr>
<tr>
<td>4.2</td>
<td>Numerical illustration of state inference</td>
<td>131</td>
</tr>
<tr>
<td>4.3</td>
<td>Implementing particle Metropolis-Hastings</td>
<td>133</td>
</tr>
<tr>
<td>4.4</td>
<td>Numerical illustration of parameter inference</td>
<td>135</td>
</tr>
<tr>
<td>5</td>
<td>Application example: volatility estimation OMXS30</td>
<td>137</td>
</tr>
<tr>
<td>6</td>
<td>Improving the PMH algorithm</td>
<td>141</td>
</tr>
<tr>
<td>6.1</td>
<td>Initialisation</td>
<td>141</td>
</tr>
<tr>
<td>6.2</td>
<td>Diagnosing convergence</td>
<td>142</td>
</tr>
<tr>
<td>6.3</td>
<td>Improving mixing</td>
<td>142</td>
</tr>
<tr>
<td>7</td>
<td>Outlook and conclusions</td>
<td>146</td>
</tr>
<tr>
<td>7.1</td>
<td>Improving the particle filter</td>
<td>147</td>
</tr>
<tr>
<td>7.2</td>
<td>Improving particle Metropolis-Hastings</td>
<td>148</td>
</tr>
<tr>
<td>7.3</td>
<td>Additional software implementations</td>
<td>149</td>
</tr>
</tbody>
</table>

### Bibliography

- Page 151

### B: PMH using gradient and Hessian information

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>158</td>
</tr>
<tr>
<td>2</td>
<td>Particle Metropolis-Hastings</td>
<td>159</td>
</tr>
<tr>
<td>2.1</td>
<td>MH sampling with unbiased likelihoods</td>
<td>159</td>
</tr>
<tr>
<td>2.2</td>
<td>Constructing PMH1 and PMH2</td>
<td>161</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3</td>
<td>Properties of the PMH1 and PMH2 proposals</td>
<td>162</td>
</tr>
<tr>
<td>3</td>
<td>Estimation of the likelihood, gradient, and Hessian</td>
<td>162</td>
</tr>
<tr>
<td>3.1</td>
<td>Auxiliary particle filter</td>
<td>163</td>
</tr>
<tr>
<td>3.2</td>
<td>Estimation of the likelihood</td>
<td>164</td>
</tr>
<tr>
<td>3.3</td>
<td>Estimation of the gradient</td>
<td>164</td>
</tr>
<tr>
<td>3.4</td>
<td>Estimation of the Hessian</td>
<td>165</td>
</tr>
<tr>
<td>3.5</td>
<td>Regularisation of the estimate of the Hessian</td>
<td>167</td>
</tr>
<tr>
<td>3.6</td>
<td>Resulting SMC algorithm</td>
<td>168</td>
</tr>
<tr>
<td>4</td>
<td>Numerical illustrations</td>
<td>168</td>
</tr>
<tr>
<td>4.1</td>
<td>Estimation of the log-likelihood and the gradient</td>
<td>169</td>
</tr>
<tr>
<td>4.2</td>
<td>Burn-in and scale-invariance</td>
<td>169</td>
</tr>
<tr>
<td>4.3</td>
<td>The mixing of the Markov chains at stationarity</td>
<td>173</td>
</tr>
<tr>
<td>4.4</td>
<td>Parameter inference in a Poisson count model</td>
<td>175</td>
</tr>
<tr>
<td>4.5</td>
<td>Robustness in the lag and step size</td>
<td>178</td>
</tr>
<tr>
<td>5</td>
<td>Discussion and future work</td>
<td>178</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>181</td>
</tr>
</tbody>
</table>

### C Quasi-Newton particle Metropolis-Hastings

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>188</td>
</tr>
<tr>
<td>2</td>
<td>Particle Metropolis-Hastings</td>
<td>189</td>
</tr>
<tr>
<td>3</td>
<td>Proposal for parameters</td>
<td>190</td>
</tr>
<tr>
<td>3.1</td>
<td>Zeroth and first-order proposals (PMHe/1)</td>
<td>190</td>
</tr>
<tr>
<td>3.2</td>
<td>Second-order proposal (PMH2)</td>
<td>191</td>
</tr>
<tr>
<td>4</td>
<td>Proposal for auxiliary variables</td>
<td>192</td>
</tr>
<tr>
<td>4.1</td>
<td>SMC-ABC algorithm</td>
<td>192</td>
</tr>
<tr>
<td>4.2</td>
<td>Estimation of the likelihood</td>
<td>193</td>
</tr>
<tr>
<td>4.3</td>
<td>Estimation of the gradient of the log-posterior</td>
<td>193</td>
</tr>
<tr>
<td>5</td>
<td>Numerical illustrations</td>
<td>194</td>
</tr>
<tr>
<td>5.1</td>
<td>Linear Gaussian SSM</td>
<td>194</td>
</tr>
<tr>
<td>5.2</td>
<td>Modelling the volatility in coffee futures</td>
<td>197</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions</td>
<td>197</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>206</td>
</tr>
</tbody>
</table>

