Particle filters and Markov chains for learning of dynamical systems

Fredrik Lindsten
Cover illustration: Sample path from a Gibbs sampler targeting a three-dimensional standard normal distribution.

Linköping studies in science and technology. Dissertations.
No. 1530

Particle filters and Markov chains for learning of dynamical systems

Fredrik Lindsten

lindsten@isy.liu.se
www.control.isy.liu.se
Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping
Sweden

ISBN 978-91-7519-559-9    ISSN 0345-7524

Copyright © 2013 Fredrik Lindsten

Printed by LiU-Tryck, Linköping, Sweden 2013
To Åsa
Abstract

Sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC) methods provide computational tools for systematic inference and learning in complex dynamical systems, such as nonlinear and non-Gaussian state-space models. This thesis builds upon several methodological advances within these classes of Monte Carlo methods.

Particular emphasis is placed on the combination of SMC and MCMC in so called particle MCMC algorithms. These algorithms rely on SMC for generating samples from the often highly autocorrelated state-trajectory. A specific particle MCMC algorithm, referred to as particle Gibbs with ancestor sampling (PGAS), is suggested. By making use of backward sampling ideas, albeit implemented in a forward-only fashion, PGAS enjoys good mixing even when using seemingly few particles in the underlying SMC sampler. This results in a computationally competitive particle MCMC algorithm. As illustrated in this thesis, PGAS is a useful tool for both Bayesian and frequentistic parameter inference as well as for state smoothing. The PGAS sampler is successfully applied to the classical problem of Wiener system identification, and it is also used for inference in the challenging class of non-Markovian latent variable models.

Many nonlinear models encountered in practice contain some tractable substructure. As a second problem considered in this thesis, we develop Monte Carlo methods capable of exploiting such substructures to obtain more accurate estimators than what is provided otherwise. For the filtering problem, this can be done by using the well known Rao-Blackwellized particle filter (RBPF). The RBPF is analysed in terms of asymptotic variance, resulting in an expression for the performance gain offered by Rao-Blackwellization. Furthermore, a Rao-Blackwellized particle smoother is derived, capable of addressing the smoothing problem in so called mixed linear/nonlinear state-space models. The idea of Rao-Blackwellization is also used to develop an online algorithm for Bayesian parameter inference in nonlinear state-space models with affine parameter dependencies.
Populärvetenskaplig sammanfattning


I den här avhandlingen följer vi det sistnämnda alternativet. Mer specifikt så används en klass av approximativa inlärningsmetoder som kallas för *Monte Carlo-metoder*. Namnet är en anspielning på det kända kasinot i Monte Carlo och syftar på att dessa metoder baseras på slumpat. För att illustrera konceptron, antag att du lägger en patients, dvs. ett enmanskortspel som syftar till att lägga ut korten enligt vissa spelregler vilket leder antingen till vinst eller till förlust. Att på förhand räkna ut vad sannolikheten för vinst är kräver
komplicerade kombinatoriska uträkningar, som lätt blir praktiskt taget omöjliga att utföra. Ett mer pragmatiskt sätt är att spela ut korten, säg, 100 gånger och notera hur många av dessa försök som resulterar i vinst. Vinstfrekvensen blir en naturlig skattning av vinstsannolikheten. Denna skattning är inte helt tillförlitlig eftersom den är baserad på slumpen, men ju fler försök som utförs, desto högre noggrannhet uppnås i skattningen.


Acknowledgments

When I look back at the years that I have spent in the group of Automatic Control (henceforth abbreviated RT), the first thought that comes to mind is why was there a Peruvian giraffe on a backstreet in Brussels? Unfortunately, I don’t think that we will ever know the truth. The second thing that comes to my mind, however, is that those years at RT have been lots of fun. They have been fun for many reasons – the great atmosphere in the group, the fact that I have had so many good friends as colleagues, the pub nights, the barbecues… However, I also want to acknowledge all the members of the group who, skillfully and with great enthusiasm, engage in every activity whether it’s research, teaching or something else. To me, this has been very motivating. Let me spend a few lines on expressing my gratitude to some of the people who have made the years at RT, and the years before, a really great time.

First of all, I would like to thank my supervisor Prof Thomas B. Schön for all the guidance and support. Your dedication and enthusiasm is admirable and I have very much enjoyed working with you. Having you as a supervisor has resulted in (what I think is) a very nice research collaboration and a solid friendship, both which I know will continue in the future. I would also like to thank my co-supervisors Prof Lennart Ljung and Prof Fredrik Gustafsson for providing me with additional guidance and valuable expertise. Thanks to Prof Svante Gunnarsson and to Ninna Stensgård for creating such a nice atmosphere in the group and for always being very helpful.

I am most grateful for the financial support from the projects Calibrating Nonlinear Dynamical Models (Contract number: 621-2010-5876) funded by the Swedish Research Council and CADICS, a Linnaeus Center also funded by the Swedish Research Council.

In 2012, I had the privilege of spending some time as a visiting student researcher at the University of California, Berkeley. I want to express my sincere gratitude to Prof Michael I. Jordan for inviting me to his lab. The four months that I spent in Berkeley were interesting, fun and rewarding on so many levels. I made many new friends and started up several collaborations which I value highly today. Furthermore, it was during this visit that I got the opportunity to work on the theory which is now the foundation for this thesis!

I have also had the privilege of working with several other colleagues from outside the walls of RT. I want to express my thanks to Prof Eric Moulines, Dr Alexandre Bouchard-Côté, Dr Bonnie Kirkpatrick, Johan Wågberg, Roger Frigola, Dr Carl E. Rasmussen, Pete Bunch, Prof Simon J. Godsill, Ehsan Taghavi, Dr Lennart Svensson, Dr Adrian Wills, Prof Brett Ninness and Dr Jimmy Olsson for the time you have spent on our joint papers and/or research projects.

Among my colleagues at RT I want to particularly mention Johan Dahlin, Christian Andersson Naesseth and Dr Emre Özkan, with whom I have been working in close collaboration lately. Johan has also proofread the first part of this thesis. Thank you! Thanks also to my former roommate Lic (soon-to-be-Dr) Zoran Sjanic for being hilarious and for joining me in the creation of the most awesome music quiz even known to man! Thanks to Lic (even-sooner-to-be-Dr) Daniel Petersson for always taking the time to discuss various
problems that I have encountered in my research, despite the fact that it has been quite different from yours. Lic Martin Skoglund, Lic André Carvalho Bittencourt and Lic Peter Rosander have made sure that long-distance running is not as lonely as in Alan Sillitoe’s story. Thanks! Thanks also to Lic Sina Khoshfetrat Pakazad for being such a good friend and for keeping all the uninvited guests away from *Il Kraken*. Thanks to Floyd – by the way, where are you?

I also want to thank my old (well, we are starting to get old) friends from Y04C, my brother Mikael Lindsten and my other friends and family. My parents Anitha Lindsten and Anders Johansson have my deepest gratitude. I am quite sure that neither *particle filters* nor *Markov chains* mean a thing to you. Despite this you have always been very supportive and I love you both!

Most of all, I want to thank my wonderful wife Åsa Lindsten. Thank you for always believing in me and for all your encouragement and support. To quote Peter Cetera,

*You’re the inspiration!*

I love you!

Finally, in case I have forgotten someone, I would like to thank (your name here) for (fill in the reason, e.g., “all the good times”). Thanks!

*Linköping, September 2013*

*Fredrik Lindsten*
I Background

1 Introduction

1.1 Models of dynamical systems

1.2 Inference and learning

1.3 Contributions

1.4 Publications

1.5 Thesis outline

1.5.1 Outline of Part I

1.5.2 Outline of Part II

2 Learning of dynamical systems

2.1 Modeling

2.2 Maximum likelihood

2.3 Bayesian learning

2.4 Data augmentation

2.5 Online learning

3 Monte Carlo methods

3.1 The Monte Carlo idea

3.2 Rejection Sampling

3.3 Importance sampling

3.4 Particle filters and Markov chains

3.5 Rao-Blackwellization

4 Concluding remarks

4.1 Conclusions and future work

4.2 Further reading

Bibliography
## Publications

### A Backward simulation methods for Monte Carlo statistical inference

1. **Introduction**
2. **Background and motivation**
3. **Notation and definitions**
4. **A preview example**
5. **State-space models**
6. **Parameter learning in SSMs**
7. **Smoothing recursions**
8. **Backward simulation in linear Gaussian SSMs**
9. **Outline**

2. **Monte Carlo preliminaries**
   1. **Sequential Monte Carlo**
   2. **Markov chain Monte Carlo**

3. **Backward simulation for state-space models**
   1. **Forward filter/backward simulator**
   2. **Analysis and convergence**
   3. **Backward simulation with rejection sampling**
   4. **Backward simulation with MCMC moves**
   5. **Backward simulation for maximum likelihood inference**

4. **Backward simulation for general sequential models**
   1. **Motivating examples**
   2. **SMC revisited**
   3. **A general backward simulator**
   4. **Rao-Blackwellized FFBSi**
   5. **Non-Markovian latent variable models**
   6. **From state-space models to non-Markovian models**

5. **Backward simulation in particle MCMC**
   1. **Introduction to PMCMC**
   2. **Particle Marginal Metropolis-Hastings**
   3. **PMMH with backward simulation**
   4. **Particle Gibbs with backward simulation**
   5. **Particle Gibbs with ancestor sampling**
   6. **PMCMC for maximum likelihood inference**
   7. **PMCMC for state smoothing**

6. **Discussion**

---

## Bibliography

---

## B Ancestor Sampling for Particle Gibbs

1. **Introduction**
2. **Sequential Monte Carlo**
3. **Particle Gibbs with ancestor sampling**
4. **Truncation for non-Markovian state-space models**
5. **Application areas**
   1. **Rao-Blackwellized particle smoothing**
<table>
<thead>
<tr>
<th>Content</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2 Particle smoothing for degenerate state-space models</td>
<td>160</td>
</tr>
<tr>
<td>5.3 Additional problem classes</td>
<td>161</td>
</tr>
<tr>
<td>6 Numerical evaluation</td>
<td>161</td>
</tr>
<tr>
<td>6.1 RBPS: Linear Gaussian state-space model</td>
<td>161</td>
</tr>
<tr>
<td>6.2 Random linear Gaussian systems with rank deficient process noise</td>
<td>162</td>
</tr>
<tr>
<td>covariances</td>
<td></td>
</tr>
<tr>
<td>7 Discussion</td>
<td>163</td>
</tr>
<tr>
<td>A Proof of Proposition 1</td>
<td>163</td>
</tr>
<tr>
<td>Bibliography</td>
<td>165</td>
</tr>
<tr>
<td><strong>C Bayesian semiparametric Wiener system identification</strong></td>
<td>167</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>169</td>
</tr>
<tr>
<td>2 A Bayesian semiparametric model</td>
<td>171</td>
</tr>
<tr>
<td>2.1 Alt. I – Conjugate priors</td>
<td>171</td>
</tr>
<tr>
<td>2.2 Alt. II – Sparsity-promoting prior</td>
<td>172</td>
</tr>
<tr>
<td>2.3 Gaussian process prior</td>
<td>173</td>
</tr>
<tr>
<td>3 Inference via particle Gibbs sampling</td>
<td>174</td>
</tr>
<tr>
<td>3.1 Ideal Gibbs sampling</td>
<td>174</td>
</tr>
<tr>
<td>3.2 Particle Gibbs sampling</td>
<td>175</td>
</tr>
<tr>
<td>4 Posterior parameter distributions</td>
<td>178</td>
</tr>
<tr>
<td>4.1 MNIW prior – Posterior of Γ and Q</td>
<td>178</td>
</tr>
<tr>
<td>4.2 GH prior – Posterior of Γ, Q and ( \hat{\tau} )</td>
<td>178</td>
</tr>
<tr>
<td>4.3 Posterior of ( r )</td>
<td>180</td>
</tr>
<tr>
<td>4.4 Posterior of ( h(\cdot) )</td>
<td>180</td>
</tr>
<tr>
<td>4.5 Posterior of ( \eta )</td>
<td>181</td>
</tr>
<tr>
<td>5 Convergence analysis</td>
<td>181</td>
</tr>
<tr>
<td>5.1 Convergence of the Markov chain</td>
<td>182</td>
</tr>
<tr>
<td>5.2 Consistency of the Bayes estimator</td>
<td>184</td>
</tr>
<tr>
<td>6 Numerical illustrations</td>
<td>185</td>
</tr>
<tr>
<td>6.1 6th-order system with saturation</td>
<td>186</td>
</tr>
<tr>
<td>6.2 4th-order system with non-monotone nonlinearity</td>
<td>187</td>
</tr>
<tr>
<td>6.3 Discussion</td>
<td>188</td>
</tr>
<tr>
<td>7 Conclusions and future work</td>
<td>189</td>
</tr>
<tr>
<td>A Choosing the hyperparameters</td>
<td>190</td>
</tr>
<tr>
<td>Bibliography</td>
<td>191</td>
</tr>
<tr>
<td><strong>D An efficient SAEM algorithm using conditional particle filters</strong></td>
<td>195</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>197</td>
</tr>
<tr>
<td>2 The EM, MCEM and SAEM algorithms</td>
<td>198</td>
</tr>
<tr>
<td>3 Conditional particle filter SAEM</td>
<td>199</td>
</tr>
<tr>
<td>3.1 Markovian stochastic approximation</td>
<td>200</td>
</tr>
<tr>
<td>3.2 Conditional particle filter with ancestor sampling</td>
<td>200</td>
</tr>
<tr>
<td>3.3 Final identification algorithm</td>
<td>203</td>
</tr>
<tr>
<td>4 Numerical illustration</td>
<td>203</td>
</tr>
<tr>
<td>5 Conclusions</td>
<td>205</td>
</tr>
<tr>
<td>Bibliography</td>
<td>207</td>
</tr>
</tbody>
</table>
### E Rao-Blackwellized particle smoothers for mixed linear/nonlinear SSMs 209
1 Introduction .......................................................... 211
2 Background .......................................................... 212
   2.1 Particle filtering and smoothing .......................... 212
   2.2 Rao-Blackwellized particle filtering ..................... 213
3 Rao-Blackwellized particle smoothing ........................ 213
4 Proofs ............................................................... 216
5 Numerical results .................................................. 217
6 Conclusion .......................................................... 219
Bibliography .......................................................... 221

### F A non-degenerate RBPF for estimating static parameters in dynamical models 223
1 Introduction .......................................................... 225
2 Degeneracy of the RBPF – the motivation for a new approach 227
3 A non-degenerate RBPF for models with static parameters 229
   3.1 Sampling from the marginals ............................. 230
   3.2 Gaussian mixture approximation ......................... 230
   3.3 Resulting Algorithm ....................................... 232
4 Numerical illustration .............................................. 232
5 Discussion and future work ..................................... 233
6 Conclusions .......................................................... 234
Bibliography .......................................................... 237