### D Accelerating pmMH using correlated likelihood estimators

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>212</td>
</tr>
<tr>
<td>2</td>
<td>Introducing correlation into the auxiliary variables</td>
<td>213</td>
</tr>
<tr>
<td>3</td>
<td>Theoretical analysis</td>
<td>216</td>
</tr>
<tr>
<td>3.1</td>
<td>Setting up the model</td>
<td>216</td>
</tr>
<tr>
<td>3.2</td>
<td>Analysis by discretisation of the state space</td>
<td>217</td>
</tr>
<tr>
<td>3.3</td>
<td>Tuning the CN proposal for the univariate case</td>
<td>218</td>
</tr>
<tr>
<td>4</td>
<td>Numerical illustrations</td>
<td>220</td>
</tr>
<tr>
<td>4.1</td>
<td>Gaussian IID model</td>
<td>220</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Stochastic volatility model with leverage</td>
<td>222</td>
</tr>
<tr>
<td>5</td>
<td>Conclusions and future work</td>
<td>225</td>
</tr>
<tr>
<td>A</td>
<td>Implementation details</td>
<td>226</td>
</tr>
<tr>
<td>A.1</td>
<td>Gaussian IID model</td>
<td>226</td>
</tr>
<tr>
<td>A.2</td>
<td>Stochastic volatility model with leverage</td>
<td>226</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>229</td>
</tr>
<tr>
<td>E</td>
<td>Approximate Bayesian inference for models with intractable likelihoods</td>
<td>233</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>236</td>
</tr>
<tr>
<td>2</td>
<td>An intuitive overview of GPO-ABC</td>
<td>238</td>
</tr>
<tr>
<td>3</td>
<td>Estimating the log-posterior</td>
<td>240</td>
</tr>
<tr>
<td>3.1</td>
<td>Particle filtering with approximate Bayesian computations</td>
<td>241</td>
</tr>
<tr>
<td>3.2</td>
<td>The estimator and its statistical properties</td>
<td>243</td>
</tr>
<tr>
<td>4</td>
<td>Constructing the surrogate of the log-posterior</td>
<td>244</td>
</tr>
<tr>
<td>4.1</td>
<td>Gaussian process prior</td>
<td>244</td>
</tr>
<tr>
<td>4.2</td>
<td>Acquisition function</td>
<td>246</td>
</tr>
<tr>
<td>5</td>
<td>The GPO-ABC algorithm</td>
<td>247</td>
</tr>
<tr>
<td>5.1</td>
<td>Initialisation and convergence criteria</td>
<td>247</td>
</tr>
<tr>
<td>5.2</td>
<td>Extracting the Laplace approximation</td>
<td>248</td>
</tr>
<tr>
<td>5.3</td>
<td>Convergence properties</td>
<td>249</td>
</tr>
<tr>
<td>6</td>
<td>Numerical illustrations and real-world applications</td>
<td>249</td>
</tr>
<tr>
<td>6.1</td>
<td>Stochastic volatility model with Gaussian log-returns</td>
<td>249</td>
</tr>
<tr>
<td>6.2</td>
<td>Stochastic volatility model with α-stable log-returns</td>
<td>251</td>
</tr>
<tr>
<td>6.3</td>
<td>Modelling price dependencies between oil futures</td>
<td>253</td>
</tr>
<tr>
<td>7</td>
<td>Conclusions</td>
<td>260</td>
</tr>
<tr>
<td>A</td>
<td>Implementation details</td>
<td>261</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>263</td>
</tr>
<tr>
<td>F</td>
<td>Hierarchical Bayesian approaches for robust inference in ARX models</td>
<td>267</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>270</td>
</tr>
<tr>
<td>2</td>
<td>Hierarchical Bayesian ARX models</td>
<td>271</td>
</tr>
<tr>
<td>2.1</td>
<td>Student’s t distributed innovations</td>
<td>271</td>
</tr>
<tr>
<td>2.2</td>
<td>Parametric model order</td>
<td>271</td>
</tr>
<tr>
<td>2.3</td>
<td>Automatic relevance determination</td>
<td>272</td>
</tr>
<tr>
<td>3</td>
<td>Markov chain Monte Carlo</td>
<td>273</td>
</tr>
<tr>
<td>4</td>
<td>Posteriors and proposal distributions</td>
<td>274</td>
</tr>
<tr>
<td>4.1</td>
<td>Model order</td>
<td>274</td>
</tr>
<tr>
<td>4.2</td>
<td>ARX coefficients</td>
<td>275</td>
</tr>
<tr>
<td>4.3</td>
<td>ARX coefficients variance</td>
<td>275</td>
</tr>
<tr>
<td>4.4</td>
<td>Latent variance variables</td>
<td>276</td>
</tr>
<tr>
<td>4.5</td>
<td>Innovation scale parameter</td>
<td>277</td>
</tr>
<tr>
<td>4.6</td>
<td>Innovation DOF</td>
<td>277</td>
</tr>
<tr>
<td>5</td>
<td>Numerical illustrations</td>
<td>277</td>
</tr>
<tr>
<td>5.1</td>
<td>Average model performance</td>
<td>277</td>
</tr>
<tr>
<td>5.2</td>
<td>Robustness to outliers and missing data</td>
<td>280</td>
</tr>
<tr>
<td>5.3</td>
<td>Real EGG data</td>
<td>282</td>
</tr>
</tbody>
</table>
Contents

6 Conclusions and Future work ........................................ 282
Bibliography ................................................................. 285

G Bayesian inference for mixed effects models with heterogeneity 287
1 Introduction ............................................................... 290
2 Bayesian mixture modelling .......................................... 291
  2.1 Infinite mixture model using a Dirichlet process ............ 291
  2.2 Finite mixture model .............................................. 293
3 Sampling from the posterior ........................................... 294
  3.1 Prior distributions ................................................ 294
  3.2 Gibbs sampling .................................................... 295
  3.3 Conditional posteriors ............................................ 296
4 Numerical illustrations ................................................ 298
  4.1 Mixture of Gaussians ............................................ 298
  4.2 Mixed effects model with synthetic data ...................... 300
  4.3 Mixed effects model with sleep deprivation data .......... 302
5 Conclusions .............................................................. 304
A Implementation details ................................................ 306
  A.1 Mixture of Gaussians ............................................ 306
  A.2 Mixed effects model with synthetic data ...................... 306
  A.3 Mixed effects model with sleep deprivation data .......... 306
Bibliography ................................................................. 307

H Robust Input design for non-linear dynamical models 309
1 Introduction .............................................................. 312
2 Problem formulation .................................................... 314
3 Describing the set of stationary processes ....................... 316
4 Estimation of the Fisher information matrix ...................... 319
  4.1 Estimating the score function .................................. 319
  4.2 The resulting estimator .......................................... 321
5 Final input design method ............................................. 324
6 Input signal generation ................................................ 324
7 Numerical illustrations ................................................ 326
  7.1 Accuracy of information matrix estimation ................. 326
  7.2 Input design for the LGSS model .............................. 328
  7.3 Input design for a non-linear model ........................... 332
8 Conclusions .............................................................. 332
Bibliography ................................................................. 333
Part I

Background
Modelling of dynamical systems is an integrative 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

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

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.

1.1 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.

1.1.1 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).

1.1.2 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.
1.2 Main contribution

(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 plane. 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).

1.2 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 PMH 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).

1.3 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


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


which is a development of the two earlier contributions:


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.
1.3 Thesis outline

Paper C

Paper C of this thesis is an edited version of


Source code and data: https://github.com/compsops/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 pmmh 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


Source code and data: https://github.com/compsops/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


which is the development of the two earlier contributions:


**Source code and data:** https://github.com/compos/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


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


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,


which is a development of the earlier contribution


**Summary:** Input design is an important sub-field of system identification. Its main aim is 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.
1.4 Publications

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


Other publications not included in the thesis are:


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 Frequentist 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 electronic 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.
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.

### 2.1 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 an 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 un-
employment 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 \textit{nairu} and vice versa. Estimating the \textit{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 centralbyrå\textsuperscript{1} (\textsc{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.

\textit{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 $y = \{ (y_{it})_{t=1}^{T} \}_{i=1}^{n}$ 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 \textit{Sozio-oekonomisches Panel} (\textsc{soef}\textsuperscript{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 \textit{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.

\textbf{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

\textsuperscript{1}See http://www.scb.se/en/ for more information.

\textsuperscript{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.
nominate 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 Speth 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 (i) or conservative (o) 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.

2.2 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 $\Theta$ 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.

2.2.1 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

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

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., \( \mathbb{E}[x_{ij} e_j] = 0 \) for every \( i \) and \( j \).

In this model, the parameters are given by \( \theta = \{ \beta_0, \sigma \} \), 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 \( \theta \) from data is to make use of the least squares (L.S) approach. The main objective in L.S is to minimise the squared prediction error, i.e.,

\[
\hat{\beta}_{L.S} = \underset{\beta \in \Theta}{\text{argmin}} \|y - X \beta\|_2^2, \tag{2.2}
\]

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

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

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

\[
\hat{\beta}_{L.S} = (X^T X)^{-1}(X^T y), \tag{2.3}
\]

which we refer to as the normal equations. The noise variance \( \sigma^2 \) can be estimated directly using the standard sample estimator by

\[
s^2 = \hat{\sigma}^2 = \frac{1}{n-p-1} \sum_{i=1}^{n} (y_i - X_i \hat{\beta}_{L.S})^2, \tag{2.4}
\]

where \( X_i \) denotes row \( i \) of the matrix \( X \). It is possible to show that the L.S 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

\[
\hat{\beta}_{\text{elastic}} = \underset{\beta \in \Theta}{\text{argmin}} \|y - X \beta\|_2^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2, \tag{2.5}
\]

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) \( L_1 \cdot \text{RLS} \) when \( \lambda_2 = 0 \) and (ii) \( L_2 \cdot \text{RLS} \) 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 it 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 $E[y_i] = h^{-1}(\eta_i)$. 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 $(0, 1)$.

### 2.2.2 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

$$ y_t | y_{t-1}, \ldots, y_{t-p} \sim N(y_t; \mu + \sum_{k=1}^{p} \phi_k (y_{t-k} - \mu), \sigma^2), $$

where $y_t$ denotes the observation at time $t$. The model is specified by the parameters $\theta = \{ \mu, \phi_1, \ldots, \phi_p, \sigma \}$ 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 $\phi$ 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 $\theta$ 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

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

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 in a Bayesian setting. 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).

### 2.2.3 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 of 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

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

where $\theta$ denotes the parameters of the model. Here, we assume that the model can be described by probability density functions (pdfs) denoted $\mu_\theta$, $f_\theta$ and $g_\theta$.

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 $\theta$. 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

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

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

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

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

$$\mu(x_{t-1}) \triangleq \alpha \left[ 1 + \exp \left( -x_{t-1} \right) \right]^{-1}, \quad \sigma^{-2}(x_{t-1}, u_{t-1}) \triangleq 1 + \exp \left[ -|u_{t-1} - x_{t-1}| \right].$$

The parameters of this Phillips curve model are $\theta = \{\alpha, \beta, \phi, \sigma_c\}$. 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 $\sigma_c > 0$. The parameter $\beta$ 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

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

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

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

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.
2.2.4 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

\[ y_{it} = \alpha_t x_{it} + \beta_i z_{it} + e_{it}, \]

where \( e_{it} \) denotes some error term, e.g., a Gaussian i.i.d 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

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

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

\[ u_{it} = \alpha_{t,1} x_i \begin{bmatrix} -1 \end{bmatrix} + e_{it}, \quad y_{it} = \begin{cases} 1, & u_{it} > 0, \\ 0, & u_{it} \leq 0, \end{cases} \]

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 \( \alpha_t \).

We return to this model in Example 3.8 on page 63.
2.3 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 $\theta$ 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 $p(\theta|y)$, where $y$ denotes some data. This procedure is known as the prior-posterior update and is carried out using Bayes’ theorem given by

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \propto p_\theta(y)p(\theta).$$  

(2.10)

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

In maximum likelihood inference, it is assumed that the parameter $\theta$ is fixed and that the uncertainty comes from the data. The point estimate of $\theta$ 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.

\textsc{r.l.s} 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 \textsc{r.l.s} is to view it as Bayesian linear regression with specific choices of prior distributions. Two choices for promoting sparsity are given by

\[ p(\beta_j) = \mathcal{L}(\beta_j; 0, \sigma), \quad p(\beta_j) = \mathcal{N}(\beta_j; 0, \sigma^2), \]

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 1.1-\textsc{r.l.s} (\textsc{lasso}) and to 1.2-\textsc{r.l.s} (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 4.1.2 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

\[ p(y | \theta) = \prod_{i=1}^{n} \prod_{t=1}^{T} p_{iit}^{y_{it}} (1 - p_{iit})^{1-y_{it}}, \]

with the success probability \( p_{iit} \) is given by

\[ p_{iit} = \Phi \left( [\alpha_{t,1} \quad \alpha_{t,2}] \begin{bmatrix} -1 \\ x_i \end{bmatrix} \right), \]

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 (2.16), we obtain directly that

\[ p(x_i | y, \alpha, u) \propto \mathcal{N}(x_i; 0, 1) \prod_{t=1}^{T} \mathcal{N}(u_{iit}; -\alpha_{t,1} + \alpha_{t,2} x_i, 1), \]

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

\[ p(x_i | y, \alpha, u) = \mathcal{N} \left( x_i; \sum_{\text{post}}^{-1} \sum_{t=1}^{T} \alpha_{t,2} (u_{iit} + \alpha_{t-1}), \sum_{\text{post}}^{-1} \right), \quad \sum_{\text{post}} = 1 + \sum_{t=1}^{T} \alpha_{t,2}^2. \]

It is possible to also compute conditional posteriors for \( u \) and \( \alpha \), 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

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

(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

\[ p_\theta(y_t | y_{1:t-1}) = \int_{\mathcal{X}} \theta(y_t | x_t) f_\theta(x_t | x_{t-1}) p_\theta(x_{t-1} | y_{1:t-1}) \, dx_{t-1:t}, \]

(2.12)

which follows from the Markov property of the ssm. However, we cannot evaluate this integral in closed form for most sssms 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

\[ x_t | x_{t-1} \sim \mathcal{N}(x_t; \mu + \phi(x_{t-1} - \mu), \sigma^2_x), \quad y_t | x_t \sim \mathcal{N}(y_t; x_t, \sigma^2_y), \]

(2.13)

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

\[ \pi_t(x_t) = \frac{\theta(y_t | x_t)}{\int_{\mathcal{X}} \theta(y_t | x_t) \pi_{t-1}(x_{t-1}) \, dx_{t-1}}, \]

(2.14)

for \( 0 < t \leq T \). From the structure of the lGSS model, we assume that the prior distribution can be denoted by \( \pi_{t-1}(x_{t-1}) = \mathcal{N}(x_{t-1}; \tilde{x}_{t-1|t-1}, P_{t-1|t-1}) \). Here, \( \tilde{x}_{t-1|t-1} \) 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

\[ \hat{x}_{t|t-1} = \mu + \phi(\hat{x}_{t-1|t-1} - \mu), \quad P_{t|t-1} = \phi^2 P_{t-1|t-1} + \sigma^2_x. \]

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

\[ \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - \hat{x}_{t|t-1}), \quad P_{t|t} = P_{t|t-1} - P_{t|t-1} K_t, \]

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

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

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 \( P_{1|0} = \sigma^2_e (1 - \phi^2)^{-1} \) in this particular model.