### G An explicit variance reduction expression for the Rao-Blackwellised particle filter 239
1 Introduction and related work .................................. 241
2 Background .......................................................... 243
   2.1 Notation ......................................................... 243
   2.2 Particle filtering .............................................. 243
   2.3 Rao-Blackwellised particle filter ....................... 244
3 Problem formulation ............................................... 245
4 The main result ..................................................... 247
5 Relationship between the proposals kernels ................. 248
   5.1 Example: Bootstrap kernels ............................... 249
6 Discussion .......................................................... 250
7 Conclusions .......................................................... 251
A Proofs ............................................................... 251
Bibliography .......................................................... 253
## Notation

### Probability

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>Sampled from or distributed according to</td>
</tr>
<tr>
<td>( \mathbb{P}, \mathbb{E} )</td>
<td>Probability, expectation</td>
</tr>
<tr>
<td>( \text{Var, Cov} )</td>
<td>Variance, covariance</td>
</tr>
<tr>
<td>( \xrightarrow{D} )</td>
<td>Convergence in distribution</td>
</tr>
<tr>
<td>( \mathcal{L}(X \in \cdot) )</td>
<td>Law of the random variable ( X )</td>
</tr>
<tr>
<td>( | \mu_1 - \mu_2 |_{TV} )</td>
<td>Total variation distance, ( \sup_A</td>
</tr>
</tbody>
</table>

### Common Distributions

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Cat}({p_i}_{i=1}^n) )</td>
<td>Categorical over ( {1, \ldots, n} ) with probabilities ( {p_i}_{i=1}^n )</td>
</tr>
<tr>
<td>( U([a, b]) )</td>
<td>Uniform over the interval ([a, b])</td>
</tr>
<tr>
<td>( \mathcal{N}(m, \Sigma) )</td>
<td>Multivariate Gaussian with mean ( m ) and covariance ( \Sigma )</td>
</tr>
<tr>
<td>( \delta_x )</td>
<td>Point-mass at ( x ) (Dirac ( \delta )-distribution)</td>
</tr>
</tbody>
</table>

### Operators and Other Symbols

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \cup, \cap )</td>
<td>Set union, intersection</td>
</tr>
<tr>
<td>( \text{card}(S) )</td>
<td>Cardinality of the set ( S )</td>
</tr>
<tr>
<td>( S^c )</td>
<td>Complement of ( S ) in ( \Omega ) (given by the context)</td>
</tr>
<tr>
<td>( I_S(\cdot) )</td>
<td>Indicator function of set ( S )</td>
</tr>
<tr>
<td>( I_d )</td>
<td>( d )-dimensional identity matrix</td>
</tr>
<tr>
<td>( A^T )</td>
<td>Transpose of matrix ( A )</td>
</tr>
<tr>
<td>( \det(A),</td>
<td>A</td>
</tr>
<tr>
<td>( \text{tr}(A) )</td>
<td>Trace of matrix ( A )</td>
</tr>
<tr>
<td>Notation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>( \text{vec}(A) )</td>
<td>Vectorization, stacks the columns of ( A ) into a vector</td>
</tr>
<tr>
<td>( \text{diag}(v) )</td>
<td>Diagonal matrix with elements of ( v ) on the diagonal</td>
</tr>
<tr>
<td>( \otimes )</td>
<td>Kronecker product</td>
</tr>
<tr>
<td>( \text{supp}(f) )</td>
<td>Support of function ( f ), ( { x : f(x) &gt; 0 } )</td>
</tr>
<tr>
<td>( | f |_\infty )</td>
<td>Supremum norm, ( \sup_x</td>
</tr>
<tr>
<td>( \text{osc}(f) )</td>
<td>Oscillator norm, ( \sup_{(x,x')}</td>
</tr>
<tr>
<td>( a_{m:n} )</td>
<td>Sequence, ( (a_m, a_{m+1}, \ldots, a_n) )</td>
</tr>
<tr>
<td>( \equiv )</td>
<td>Definition</td>
</tr>
</tbody>
</table>

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACF</td>
<td>Autocorrelation function</td>
</tr>
<tr>
<td>ADM</td>
<td>Average derivative method</td>
</tr>
<tr>
<td>APF</td>
<td>Auxiliary particle filter</td>
</tr>
<tr>
<td>ARD</td>
<td>Automatic relevance determination</td>
</tr>
<tr>
<td>a.s.</td>
<td>almost surely</td>
</tr>
<tr>
<td>CLGSS</td>
<td>Conditionally linear Gaussian state-space</td>
</tr>
<tr>
<td>CLT</td>
<td>Central limit theorem</td>
</tr>
<tr>
<td>CPF</td>
<td>Conditional particle filter</td>
</tr>
<tr>
<td>CPF-AS</td>
<td>Conditional particle filter with ancestor sampling</td>
</tr>
<tr>
<td>CSMC</td>
<td>Conditional sequential Monte Carlo</td>
</tr>
<tr>
<td>DPMM</td>
<td>Dirichlet process mixture model</td>
</tr>
<tr>
<td>ESS</td>
<td>Effective sample size</td>
</tr>
<tr>
<td>EM</td>
<td>Expectation maximization</td>
</tr>
<tr>
<td>FFBSi</td>
<td>Forward filter/backward simulator</td>
</tr>
<tr>
<td>FFBSm</td>
<td>Forward filter/backward smoother</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite impulse response</td>
</tr>
<tr>
<td>GH</td>
<td>Generalized hyperbolic</td>
</tr>
<tr>
<td>GIG</td>
<td>Generalized inverse-Gaussian</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian mixture model</td>
</tr>
<tr>
<td>GP</td>
<td>Gaussian process</td>
</tr>
<tr>
<td>GPB</td>
<td>Generalized pseudo-Bayesian</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov model</td>
</tr>
<tr>
<td>i.i.d.</td>
<td>independent and identically distributed</td>
</tr>
<tr>
<td>IMM</td>
<td>Interacting multiple model</td>
</tr>
<tr>
<td>IW</td>
<td>Inverse Wishart</td>
</tr>
<tr>
<td>JMLS</td>
<td>Jump Markov linear system</td>
</tr>
<tr>
<td>JSD</td>
<td>Joint smoothing density</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman filter</td>
</tr>
<tr>
<td>KLD</td>
<td>Kullback-Leibler divergence</td>
</tr>
<tr>
<td>LGSS</td>
<td>Linear Gaussian state-space</td>
</tr>
<tr>
<td>LTI</td>
<td>Linear time-invariant</td>
</tr>
<tr>
<td>MBF</td>
<td>Modified Bryson-Frazier</td>
</tr>
<tr>
<td>MCEM</td>
<td>Monte Carlo expectation maximization</td>
</tr>
<tr>
<td>Notation</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>MCMC</td>
<td>Markov chain Monte Carlo</td>
</tr>
<tr>
<td>MH</td>
<td>Metropolis-Hastings</td>
</tr>
<tr>
<td>MH-FFBP</td>
<td>Metropolis-Hastings forward filter/backward proposing</td>
</tr>
<tr>
<td>MH-FFBSi</td>
<td>Metropolis-Hastings forward filter/backward simulator</td>
</tr>
<tr>
<td>MH-IPS</td>
<td>Metropolis-Hastings improved particle smoother</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum likelihood</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum likelihood estimator</td>
</tr>
<tr>
<td>MNIW</td>
<td>Matrix normal inverse Wishart</td>
</tr>
<tr>
<td>MPF</td>
<td>Marginal particle filter</td>
</tr>
<tr>
<td>MRF</td>
<td>Markov random field</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability density function</td>
</tr>
<tr>
<td>PEM</td>
<td>Prediction-error method</td>
</tr>
<tr>
<td>PF</td>
<td>Particle filter</td>
</tr>
<tr>
<td>PG</td>
<td>Particle Gibbs</td>
</tr>
<tr>
<td>PGAS</td>
<td>Particle Gibbs with ancestor sampling</td>
</tr>
<tr>
<td>PGBS</td>
<td>Particle Gibbs with backward simulation</td>
</tr>
<tr>
<td>PIMH</td>
<td>Particle independent Metropolis-Hastings</td>
</tr>
<tr>
<td>PMCMC</td>
<td>Particle Markov chain Monte Carlo</td>
</tr>
<tr>
<td>PMMH</td>
<td>Particle marginal Metropolis-Hastings</td>
</tr>
<tr>
<td>PSAEM</td>
<td>Particle stochastic approximation expectation maximization</td>
</tr>
<tr>
<td>PSELM</td>
<td>Particle smoother expectation maximization</td>
</tr>
<tr>
<td>RB-FFBSi</td>
<td>Rao-Blackwellized forward filter/backward simulator</td>
</tr>
<tr>
<td>RB-FF/JBS</td>
<td>Rao-Blackwellized forward filter/joint backward simulator</td>
</tr>
<tr>
<td>RB-F/S</td>
<td>Rao-Blackwellized filter/smoothing</td>
</tr>
<tr>
<td>RBMPF</td>
<td>Rao-Blackwellized marginal particle filter</td>
</tr>
<tr>
<td>RBPF</td>
<td>Rao-Blackwellized particle filter</td>
</tr>
<tr>
<td>RBPS</td>
<td>Rao-Blackwellised particle smoother</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root-mean-square error</td>
</tr>
<tr>
<td>RS</td>
<td>Rejection sampling</td>
</tr>
<tr>
<td>RS-FFBSi</td>
<td>Rejection sampling forward filter/backward simulator</td>
</tr>
<tr>
<td>RTS</td>
<td>Rauch-Tung-Striebel</td>
</tr>
<tr>
<td>SAEM</td>
<td>Stochastic approximation expectation maximization</td>
</tr>
<tr>
<td>SIR</td>
<td>Susceptible/infected/recovered</td>
</tr>
<tr>
<td>SSM</td>
<td>State-space model</td>
</tr>
<tr>
<td>SMC</td>
<td>Sequential Monte Carlo</td>
</tr>
<tr>
<td>TV</td>
<td>Total variation</td>
</tr>
</tbody>
</table>
Part I