We continue by presenting some useful quantities connected with the likelihood, which are used 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

\[
S(\theta') = \nabla \log p_\theta(y)|_{\theta = \theta'}.
\]  

(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, \( S(\theta^*) = 0 \). 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

\[
\mathcal{J}(\theta') = -\nabla^2 \log p_\theta(y)|_{\theta = \theta'}.
\]  

(2.16)

The observed information matrix can be seen as a measure of the total amount of information available regarding \( \theta \) 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

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

(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 \( \theta \) 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 \( \theta \) 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
Bayesian modelling and inference

<table>
<thead>
<tr>
<th>Loss function</th>
<th>Bayes point estimator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>$L(\theta, \delta) =</td>
</tr>
<tr>
<td>Quadratic</td>
<td>$L(\theta, \delta) = (\theta - \delta)^2$</td>
</tr>
<tr>
<td>$\sigma$-1</td>
<td>$L(\theta, \delta) = I(\theta = \delta)$</td>
</tr>
</tbody>
</table>

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

account in the point-estimate. Consider a loss function $L : \Theta \times \Theta \rightarrow \mathbb{R}_+$, 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

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

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,

$$\delta^*(y) = \arg\min_{\delta(y) \in \Theta} \rho(p(\theta), \delta | y).$$

**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

$$\arg\min_{\delta(y) \in \Theta} \int_{\Theta} (\hat{\theta} - \theta)^2 p(\theta | y) \, d\theta.$$

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

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

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

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

where the solution is given by

$$\hat{\theta} = \mathbb{E}_y[\theta] = \int_{\Theta} \theta p(\theta | y) \, d\theta.$$  \hspace{1cm} (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 $\sigma$-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 correspond 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 \( \theta \) 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

\[
\hat{\theta} \xrightarrow{a.s.} \theta^*,
\]

where \( \theta^* \) 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

\[
\sqrt{n}(\hat{\theta} - \theta^*) \xrightarrow{d} \mathcal{N}(0, \mathcal{I}^{-1}(\theta^*)),
\]

when \( n \to \infty \). This follows from a second-order Taylor expansion of the log-likelihood around \( \theta^* \). 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 \( \text{RLS} \) in Remark 2.5, where the estimates are biased but sometimes enjoy a much smaller variance compared with the \( \text{LS} \) solution.
2.4 Non-parametric models

Bayesian non-parametrics (BNPs; Hjort et al., 2010) is an active research field in machine learning and computational statistics. The models introduced 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 BNP s 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 BNP s 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 BNP s is provided by Fox (2009). DPs are employed to model piece-wise affine systems by Wägberg 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 a unit of genetic information and inferring these from data is important to understand genetic variations in populations of individuals.

2.4.1 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 $\kappa$ denote the mean function and the covariance function (kernel), respectively. These two functions fully specifies the GP and can be defined by

$$m(x) = \mathbb{E}[f(x)], \quad \kappa(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))^\top],$$

for some function $f : \mathbb{R}^p \to \mathbb{R}$ that we would like to model by the GP. Both $m$ and $\kappa$ 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

$$y_i = f(x_i) + \sigma_e e_i,$$

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

$$f \sim \text{GP}(m, \kappa),$$

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 $D = \{x, y\} = \{x_i, y_i\}_{i=1}^n$. The resulting predictive distribution evaluated at some test point $x_*$ is given by a Gaussian distribution with an updated mean and covariance function computed by

$$f(x_*)|D \sim \mathcal{N}(x_*; \mu_f(x_*|D), \sigma_f^2(x_*|D)),$$

$$\mu_f(x_*|D) = \kappa_*^T \left[ \kappa(x, x) + \sigma_e^2 I_n \right]^{-1} y,$$

$$\sigma_f^2(x_*|D) = \kappa(x_*, x_*) - \kappa_*^T \left[ \kappa(x, x) + \sigma_e^2 I_n \right]^{-1} \kappa_* + \sigma_e^2.$$

Here, we introduce $\kappa_* = \kappa(x_*, x)$ 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 $\mu_f$ is plotted as a solid line. The confidence intervals are computed by using the variance in the predictive posterior $\sigma_f^2$. 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 $l$. 
2.4.2 Dirichlet processes

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

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

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

$$\left(G(A_1), G(A_2), \ldots, G(A_N)\right) \sim \mathcal{D}(\alpha G_0(A_1), \alpha G_0(A_2), \ldots, \alpha G_0(A_N)), \quad (2.25)$$

for any finite (measurable) partition $A_{1:N}$ of $\Omega$. 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

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

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

$$p(\theta_* | \theta_{1:n}) = \int G(\theta_*) p(G | \theta_{1:n}) dG,$$

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

$$\theta_* | \theta_{1:n} \sim \frac{\alpha}{\alpha + n} G_0 + \frac{1}{\alpha + n} \sum_{i=1}^{n} n_i \delta_{\theta_i}. \quad (2.26)$$

Here, $n_i$ denotes the number of parameters that are identical to $\theta_i$, 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 $\theta_i$ 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 DP 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

$$G \sim \mathcal{DPM}(\alpha, G_0), \quad \theta_k \sim G, \quad x_k \sim p(\cdot | \theta_k), \quad \text{(2.27)}$$

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.

### 2.5 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).
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.

### 3.1 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

$$\pi[\varphi] \triangleq \mathbb{E}_\pi[\varphi(x)] = \int_{\mathcal{X}} \varphi(x) \pi(x) \, dx,$$

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

$$\hat{\pi}^N_{MC}(dx) = \sum_{i=1}^{N} \delta_{\varphi(x)}(dx),$$

using the particles $\{x^{(i)}\}_{i=1}^{N}$ generated from the target distribution $\pi$. Here, $\delta_{\varphi}(dx)$ 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

$$\hat{\pi}^N_{MC}[\varphi] \triangleq \int_{\mathcal{X}} \varphi(x) \hat{\pi}(dx) = \frac{1}{N} \sum_{i=1}^{N} \varphi(x^{(i)}),$$

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.,

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

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

$$\sqrt{N}(\hat{\pi}_{\text{MC}}^N[\varphi] - \pi[\varphi]) \xrightarrow{d} \mathcal{N}(0, \sigma_{\text{MC}}^2), \quad \sigma_{\text{MC}}^2 \leq \mathbb{E}_{\pi}[\varphi] < \infty,$$

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).