Background
This thesis addresses inference and learning of dynamical systems. Problems lacking closed form solutions are considered and we therefore make use of computational statistical methods based on random simulation to address these problems. In this introductory chapter, we formulate and motivate the learning problem which is studied throughout the thesis.

1.1 Models of dynamical systems

An often encountered problem in a wide range of scientific fields is to make predictions about some dynamical process based on previous observations from the process. As an example, in the field of epidemiology the evolution of contagious diseases is studied (Keeling and Rohani, 2007). Seasonal influenza epidemics each year cause millions of severe illnesses and hundreds of thousands of deaths worldwide (Ginsberg et al., 2009). Furthermore, new strains of influenza viruses can result in pandemic situations with very severe effects on the public health. In order to minimize the harm caused by an epidemic or a pandemic situation, a problem of paramount importance is to be able to predict the spread of the disease. Assume that regular observations are made of the number of infected individuals within a population, e.g. through disease case reports. Alternatively, Ginsberg et al. (2009) have demonstrated that this type of information can be acquired by monitoring search engine query data. Using these observations, we wish to predict how many new cases of illness that will occur within the population, say, during the coming weeks. The ability to accurately make such predictions can enable early response to epidemic situations, which in turn can reduce their impact.

There are numerous other areas in which similar prediction problems for dynamical processes arise. In finance, the ability to predict the future price of an asset based on previous
recordings of its value is of key relevance (Hull, 2011) and in automatic control, predictions of how a controlled plant responds to specific commands are needed for efficient control systems design (Åström and Murray, 2008; Ljung, 1999). Additional examples include automotive safety systems (Eskandarian, 2012), population dynamics (Turchin, 2003) and econometrics (Greene, 2008), to mention a few.

Despite the apparent differences between these examples, they can all be studied within a common mathematical framework. We collectively refer to these processes as dynamical systems. The word dynamical refers to the fact that these processes are evolving over time. For a thorough elementary introduction to dynamical systems, see e.g. the classical text books by Oppenheim et al. (1996) and Kailath (1980).

Common to the dynamical systems studied in this thesis is that observations, or measurements, \( y_t \) can be recorded at consecutive time points indexed by \( t = 1, 2, \ldots \). Based on these readings, we wish to draw conclusions about the system which generated the measurements. For instance, assuming that we have recorded the values \( y_{1:t} \triangleq (y_1, \ldots, y_t) \), the one-step prediction problem amounts to estimating what the value of \( y_{t+1} \) will turn out to be. Should we simply assume that \( y_{t+1} \) will be close to the most recent recording \( y_t \), or should we make use of older measurements as well, to account for possible trends? Such questions can be answered by using a model of the dynamical system. The model describes how to weigh the available information together to make as good predictions as possible.

For most applications, it is not possible to find models that exactly describe the measurements. There will always be fluctuations and variations in the data, not accounted for by the model. To incorporate such random components, the measurement sequence can be viewed as a realisation of a discrete-time stochastic process. A model of the system is then the same thing as a model of the stochastic process.

A specific class of models, known as state-space models (SSMs), is commonly used in the context of dynamical systems. These models play a central role in this thesis. The structure of an SSM can be seen as influenced by the notion of a physical system. The idea is that, at each time point, the system is assumed to be in a certain state. The state contains all relevant information about the system, i.e. if we would know the state of the system we would have full insight into its internal condition. However, the state is typically not known. Instead, we measure some quantities which depend on the state in some way. To exemplify the idea, let \( x_t \) be a random variable representing the state at time \( t \). An SSM for the system could then, for instance, be given by,

\[
\begin{align*}
    x_{t+1} &= a(x_t) + v_t, \\ 
    y_t &= c(x_t) + e_t.
\end{align*}
\] (1.1a)

(1.1b)

The expression (1.1a) describes the evolution of the system state over time. The state at time \( t + 1 \) is given by a transformation of the current state \( a(x_t) \), plus some process noise \( v_t \). The process noise accounts for variations in the system state, not accounted for by the model. Equation (1.1a) describes the dynamical evolution of the system and it is therefore known as the dynamic equation. The second part of the model, given by (1.1b), describes how the measurement \( y_t \) depends on the state \( x_t \) and some measurement noise \( e_t \). Consequently, (1.1b) is called the measurement equation. The model of a dynamical system
specified by (1.1) thus consists of the functions a and c, but also of the noise characteristics, i.e. of the probability distributions for the process noise and the measurement noise. The concept of SSMs will be further discussed in Chapter 2 and in Section 1 of Paper A.

1.2 Inference and learning

As argued above, models of dynamical systems are of key relevance in many scientific disciplines. Hence, it is crucial to have access to tools with which these models can be built. In this thesis, we consider the problem of learning models of dynamical systems based on available observations. On a high level, the learning problem can be described as follows,

\textbf{Learning:} Based on observations of the process \( \{y_t\}_{t \geq 1} \), find a mathematical model which, without being too complex, as accurately as possible can describe the observations.