### 3.2 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.

#### 3.2.1 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

$$x^{(i)} = [a x^{(i-1)} + b] \pmod{m},$$

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

$$x^{(i)} = -\frac{\log(\nu^{(i)})}{\lambda},$$

where $\nu^{(i)}$ 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

$$\pi[\varphi] = \int_{\mathcal{X}} \varphi(x) \pi(x) \, dx = \int_{\mathcal{X}} \varphi(x) \frac{\pi(x)}{q(x)} \triangleq w(x) \, dq(x) = q[w \varphi],$$

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

$$\hat{\pi}^N_{is} [\varphi] \triangleq \sum_{i=1}^{N} w^{(i)} \varphi(x^{(i)}), \quad (3.4)$$
3.2 Three sampling strategies

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 \( w^{(i)} = w(x^{(i)}) \). 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., \( \text{supp}(\varphi\pi) \subset \text{supp}(q) \). In that case, the estimator (3.4) inherits all of the properties from the vanilla Monte Carlo estimator, i.e., it is unbiased, consistent and asymptotically Gaussian. However, the asymptotic variance can be computed by

\[
\sigma^2_{IS} = \int_{\mathcal{X}} \frac{(\varphi(x)\pi(x))^2}{q(x)} \, dx - \pi[\varphi]^2 = \int_{\mathcal{X}} \frac{(\varphi(x)\pi(x) - \pi[\varphi]q(x))^2}{q(x)} \, dx.
\]

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 \( \pi(x) = \gamma(x)Z^{-1} \) where the normalisation constant \( Z \) 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

\[
\tilde{w}(x) = \frac{\gamma(x)}{q(x)}.
\]

The resulting estimator is given by

\[
\hat{\pi}_{SNIS}^N[\varphi] = \frac{1}{N} \sum_{i=1}^{N} \frac{\tilde{w}^{(i)}}{\sum_{j=1}^{N}\tilde{w}^{(j)}} \varphi(x^{(i)}),
\tag{3.5}
\]

which is known as the self-normalised importance sampling (\textit{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 \( \mathcal{O}(N^{-1}) \).

### 3.2.2 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 \( \pi_t(x_{0:t}) = \gamma_t(x_{0:t})Z_t^{-1} \), where \( \gamma_t(x_{0:t}) \) denotes the un-normalised target and \( Z_t \) denotes the normalisation constant. Here, we assume that it is possible evaluate the un-normalised target point-wise but the
3.2 Three sampling strategies

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 $\pi_t(x_{\mathcal{O}_t})$ 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 obtained 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 \( \phi_t \) 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

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

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

\[
\omega_t(x_{0:t}) = \frac{\gamma_t(x_{0:t})}{q_t(x_{0:t})} = \frac{\gamma_{t-1}(x_{0:t-1})}{q_{t-1}(x_{0:t-1})} \frac{\gamma_t(x_{0:t})}{\gamma_{t-1}(x_{0:t-1})} \frac{q_{t-1}(x_{t-1})}{q_t(x_{t-1})}.
\]

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) \( \{x^{(i)}_t\}^{N}_{i=1} \) and their corresponding self-normalised importance weights \( \{\omega^{(i)}_t\}^{N}_{i=1} \). From this system, we can construct an empirical approximation of \( \pi_t(x_{0:t}) \) by

\[
\hat{\pi}^N_{t, SMC}(dx_{0:t}) = \sum_{i=1}^{N} \omega_t^{(i)} \delta_{x^{(i)}_t}(dx_{0:t}), \quad (3.6)
\]
together with an estimator in analogue with (3.4) given by
\[
\tilde{\pi}_{t,\text{SMC}}^N[\varphi] = \sum_{i=1}^{N} w_t^{(i)} \varphi(x_{0:t}^{(i)}),
\]
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 discards 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
\[
\mathbb{P}(a_t^{(i)} = j) = w_{t-1}^{(j)}, \quad j = 1, \ldots, N,
\]
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
\[
x_t^{(i)} \sim q_t(x_t \mid x_{t-1}^{(i)}), \quad x_{0:t}^{(i)} \triangleq \{x_{0:t-1}^{(i)}, x_t^{(i)}\},
\]
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

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

The resulting un-normalised and normalised weights are computed by

$$\tilde{w}_t^{(i)} = W_\theta(x_t^{(i)}, x_{0:t-1}^{(i)}), \quad w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^N \tilde{w}_t^{(j)}},$$

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 $Z$ for the target by making use of the un-normalised importance weights $\tilde{w}_t^{(i)}$, 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., $p_t(x_{0:t}) = p_\theta(x_{0:t} | y_{1:t})$. We can compute the mean of this distribution at time $t$ by using $\varphi(x_t) = x_t$ in (3.7),

$$\hat{x}_{t|t} \doteq \mathbb{E}_{t, \text{SMC}}[x_t] = \int_{x_{t|t+1}} x_t \tilde{p}_t^N(x_{0:t} | y_{1:t}) \, dx_{0:t} = \sum_{i=1}^N \tilde{w}_t^{(i)} x_t^{(i)},$$

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

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

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

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).
3.2 Three sampling strategies

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

$$\tilde{P}_\theta^N(y_{1:t-1}) = \int_{\mathcal{X}} \theta(y_t | x_t) \tilde{P}_\theta^N(x_t | y_{1:t-1}) \, dx_t = \frac{1}{N} \sum_{i=1}^N \omega_t^{(i)}.$$ 

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

$$\tilde{P}_\theta^N(y_{1:T}) = \frac{1}{NT+1} \sum_{i=1}^T \sum_{j=1}^N \omega_t^{(i)}.$$ (3.12)

We refer the interested reader to Section 4.4 (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 $\pi_t(x_t) \equiv p_\theta(x_t | y_{1:t})$. 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 $\pi_T(x_T) \equiv p_\theta(x_T | y_{1:T})$ 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 $\pi_T(x_{1:T}) \equiv p_\theta(x_{1:T} | y_{1:T})$. 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 $\pi_{t-D}(x_{t-D:t}) \equiv p_\theta(x_{t-D:t} | y_{1:t})$ 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 NAIRO using data from Sweden. This corresponds to the proposal and weight function given by

$$q_t(x_t | x_{0:t-1}) = \mathcal{N}(x_t; \phi x_{t-1} + \mu(x_{t-1}), \sigma^2_\phi(x_{t-1}, u_{t-1})),$$

$$W_\theta(x_t, x_{0:t}) = \mathcal{N}(y_t; y_{t-1} + \beta(u_t - x_t), \sigma^2),$$

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

<table>
<thead>
<tr>
<th></th>
<th>( N = 50 )</th>
<th>( N = 100 )</th>
<th>( N = 250 )</th>
<th>( N = 500 )</th>
<th>( N = 1,000 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>-45.90</td>
<td>-45.89</td>
<td>-45.88</td>
<td>-45.88</td>
<td>-45.89</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.15</td>
<td>0.11</td>
<td>0.06</td>
<td>0.05</td>
<td>0.03</td>
</tr>
</tbody>
</table>

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.,

\[
\lim_{N \to \infty} \hat{\pi}^N_{t,\text{SMC}} = \pi_t, \quad \hat{\pi}^N_{t,\text{SMC}}[\varphi] \xrightarrow{a.s.} \pi_t[\varphi],
\]

where the first property holds almost surely (a.s.) and the second property holds for any integrable test function \( \varphi \) when \( N \to \infty \). However, note that \( \hat{\pi}^N_{t,\text{SMC}}[\varphi] \) 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

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

when using multinomial resampling and where \( \sigma^2_{\text{SMC}} \) 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