A complicating factor when addressing this problem is that the state process \( \{x_t\}_{t \geq 1} \) in (1.1) is unobserved; it is said to be latent or hidden. Instead, as recognized in the description above, any conclusions that we wish to draw regarding the system must be inferred from observations of the measurement sequence \( \{y_t\}_{t \geq 1} \). A task which is tightly coupled to the learning problem is therefore to draw inference about the latent state,

\textbf{State inference:} Given a fully specified SSM and based on observations \( \{y_t\}_{t \geq 1} \), draw conclusions about some past, present or future state of the system, which is not directly visible but related to the measurements through the model.

For instance, even if the system model would be completely known, making a prediction about a future value of the system state amounts to solving a state inference problem. As we shall see in Chapter 2, state inference often plays an important role as an intermediate step when addressing the learning problem.

There exists a wide variety of models and modeling techniques. One common approach is to make use of parametric models. That is, the SSM in (1.1) is specified only up to some unknown (possibly multi-dimensional) parameter, denoted \( \theta \). The learning problem then amounts to draw inference about the value of \( \theta \) based on data collected from the system. This problem is studied in several related scientific fields, e.g. statistics, system identification and machine learning, all with their own notation and nomenclature. We will mainly use the word learning, but we also refer to this problem as identification, parameter inference and parameter estimation. We provide an example of a parametric SSM below. Alternative modeling techniques are discussed in more detail in Chapter 2. See also the monographs by Cappé et al. (2005), Ljung (1999) and West and Harrison (1997) for a general treatment of the learning problem in the context of dynamical systems.

--- Example 1.1 ---

To describe the evolution of a contagious disease, a basic epidemiological model is the susceptible/infected/recovered (SIR) model (Keeling and Rohani, 2007). In a population of constant size, we let \( S_t, I_t \) and \( R_t \) represent the fractions of susceptible, infected and recovered individuals at time \( t \), respectively. Rasmussen et al. (2011) and Lindsten and
Schön (2012) study a time-discrete SIR model with environmental noise and seasonal fluctuations, which is given by

\[
S_{t+1} = S_t + \mu - \mu S_t - \beta_t S_t I_t v_t, \tag{1.2a}
\]

\[
I_{t+1} = I_t - (\gamma + \mu) I_t + \beta_t S_t I_t v_t, \tag{1.2b}
\]

\[
R_{t+1} = R_t + \gamma I_t - \mu R_t. \tag{1.2c}
\]

Here, \(\beta_t\) is a seasonally varying transmission rate given by \(\beta_t = \bar{\beta}(1 + \alpha \sin(2 \pi t / 365))\), where it is assumed that the time \(t\) is measured in days. Together with \(\alpha\) and \(\bar{\beta}\), the parameters of the model are the birth/death rate \(\mu\), the recovery rate \(\gamma\) and the variance \(\sigma_v^2\) of the zero-mean Gaussian process noise \(v_t\). That is, we can collect the system parameters in a vector

\[
\theta = (\alpha \quad \bar{\beta} \quad \mu \quad \gamma \quad \sigma_v^2)^T.
\]

The SIR model in (1.2) corresponds to the process model (1.1a). Note that the system state \(x_t = (S_t, I_t, R_t)\) is not directly observed. Instead, Lindsten and Schön (2012) consider an observation model which is inspired by the Google Flu Trends project (Ginsberg et al., 2009). The idea is to use the frequency of influenza related search engine queries to infer knowledge about the dynamics of the epidemic. The observation model, corresponding to (1.1b), is a linear relationship between the observations and the log-odds of infected individuals, i.e.

\[
y_t = \log \left( \frac{I_t}{1 - I_t} \right) + e_t, \tag{1.3}
\]

with \(e_t\) being a zero-mean Gaussian noise.

Lindsten and Schön (2012) use a method denoted as particle Gibbs with backward simulation (PGBS; see Section 5 of Paper A) to learn the parameters of this SIR model. Using the identified model, a state inference problem is solved in order to make one-month-ahead predictions of the number of infected individuals. The results from a simulation study are shown in Figure 1.1, illustrating the possibility of forecasting the disease activity by using a dynamical model.

The SIR model (1.2) is an example of an SSM in which the functions \(a\) and \(c\) in (1.1) depend nonlinearly on the state \(x_t\). Such SSMs are referred to as nonlinear. Conversely, if both \(a\) and \(c\) are linear (of affine) functions of \(x_t\), the SSM is also called linear. Linear models play an important role for many applications. However, there are also many cases in which they are inadequate for capturing the dynamics of the system under study; the epidemiological model above being one example. Despite this limitation, much emphasis has traditionally been put on linear models. One factor contributing to this is that nonlinear models by nature are much more difficult to work with. However, as we develop more sophisticated computational tools and acquire more and more computational resources, we can also address increasingly more challenging problems. Inspired by this fact, this thesis is focused on the use of computational methods for inference and learning of nonlinear dynamical models.
1.3 Contributions

The main contribution of this thesis is the development of new methodology for state inference and learning of dynamical systems. In particular, an algorithm referred to as particle Gibbs with ancestor sampling (PGAS) is proposed. It is illustrated that PGAS is a useful tool for both Bayesian and frequentistic learning as well as for state inference. The following contributions are made:

- The PGAS sampler is derived and its validity assessed by viewing the individual steps of the algorithm as a sequence of partially collapsed Gibbs steps (Paper B).
- A truncation strategy for backward sampling in so called non-Markovian latent variable models is developed and used together with PGAS (Paper B). The connections between non-Markovian models and several important types of SSMs are discussed (Paper A), motivating the development of inference strategies for this model class.
- An algorithm based on PGAS is developed for the classical problem of Wiener system identification (Paper C).
- PGAS is combined with stochastic approximation expectation maximization, resulting in method for frequentistic learning of nonlinear SSMs (Paper D).

In particular, we make use of a class of methods based on random simulation, referred to as Monte Carlo methods (Robert and Casella, 2004; Liu, 2001). This is a broad class of computational algorithms which are useful for addressing high-dimensional, intractable integration problems. We make use of Monte Carlo methods to address both state inference and learning problems. In particular, we employ methods based on so called Markov chains and on interacting particle systems. An introduction to basic Monte Carlo methods is given in Chapter 3. More advanced methods are discussed in Paper A in Part II of this thesis.

**Figure 1.1:** Number of infected individuals $I_t$ in a population of size $10^6$ over an 8 year period. Data from the first 4 years are used to learn the unknown parameters of the model. For the consecutive 4 years, one-month-ahead predictions are computed using the estimated model. See (Lindsten and Schön, 2012) for details on the experiment.
Many nonlinear models encountered in practice contain some tractable substructure. When addressing learning and inference problems for such models, this structure can be exploited to improve upon the performance of the algorithms. In this thesis we consider a type of structure exploitation referred to as Rao-Blackwellization. We develop and analyse several Rao-Blackwellized Monte Carlo methods for inference and learning in nonlinear SSMs. The following contributions are made:

- A Rao-Blackwellized particle smoother is developed for a class of mixed linear/nonlinear SSMs (Paper E).
- An online, Bayesian identification algorithm, based on the Rao-Blackwellized particle filter, is developed (Paper F).
- The asymptotic variance of the Rao-Blackwellized particle filter is analysed and an expression for the variance reduction offered by Rao-Blackwellization is derived (Paper G).

1.4 Publications

Published work of relevance to this thesis are listed below in reversed chronological order. Items marked with a star are included in Part II of the thesis.


Other publications, loosely connected to the material presented in this thesis, are:


1.5 Thesis outline

The thesis is divided into two parts. The first part contains background material and an introduction to the problem studied throughout the thesis. The second part is a compilation of seven edited publications. However, the first publication, Paper A, is a self-contained tutorial article covering many of the topics studied in the thesis. Paper A should therefore be viewed as part of the introduction, complementing the material presented in Part I.

1.5.1 Outline of Part I

Chapter 2 introduces the learning problem for dynamical systems. The maximum likelihood and the Bayesian learning criteria are defined and we discuss the basic strategies for addressing these problems. Chapter 3 is an introduction to basic Monte Carlo methods. The algorithms discussed in this chapter are the building blocks needed for constructing more advanced methods later in the thesis. Readers familiar with Monte Carlo statistical inference can skip this chapter. Finally, Chapter 4 concludes the thesis and point out possible directions for future work.

1.5.2 Outline of Part II

Part II is a compilation of seven edited publications.


is a self-contained tutorial article covering a branch of Monte Carlo methods referred to as backward simulators. These methods are useful for inference in probabilistic models containing latent stochastic processes, e.g. SSMs. The first two sections of this paper should preferably be read as part of the introduction, as they complement the background material presented in Part I of the thesis. In particular,

1. SSMs are introduced in Chapter 2, but a more thorough discussion is provided in Section 1 of Paper A.

2. Particle filters and Markov chains, the two main computational tools which are employed throughout this thesis, are briefly discussed in Chapter 3. However, a more thorough introduction is given in Section 2 of Paper A.

In the remaining sections of Paper A, several Monte Carlo methods based on particle filters and on Markov chains are discussed. In particular, it is illustrated how backward simulation can be used to address the so called smoothing problem and many state-of-the-art particle smoothers are surveyed.

contains the derivation of the PGAS method. PGAS belongs to the family of so called particle Markov chain Monte Carlo (PMCMC) algorithms. PMCMC is a combination of particle filters and Markov chain theory, resulting in potent tools for Bayesian learning and state inference. PGAS makes use of a technique reminiscent of backward simulation, albeit implemented in a forward-only fashion, to improve the performance of the algorithm. In particular, PGAS has been found to work well even when using few particles in the underlying particle filter. This implies that the algorithm is computationally competitive when compared with many other particle-filter-based methods. It is also discussed how PGAS can be used for inference in the challenging class of non-Markovian latent variable models.

**Paper C,**


makes use of PGAS for addressing the classical problem of Wiener system identification. A Wiener system is composed of a linear dynamical system followed by a static nonlinearity. That is, the measured quantity is a nonlinear transformation of the output from the linear dynamical system. A semiparametric model is assumed for the Wiener system. The model consists of a parametric model for the linear dynamical system and a nonparametric model for the static nonlinearity. The resulting identification algorithm can handle challenging situations, such as process noise and non-monotonicity of the nonlinearity, in a systematic manner.

**Paper D,**


is also based on the PGAS algorithm. In its original formulation, PGAS is useful for addressing the Bayesian learning problem. In this paper, the algorithm is adapted to instead solve the maximum likelihood problem. This is accomplished by using PGAS together with, so called, stochastic approximation expectation maximization. The resulting algorithm is shown to be computationally very competitive when compared with alternative particle-filter-based expectation maximization methods.

The last three papers are not (directly) related to PGAS. Instead, the common denominator in these papers is that they make use of Rao-Blackwellization. Many nonlinear models encountered in practice contain some tractable substructure. In the context of particle filtering, Rao-Blackwellization refers to the process of exploiting such substructures to improve the performance of the algorithms.
Paper E,


presents a Rao-Blackwellized backward simulation method. This algorithm can be used to address the state inference problem in a class of SSMs referred to as mixed linear/nonlinear. In these models, the state can be partitioned into two components, one which enters linearly and one which enters nonlinearly. By exploiting this structure, the proposed algorithm results in more accurate estimators than what is obtained otherwise.

Paper F,


considers the problem of online Bayesian learning. That is, we seek to learn a model which is continuously updated as new information is collected from the system. Inspired by the Rao-Blackwellized particle filter (RBPF), an approximate method capable of addressing this challenging problem is proposed. The method is applicable for Gaussian models with a linear dependence on the model parameters, but a possibly nonlinear dependence on the system state. At each time point, the posterior distribution of the system parameters is approximated by a Gaussian mixture. The components of this mixture distribution are systematically updated as new information becomes available by using moment matching.

Paper G,


the final paper of the thesis, provides an analysis of the RBPF. By considering the asymptotic variances of the particle filter and the RBPF, respectively, an expression for the improvement offered by Rao-Blackwellization is obtained.
This chapter introduces the learning problem for dynamical systems. We define the maximum likelihood and the Bayesian learning criteria and discuss the technique of data augmentation.

2.1 Modeling

On a high level, we can distinguish between different strategies for building models of dynamical systems as being white-, gray- or black-box modeling techniques (Ljung, 1999). A white-box model is based solely on first principles, such as Newton’s laws of motion. A gray-box model is constructed using similar insight into the structure of the dynamical system, but it also contains unknown parameters. These parameters have to be estimated from observations taken from the system. Finally, a black-box model is constructed using only observed data, with no structural knowledge about the system. Black-box models thus have to be flexible in order to capture different types of dynamical phenomena which are present in the data.

For gray- and black-box models, the process of estimating unknown model quantities based on observed data is what we refer to as learning. It should be noted, however, that learning sometimes refers to a more general problem, including how to specify the model structure, how to the design experiments for data collection etc. However, we shall restrict our attention to the aforementioned subtask, i.e. to estimate parameters or other unknown model quantities once the model structure has been specified.

As pointed out in Chapter 1, we will primarily be concerned with SSMs. This is a comprehensive and flexible class of models of dynamical systems. The additive noise model (1.1) is an example of an SSM. More generally, we can express the model in terms of probabil-
ity density functions (PDFs) as,

\[ x_{t+1} \sim f_{\theta}(x_{t+1} \mid x_t), \quad (2.1a) \]
\[ y_t \sim g_{\theta}(y_t \mid x_t), \quad (2.1b) \]

with the initial state \( x_1 \) distributed according to \( \mu_\theta(x_1) \). Here, \( f_{\theta}(x_{t+1} \mid x_t) \) is a Markov kernel encoding the probability of moving from state \( x_t \) at time \( t \) to state \( x_{t+1} \) at time \( t+1 \). Similarly, \( g_{\theta}(y_t \mid x_t) \) denotes the probability density of obtaining an observation \( y_t \), given that the current state of the system is \( x_t \). The latent state process \( \{x_t\}_{t \geq 1} \) is Markov and, conditionally on \( x_t \), the observation \( y_t \) is independent of past and future states and observations. SSMs are further discussed and exemplified in Section 1 of Paper A.

**Remark 2.1.** In the system identification literature (see e.g. Ljung (1999)), particular emphasis is put on learning of dynamical systems used in control applications. Hence, it is common to let the system be excited by some known control input \( \{u_t\}_{t \geq 1} \), i.e. by adding a dependence on \( u_t \) on the right hand side of (2.1). In this thesis, we will not make such dependence explicit, but this is purely for notational convenience. The learning methods that we consider are indeed applicable also in the presence of a known input signal.

The model (2.1) is said to be **parametric**, since it is specified only up to some finite-dimensional parameter \( \theta \in \Theta \), where \( \Theta \) denotes a set of plausible parameters. As noted in Chapter 1, an often encountered problem is to make predictions about some future output from the system. Based on the model (2.1), the PDF of the one-step predictor can be computed as,

\[ p_{\theta}(y_{t+1} \mid y_{1:t}) = \int g_{\theta}(y_{t+1} \mid x_{t+1}) p_{\theta}(x_{t+1} \mid y_{1:t}) \, dx_{t+1}, \quad (2.2) \]

where \( y_{1:t} = (y_1, \ldots, y_t) \) denotes the observations collected up to time \( t \). There are two things that are interesting to note about this expression. First, the predictor depends on the model parameter \( \theta \). Hence, to be able to use the model for making predictions, we need to obtain knowledge about its unknown parameters. Second, the expression (2.2) depends on the predictive density for the latent state \( p_{\theta}(x_{t+1} \mid y_{1:t}) \). Consequently, making a prediction about a future output from the system amounts to solving a state inference problem.