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

\(^1\)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 \( \hat{x}_{1|t} \) 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

$$
\sqrt{N}[p_0(y_{1:T}) - \hat{p}_0^N(y_{1:T})] \xrightarrow{d} N(0, \sigma^2_T),
$$

for some asymptotic variance $\sigma^2_T$, 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

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

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 $f_0(x_{t+1} | x_t)$ and $g_\theta(y_t | x_{\gamma})$ due to the Markov property. The resulting expectation can be expressed by

$$
\pi_T[S_\theta] = \sum_{t=1}^T \int_{X^2} \xi_{\theta,t}(x_{t-1:t}) p_\theta(x_{t-1:t} | y_{1:T}) \, dx_{t-1:t},
$$

where $p_\theta(x_{t-1:t} | y_{1:T})$ 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).
3.2 Three sampling strategies

**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

\[ S(\theta') = \int \left[ \nabla \log \hat{p}_\theta(x_{0:T}, y_{1:T}) \right] \hat{p}_\theta(x_{0:T} | y_{1:T}) \, dx_{0:T}, \]

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

\[ \log \hat{p}_\theta(x_{0:T}, y_{1:T}) = \log \mu(x_0) + \sum_{t=1}^{T} \left[ \log f_\theta(x_t | x_{t-1}) + \log g_\theta(y_t | x_t) \right]. \quad (3.18) \]

This results in the additive functional

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

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

\[ J(\theta') = \left[ S(\theta') \right]^2 - \left[ \nabla^2 \hat{p}_\theta(y_{1:T}) \big|_{\theta = \theta'} \right] \left[ \hat{p}_\theta(y_{1:T}) \right]^{-1}, \quad (3.20) \]

where the second term can be expressed as

\[ \left[ \nabla^2 \hat{p}_\theta(y_{1:T}) \big|_{\theta = \theta'} \right] \left[ \hat{p}_\theta(y_{1:T}) \right]^{-1} = \int \left[ \nabla \log \hat{p}_\theta(x_{0:T}, y_{1:T}) \big|_{\theta = \theta'} \right] \hat{p}_\theta(x_{0:T} | y_{1:T}) \, dx_{0:T} \]

\[ + \int \left[ \nabla^2 \log \hat{p}_\theta(x_{0:T}, y_{1:T}) \big|_{\theta = \theta'} \right] \hat{p}_\theta(x_{0:T} | y_{1:T}) \, dx_{0:T}. \]

Note that the first term in (3.20) is the square of the score function, which can be estimated by (3.19). The term \( \mathbb{E}[\nabla^2 \log \hat{p}_\theta(x_{0:T}, y_{1:T}) | y_{1:T}] \) 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 \( \phi \). 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 \( \phi \).

Here, it could be possible to make use of a direct optimisation method for estimating \( \phi \) 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 \( \phi \) (lower) in the Phillips curve model computed by the bPF and FL particle smoothing when varying \( \phi \). 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.
3.2.3 Markov chain Monte Carlo

Another approach for sampling from complicated target distributions is to make use of \textsc{mcmc} algorithms. The main idea is to construct a Markov chain with the target as its stationary distribution. An advantage of \textsc{mcmc} compared to e.g. importance sampling is that it can be easier to find a good proposal distribution for an \textsc{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 \textsc{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

\[
\mathbb{P}[x_k \in A | x_{2:k-1}] = \mathbb{P}[x_k \in A | x_{k-1}] = \int_A R(x_{k-1}, dx_k) = \int_A R(x_{k-1}, x_k) dx_k,
\]

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 \( R(x_{k-1}, dx_k) \) 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 \( \mu \) is invariant to the Markov chain if

\[
\mu(x_k) = \int_{\mathcal{X}} \mu(x_{k-1}) R(x_{k-1}, x_k) dx_{k-1}, \tag{3.21}
\]

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 \( \pi \). It turns out that a sufficient condition for the chain to have a stationary distribution \( \mu \) 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.,
\[
\int_{\mathcal{X}} \mu(x_k) R(x_k, x_{k-1}) \, dx_{k-1} = \int_{\mathcal{X}} \mu(x_k) R(x_k, x_{k-1}) \, dx_{k-1} \\
= \mu(x_k) \int_{\mathcal{X}} R(x_k, x_{k-1}) \, dx_{k-1} \\
= \mu(x_k),
\]
where the invariance property \( \mu R = \mu \) is recovered. Here, the detailed balance (3.22) gives the first step and the property \( \int_{\mathcal{X}} R(x_k, x_{k-1}) \, dx_{k-1} = 1 \) is used in the third step.

Another important concept in Markov chain theory is \textit{ergodicity}, which is required later to establish certain statistical properties of \textsc{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 \( \mu_0 \). 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 \textit{irreducibility}, \textit{aperiodicity} and \textit{positive recurrence}. We say that a chain is (strongly) irreducible if
\[
\int_{\mathcal{A}} R(x_{k-1}, x_k) \, dx_k > 0,
\]
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 \textsc{mcmc} algorithms.

The remaining question is how to construct the Markov kernel \( R \) in practice such that it has some sought target distribution \( \pi \) as its stationary distribution. That is, a ergodic kernel \( R \) that fulfils detailed balance for \( \pi \). We present two different approaches: the \textsc{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.
3.2 Three sampling strategies

Metropolis-Hastings

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

Remark 3.7 (Bayesian parameter inference using the \( \text{MH} \) algorithm). In many applications, we have the parameters \( \theta \) 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 \( \text{MH} \) 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 \( \pi \) denoted by \( \{x_k\}_{k=1}^{K} \). The acceptance probability is given by

\[
\alpha(x_{k-1}, x') = 1 \land \frac{\pi(x')}{\pi(x_{k-1})} \frac{q(x_{k-1}|x')}{q(x'|x_{k-1})} = 1 \land \frac{\gamma(x')}{\gamma(x_{k-1})} \frac{q(x_{k-1}|x')}{q(x'|x_{k-1})},
\]

where \( a \land b \equiv \min(a, b) \) and when the proposal can be expressed as a density. Note that the normalisation constant \( Z \) 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 \( \text{MH} \) 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 \( \text{MH} \) sampler both allows for the Markov chain to find and for it to explore areas of high posterior probability. Hence, the \( \text{MH} \) 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 \( \text{MH} \) algorithm compared with the importance sampling algorithm.
In the MH algorithm, we can express the Markov kernel by

\[ R(x_{k-1}, dx_k) = \alpha(x_{k-1}, x_k)q(x_k | x_{k-1}) \, dx_k \]

\[ + \left[ 1 - \int_{\mathcal{X}} \alpha(x_{k-1}, z)q(z | x_{k-1}) \, dz \right] \delta_{x_{k-1}}(dx_k), \]

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

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

\[ = \pi(x_{k-1})q(x_k | x_{k-1}) \wedge \pi(x_k)q(x_{k-1} | x_k), \]

\[ = \pi(x_k)q(x_{k-1} | x_k)\alpha(x_k, x_{k-1}), \]