The complexity and flexibility of a parametric model is typically related to the dimensionality of \( \theta \), i.e. to the number of adjustable parameters. However, there is a trade-off between using many parameters to obtain an expressive model, and using few parameters to unambiguously being able to learn the values of these parameters. If the model is too simplistic to capture the dynamics of the system, we say that it suffers from **under-fitting**. On the contrary, if the model is too complex and thereby prevents accurate learning of the model parameters, there is a problem of **over-fitting**. Over- and under-fitting occurs when there is a mismatch between the model complexity and the amount of available data, or more precisely the amount of information available in the data.

A different take on modeling of dynamical systems is provided by **nonparametric** models. The word **nonparametric** does not imply that these models lack parameters. On the contrary, it means that the number of parameters is allowed to grow with the amount of data. Mathematically, this is accomplished by allowing for an infinite-dimensional latent
structure in the model. For instance, a nonparametric model may contain a latent function which lacks any finite-dimensional representation. This is in contrast with a parametric model where the attention is restricted to a finite-dimensional parameter space. A simple example of a nonparametric model of a PDF is a kernel density estimate. To avoid over-fitting in the nonparametric setting, it is necessary that the model complexity grows in a controlled manner with the amount of data. However, this type of regularization is often intrinsic to the model. Nonparametric models thus avoid the intricate trade-off between model fit and model complexity and at the same time they provide a high degree of flexibility.

In this thesis, we will primarily consider parametric models. Consequently, for clarity of presentation, many of the concepts that we introduce in the sequel are specifically discussed in the context of parametric models. An exception is Paper C, in which a combination of parametric and nonparametric ideas are used to construct a model for a so-called Wiener system. The necessary background material on Bayesian nonparametric modeling is given in Section 2.3.

### 2.2 Maximum likelihood

Consider the parametric model (2.1). Assume that we have collected a batch of data \( y_{1:T} \), where \( T \) denotes some final time point, i.e. the length of the data record. We refer to the PDF of the measurement sequence \( p_{\theta}(y_{1:T}) \) as the likelihood function. The likelihood function depends on the model parameter \( \theta \). In fact, since the measurement sequence \( y_{1:T} \) is assumed to be fixed, it can be viewed as a mapping from the parameter space to the real line,

\[
p_{\theta}(y_{1:T}) : \Theta \rightarrow \mathbb{R}.
\]  

(2.3)

A sensible approach to parameter inference is to find a value of \( \theta \) which maximizes the likelihood function. That is, we seek a parameter value for which the observed data is “as likely as possible”; this idea is known as maximum likelihood (ML). Hence, we define the ML estimator as,

\[
\hat{\theta}_{\text{ML}} = \arg \max_{\theta \in \Theta} \log p_{\theta}(y_{1:T}).
\]  

(2.4)

The logarithm is introduced to simplify and to improve the numerics of the problem. Since the logarithm is strictly increasing, any maximizer of the log-likelihood function is also a maximizer of the likelihood function itself. The ML criterion was proposed, analysed and popularized by Sir Ronald Aylmer Fisher (1890–1962) in the early 20th century (Fisher, 1912, 1921, 1922). However, the idea can be traced back even further to, among others, Gauss, Hagen and Edgeworth (Hald, 1999). Aldrich (1997) provides a historical discussion on Fisher and the making of ML. Due to its appealing theoretical properties, it has a long tradition in many fields of science, including machine learning and system identification.

A challenge in computing the estimator (2.4) for a nonlinear SSM, however, is that the likelihood function in general is not available in closed form. Hence, it is not possible to evaluate the objective function in (2.4), which complicates the optimization problem.
In Section 2.4 we will see how the ML criterion can be related to a state inference problem, which can then be addressed using computational algorithms such as Monte Carlo methods.

### 2.3 Bayesian learning

An alternative inference strategy bears the name of the British statistician and reverend Thomas Bayes (1702–1761). In Bayesian learning (see e.g. Gelman et al. (2003)), model uncertainties are represented using stochasticity. A probabilistic hypothesis about the model is maintained. When observations regarding the validity of the hypothesis are obtained, the belief in the hypothesis is updated using Bayes’ rule. Bayes (1764) treated this problem, but only considered uniform priors. The ideas that we today refer to as Bayesian, were to a large extent pioneered and popularized by the French mathematician Pierre-Simon Laplace (1749–1827). In a memoir, produced at the age of 25 and supposedly unaware of Bayes’ work, Laplace (1774) discovered the more general form of Bayes’ rule that we use today.

In the parametric setting, the aforementioned hypothesis concerns the model parameters. Consequently, a Bayesian parametric model is characterized by the presence of a prior PDF $\pi(\theta)$ for the model parameter $\theta$, which is thus viewed as a random variable. The prior distribution summarizes our *a priori* knowledge about the parameter, i.e. what we know before we make any observations from the system. Such prior information is sometimes naturally available, e.g. due to physical constraints or insight into the system dynamics based on experience. In other cases, the prior is introduced simply to enable the application of Bayesian methods. In such cases, a pragmatic, but useful, strategy is to choose a prior which results in simple computations. This is achieved by using so called *conjugate priors* (see Section 2.4). It is also common to choose the prior distribution to be uninformative, meaning that it will affect the posterior degree of belief to a small extent.

Given a batch of observations $y_{1:T}$, the Bayesian learning problem amounts to computing the posterior PDF $p(\theta \mid y_{1:T})$. From Bayes’ rule, this can be expressed as

$$p(\theta \mid y_{1:T}) = \frac{p(\theta \mid y_{1:T}) \pi(\theta)}{p(y_{1:T})}, \quad (2.5)$$

The above expression relates the posterior PDF to the prior PDF and to the likelihood function. Note that in Bayesian probability $\theta$ is viewed as a random variable. Hence, the likelihood function should be thought of as the conditional PDF of the observations given $\theta$, i.e. $p(\theta \mid y_{1:T}) = p(y_{1:T} \mid \theta)$. However, to be able to discuss the different learning criteria in a common setting, we keep the notation $p(\theta \mid y_{1:T})$.

If we accept the Bayesian model, the posterior distribution provides a rich source of information about the parameter. It is a complete summary of the *a priori* knowledge and all the information which is available in the observed data. It can for instance be used to compute minimum mean-squared-error estimates of the parameters, but also to systematically reason about the uncertainties in these estimates. Since the posterior PDF depends on the likelihood function, we face similar challenges in computing (2.5) as in solving the ML problem (2.4). We discuss how to make use of computational methods to address this
In the nonparametric setting, the number of “parameters” is allowed to vary and to grow with the amount of data. Analogously to modeling unknown parameters as latent random variables, Bayesian nonparametric models accomplish this by using latent stochastic processes to represent the unknowns of the model; see e.g. Gershman and Blei (2012); Hjort et al. (2010); Jordan (2010) for an introduction to these models. Principally, learning of Bayesian nonparametric models is similar to learning of parametric models. Using Bayes’ rule, the likelihood of the data is combined with the prior to obtain the posterior distribution. However, for a nonparametric model, the target distribution is the posterior law of the latent stochastic process. To illustrate the idea, an example of a Bayesian nonparametric model is given below.