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

\[ q(x' | x) = q(x') = \mathcal{N}(x'; \mu, \Sigma), \quad q(x' | x) = \mathcal{N}(x '; x, \Sigma), \]

for some mean vector \( \mu \) and covariance matrix \( \Sigma \). 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 \( \Sigma \) 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 $k$:

$$x_{k,i} \sim \pi_i(x_{k,i} \mid x_{k,1}, \ldots, x_{k,i-1}, x_{k,i+1}, \ldots, x_{k-1,p}).$$

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 $\pi_i(x_i \mid \cdot)$. The resulting Markov kernel is given by

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

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.,

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

An example of partial collapsing is the update

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

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

$$\alpha_i \sim \mathcal{N}(\alpha_i; 0, 0.25I_2), \quad x_i \sim \mathcal{N}(x_i; 0, 1),$$

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

\[ u'_{it} \sim \begin{cases} \mathcal{N}(0,\infty)(u'_{it};-\alpha_{t,1} + \alpha_{t,2}x_i, 1) & \text{if } y_{it} = 1, \\ \mathcal{N}(-\infty,0)(u'_{it};-\alpha_{t,1} + \alpha_{t,2}x_i, 1) & \text{if } y_{it} = 0, \end{cases} \]

\[ \alpha'_t \sim \mathcal{N}(\alpha_t';m_{\alpha,t},(X^TX)^{-1}), \]

\[ x'_i \sim \mathcal{N}(x_i';m_{x,i},\sigma_x^{-2}), \]

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

We introduce the notation \( X \triangleq [-1 \ x_{1:n}]^\top \) for the design matrix. Furthermore, we use \( u_i = \{u_{it}\}_{t=1}^T \) and introduce \( \mathcal{N}_{(a,b)}(\mu,\sigma^2) \) 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

\[ m_{\alpha,t} = (X^TX)^{-1}Xu_t, \quad m_{x,i} = \sigma_x^{-2} \sum_{t=1}^{T} \alpha'_{t,2}(u_{it} + \alpha'_{t,1}), \quad \sigma_x^2 = 1 + \sum_{t=1}^{T} (\alpha'_{t,1})^2, \]

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

We make use of the command \texttt{MCMCirt1d} implemented in the R package \texttt{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 \textsc{mcmc} algorithms relies upon and briefly mention their underlying assumptions. For more information about the properties of \textsc{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 \( \varphi \) using the Markov chain generated by a \textsc{mcmc} algorithm is given by

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

(3.26)

where \( x_k \) denotes the state of the Markov chain at time step \( k \) with \( \pi \) 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
3.2 Three sampling strategies

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.,

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

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.,

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

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

\[ \| \mu - \pi \|_{TV} \triangleq \frac{1}{2} \int_{\mathcal{X}} |\mu(x) - \pi(x)| \, dx, \]

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

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

when \( K \to \infty \). Here, \( \sigma^2_{\text{MCMC}} \) denotes the variance of the estimator given by

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

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

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

where the integrated autocorrelation time (IACT) is given by

\[ \text{IACT}(x_{1:K}) \equiv 1 + 2 \sum_{k=1}^{\infty} \rho_k(x_{1:K}). \]
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 $\phi$ (green), $\sigma$ (orange), $\beta$ (purple) and $\sigma_e$ (magenta) in the Phillips curve model using the Swedish data. The prior distributions are indicated by grey curves and the posterior means by dotted lines.
3.3 Pseudo-marginal Metropolis-Hastings

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

\[
q(x \mid x') = \mathcal{N}(x'; x, \frac{2.38^2}{p} \hat{\Sigma}),
\]

where \( \hat{\Sigma} \) 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.

3.3 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 \( p(y | \theta) \) 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 \( p(y | \theta) \).

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

\[
\mathbb{E}_u \left[ \hat{\gamma}^N(x | u) \right] = \int_{\mathcal{U}} \hat{\gamma}^N(x | u) m(u) \, du = \gamma(x),
\]  

(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 \( \Sigma \) 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 (PMMH) 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

\[
q(x', u' | x, u) = q_u(x' | u, x) q_u(u' | u).
\]  

(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

\[
\gamma(x, u) = \hat{\gamma}^N(x | u) m(u).
\]  

(3.30)

As a result, we can recover the original target by the marginalisation given by the unbiasedness 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
\[
\alpha(x, u, x', u') = 1 \max \left\{ \frac{\overline{\gamma}(x' | u')}{\overline{\gamma}(x' | u)} q(x, u | x', u')/q(x', u' | x, u) \right\}.
\] (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 \(x' \sim q(x' | x_{k-1}, u_{k-1})\).

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

c) Compute the estimate of the un-normalised target \(\overline{\gamma}(x' | u')\).

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 \(\theta\).

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
\[
q(x' | x) = \mathcal{N} \left( x' ; x, \frac{2.56^2}{p} \hat{\Sigma} \right),
\] (3.32)

where \(\hat{\Sigma}\) 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 $p(\theta)$ where $\theta = \{\phi, \alpha, \beta, \sigma_\varepsilon\}$. To do this, we need to augment the model with prior distributions for each parameters, where we choose

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

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 $\theta$. The posterior covariance is estimated using a pilot run and it is given by

$$
\bar{\Sigma} = 10^{-3}. \begin{bmatrix}
7 & 2 & 1 & 0 \\
2 & 4 & 3 & -1 \\
1 & 3 & 1 & 0 \\
0 & -1 & 0 & 0.05
\end{bmatrix}.
$$

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 $\beta$ is negative. Remember that the sign of $\beta$ 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 $\beta$.

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.

3.4 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 SMC² 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 and 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.
4.1 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 $\text{smc}$ algorithm, this results in a linear\(^1\) decrease of the computational cost that also carries over to the $\text{pmh}$ algorithm and similar methods. In Paper H, we present results that illustrates how more informative data can actually increase the convergence rate of the $\text{em}$ 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 $\mathcal{I}(\theta^*)$, see (2.17), which is the curvature of the log-likelihood at the true parameter $\theta^*$. 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 $h(\mathcal{I}(\theta^*)) = \log \det \mathcal{I}(\theta^*)$.

A suitable input can then be computed by

$$u^* = u(\alpha^*), \quad \alpha^* = \underset{\alpha}{\text{argmax}} \, h(\mathcal{I}(\theta^*, u(\alpha))).$$

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

We encounter two main problems with this approach for finding $u^*$ namely: (i) how do we parametrise a good input $u(\alpha)$ for a specific model and (ii) how do we compute/estimate $\mathcal{I}(\theta^*, u)$. 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 $\alpha$ 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 $\text{smc}$ methods and the Fisher identity introduced in Remark 3.5. The major challenge is to obtain accurate estimates of $\mathcal{I}(\theta^*, u)$ 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 $\text{smc}$ 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.

4.2 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 $n T \propto T^2$. 
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 SMs 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 dataset. 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 \( \theta \) 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 a hierarchical model where a zero-mean Gaussian prior is selected for the AR coefficients, i.e., \( \phi_k \sim \mathcal{N}(0, \sigma_\phi^2) \). Moreover, we assume that \( \sigma_\phi^2 \sim \text{IG}(a_\sigma^\sigma, b_\sigma^\sigma) \), where the shape \( a_\sigma^\sigma > 0 \) and scale \( b_\sigma^\sigma > 0 \) 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 $g_\theta(y_t | x_t)$ 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 $\alpha$-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

$$y^*_t = y_t + \epsilon z_t,$$

where $\epsilon \geq 0$ denote the tolerance parameter and $z_t$ denote a standard Gaussian random variable. Hence, we make use of $y^* = \{y^*_t\}_{t=1}^T$ 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 $g_\theta(y_t | x_t)$ 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 $w_t(i) = \mathcal{N}(w_t(i); y_t - y^*_t, \epsilon^2)$, where $y^*_t$ denotes a sample from $g_\theta(\cdot | x_t(i))$.

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
4.2 Approximating the model

Figure 4.1. Illustration of two multi-modal distributions of the individual random effects $\beta_i$ 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 (right) random values of $\phi$. 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 $\theta$. 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.
4.3 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

\[
q(u' | u) = \mathcal{N}(u'; \sqrt{1 - \sigma_u^2}, \sigma_u^2 I_{N_u}),
\]

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 \( \theta \) 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 \( \sigma_u \) on the unit interval. We note that the correlation decreases from one to zero as we change the value of \( \sigma_u \). 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 \( \sigma_u \).

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 $\sigma_u$.

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 \( N \propto T^a \) for \( a < 1 \). 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 not 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

\[
\frac{dX_t}{dt} = b(X_t)dt + \sigma(X_t)dB_t, \quad X_0 = x_0,
\]

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

\[
b(X_t) = \frac{1}{2} \nabla \log \pi(X_t), \quad \sigma(X_t) = 1,
\]

then the resulting process is known as a Langevin diffusion or Brownian dynamics. This process has the interesting property that it has \( \pi \) 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

\[ q(x' | x) = \mathcal{N} \left( x'; x + \frac{\epsilon^2}{2} \nabla \log \pi(x), \epsilon^2 I_d \right), \tag{4.1} \]

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

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

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 \( \mu \) and the standard deviation \( \sigma \) 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-
4.3 Improving the inference algorithm

If we 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 $\epsilon$ 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 $\beta$ in the Phillips curve model to compare the PMHO and the QPMH2 algorithms. Remember that the PMHO 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

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

where $\mathcal{G}(\beta_k) = \nabla \log p(\theta|y)|_{\beta=\beta_k}$ denotes the gradient of the log-posterior evaluated at $\beta_k$. Hence, we can compute the mean of $\tilde{\beta}_{1:K}$ 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 PMHO 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 $\beta$ 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.
4.4 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 $\varphi$ 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 $\alpha_i$ 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 chooses an 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.

5.1 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 $u$.
5.1 Summary of contributions

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.

5.2 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 byQuiroz 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-numeric.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).

5.3 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.johandahl.in. 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**
- \( \overset{a.s.}{\rightarrow} \) Almost sure convergence.
- \( \overset{d}{\rightarrow} \) Convergence in distribution.
- \( \overset{p}{\rightarrow} \) 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 \( \varphi \) under the distribution \( \pi \).

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

- \( \mathbf{0}_p \): zero \( p \)-vector.
- \( \mathbf{I}_d \): \( d \times d \) identity matrix.
- \( \Delta \): Definition.
- \( \text{diag}(v) \): 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.
- \( \det(A), |A| \): Matrix determinant of \( A \).
- \( A^{-1}, A^\top \): Matrix inverse/transpose of \( A \).
- \( \text{tr}(A) \): Matrix trace of \( A \).
- \( v^2 = vv^\top \): Outer product of the vector \( v \).
- \( a_{nm} \): Sequence \( \{a_n, a_{n+1}, \ldots, a_{m-1}, a_m\} \), for \( m > n \).
- \( \text{sign}(x) \): Sign of \( x \).
- \( \text{supp}(f) \): Support of the function \( f \), \( \{x : f(x) > 0\} \).

Statistical quantities

- \( G(\theta) \): Gradient of log-posterior/target evaluated at \( \theta \).
- \( \mathcal{I}(\theta) \): Expected information matrix evaluated at \( \theta \).
- \( \mathcal{H}(\theta) \): Negative Hessian of log-posterior/target evaluated at \( \theta \).
- \( \mathcal{J}(\theta) \): Observed information matrix evaluated at \( \theta \).
- \( \hat{\theta} \): Parameter estimate.
- \( p(\theta|y) \): Parameter posterior distribution.
- \( p(\theta) \): Parameter prior distribution.
- \( \theta \): Parameter vector, \( \theta \in \Theta \subseteq \mathbb{R}^d \).
- \( S(\theta) \): Score function evaluated at \( \theta \).

Algorithmic quantities

- \( a_{t}^{(i)} \): Ancestor of particle \( i \) at time \( t \).
- \( R(x_{k-1}, x_k) \): Markov kernel.
- \( Z \): Normalisation constant.
- \( x_t^{(i)} \): Particle \( i \) at time \( t \).
- \( q_t(x_t|x_{0:t-1}) \): Particle proposal kernel.
- \( W_{0_t}(x_t, x_{t-1}) \): 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.
- \( w_t^{(i)}, \tilde{w}_t^{(i)} \): 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.


**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.)

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS</td>
<td>Particle smoother.</td>
</tr>
<tr>
<td>RJMH</td>
<td>Reversible jump Metropolis-Hastings.</td>
</tr>
<tr>
<td>RLS</td>
<td>Regularised least squares.</td>
</tr>
<tr>
<td>RTS</td>
<td>Rauch-Tung-Stribel.</td>
</tr>
<tr>
<td>SBP</td>
<td>Stick-breaking process.</td>
</tr>
<tr>
<td>SIS</td>
<td>Sequential importance sampling.</td>
</tr>
<tr>
<td>SIR</td>
<td>Sequential importance sampling and resampling.</td>
</tr>
<tr>
<td>SLLN</td>
<td>Strong law of large numbers.</td>
</tr>
<tr>
<td>SMC</td>
<td>Sequential Monte Carlo.</td>
</tr>
<tr>
<td>SPSA</td>
<td>Simultaneous perturbation stochastic approximation.</td>
</tr>
<tr>
<td>SSM</td>
<td>State space model.</td>
</tr>
</tbody>
</table>


A. Martin, K. Quinn, and J. H. Park. MCMCpack: Markov chain Monte Carlo in R. 


Part II

Papers
Papers

The articles associated with this thesis have been removed for copyright reasons. For more details about these see:
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-125992
PhD Dissertations
Division of Automatic Control
Linköping University