---

**Example 2.1: Gaussian process regression**

Regression analysis amounts to learning the relationships within a group of variables. With \( \xi \in \mathbb{R}^d \) representing an input variable and \( y \in \mathbb{R} \) representing an output variable, we seek a functional relationship such that \( y \approx f(\xi) \). The approximate equality reflects the fact that we often observe the function values only up to some uncertainty. Formally, we can write

\[
y = f(\xi) + e,
\]

where \( e \) is an error term, here assumed to be zero-mean Gaussian: \( e \sim \mathcal{N}(0, \sigma^2) \).

Assume that we observe an input/output data set \( D = \{\xi_i, y_i\}_{i=1}^n \) and wish to estimate the function \( f \). A parametric model can be obtained by fitting, for instance, a polynomial or a trigonometric function to the data. In the nonparametric setting, however, we seek a flexible model where the complexity increases with the number of data points \( n \). One way to accomplish this is to make use of a Gaussian process (GP) regression model (Rasmussen and Williams, 2006).

A GP is a stochastic process, such that any finite collection of sample points have a joint Gaussian distribution. This construction can be used in regression analysis by modeling \( f \) (which is indexed by \( \xi \)) as a GP with index set \( \mathbb{R}^d \). That is, any finite collection of function values, in particular the collection \( \{f(\xi_i)\}_{i=1}^n \), have a joint Gaussian distribution. The mean vector and the covariance matrix of this Gaussian distribution follow from the specification of the GP, i.e. from the definition of the prior. Typical choices allow for a high degree of flexibility, but ensure continuity (and sometimes smoothness) of the function \( f \); see Rasmussen and Williams (2006) for a discussion.

Consider now a previously unseen input value \( \xi^* \). From standard manipulations of Gaussian random variables, it follows that the conditional distribution of \( f(\xi^*) \), given \( D \), is also Gaussian with tractable mean and variance. Hence, the posterior GP can be used to predict output values at previously unseen inputs, i.e. it constitutes a model of the function \( f \). The process of GP regression is illustrated in Figure 2.1.
2.4 Data augmentation

The intractability of the likelihood function appearing in (2.4) and (2.5) is a result of the fact that the state sequence \( x_{1:T} \) is latent. Hence, to compute the likelihood of the data, we need to average over all possible state trajectories. More precisely, the likelihood function is given by a marginalization over \( x_{1:T} \) according to,

\[
p_\theta(y_{1:T}) = \int p_\theta(x_{1:T}, y_{1:T}) \, dx_{1:T}. \tag{2.7}
\]

Using the conditional independence properties of an SSM, the integrand can be written as,

\[
p_\theta(x_{1:T}, y_{1:T}) = \mu_\theta(x_1) \prod_{t=1}^{T} g_\theta(y_t \mid x_t) \prod_{t=1}^{T-1} f_\theta(x_{t+1} \mid x_t). \tag{2.8}
\]

The high-dimensional integration in (2.7) will in general lack a closed form solution. This difficulty is central when addressing the learning problem for SSMs. Indeed, the need for using computational methods, such as Monte Carlo, is tightly coupled to the intractability of the above integral. Many of the challenges discussed throughout this thesis is a manifestation of this problem, in one form or another.

The presence of a latent state suggests a technique known as data augmentation (Dempster et al., 1977; Tanner and Wong, 1987). While this technique goes beyond learning of SSMs, we discuss how it can be used in our setting below. Data augmentation is based on the idea that if the latent states \( x_{1:T} \) would be known, inference about \( \theta \) would be relatively simple.
This suggests an iterative approach, alternating between updating the belief about $x_{1:T}$ and updating the belief about $\theta$. The former step of the iteration corresponds to solving an intermediate state inference problem. In data augmentation schemes, the states are viewed as missing data, as opposed to the observed data $y_{1:T}$. That is, the intermediate state inference step amounts to augmenting the observed data, to recover the complete data set $\{x_{1:T}, y_{1:T}\}$. The complete data and the observed data likelihoods are related according to (2.7), suggesting that $p_\theta(x_{1:T}, y_{1:T})$ indeed can be useful for drawing inference about $\theta$.

Let us start by considering the Bayesian learning criterion. Assume for the time being that the complete data $\{x_{1:T}, y_{1:T}\}$ is available. From Bayes’ rule (cf. (2.5)) we then have,

$$p(\theta \mid x_{1:T}, y_{1:T}) = \frac{p_\theta(x_{1:T}, y_{1:T}) \pi(\theta)}{p(x_{1:T}, y_{1:T})},$$

where the complete data likelihood is given by (2.8). While computing the normalization constant in (2.9) can be problematic, it is indeed possible for many models of interest. In particular, for many complete data likelihoods, it is possible to identify a prior PDF $\pi(\theta)$ which is such that the posterior PDF $p(\theta \mid x_{1:T}, y_{1:T})$ belongs to the same family of distributions as the prior. The prior is then said to be conjugate to the complete data likelihood (Gelman et al., 2003). For conjugate models, the posterior PDF in (2.9) can be computed in closed form (still, assuming that $x_{1:T}$ is known). All members of the extensive exponential family of distributions have conjugate priors. If the normalization constant cannot be computed in closed form, it is possible to make use of Monte Carlo integration to compute (2.9). We discuss this in more detail in Paper A. See also Paper C, where this technique is used for Wiener system identification.

The problem in using (2.9), however, is that the states $x_{1:T}$ are not known. To address this issue, we will make use of Monte Carlo methods. In particular, one of the main methods that we will consider makes use of the observed data $y_{1:T}$ to impute values for the latent variables $x_{1:T}$ by simulation. Once we have generated a (representative) sample from $x_{1:T}$, this can be used to compute $\theta$ according to (2.9). More precisely, we can draw a sample of $\theta$ from the posterior distribution (2.9). These two steps are then iterated, i.e. the method alternates between:

(i) Sample $x_{1:T}$ given $\theta$ and $y_{1:T}$.

(ii) Sample $\theta$ given $x_{1:T}$ and $y_{1:T}$.

This is a so called Gibbs sampler, originating from the method proposed by Geman and Geman (1984). Under appropriate conditions, the distribution of the $\theta$-samples will converge to the target distribution (2.5). Hence, these samples provide an empirical representation of the posterior distribution which is the object of interest in Bayesian learning. The precise way in which the states $x_{1:T}$ are sampled in Step (i) will be discussed in detail in Paper A. For now, we note that the Gibbs sampler requires us to generate samples from a, typically, complicated and high-dimensional distribution in order to impute the latent state variables.

Data augmentation is useful also when addressing the ML problem (2.4). Indeed, the technique was popularized in the statistics community by the introduction of the expectation maximization (EM) algorithm by Dempster et al. (1977). EM is a data augmentation...
algorithm which leverages the idea of missing data to construct a surrogate cost function for the ML problem. Using the relationship

\[ p_\theta(x_{1:T} \mid y_{1:T}) = \frac{p_\theta(x_{1:T}, y_{1:T})}{p_\theta(y_{1:T})}, \]

the observed data log-likelihood function can be written as

\[ \log p_\theta(y_{1:T}) = \log p_\theta(x_{1:T}, y_{1:T}) - \log p_\theta(x_{1:T} \mid y_{1:T}). \]

For any \( \theta \in \Theta \), \( p_\theta(x_{1:T} \mid y_{1:T}) \) is a PDF and it thus integrates to one. Hence, by taking an arbitrary \( \theta' \in \Theta \), multiplying (2.11) with \( p_{\theta'}(x_{1:T} \mid y_{1:T}) \) and integrating w.r.t. \( x_{1:T} \) we get,

\[ \log p_\theta(y_{1:T}) = Q(\theta, \theta') - V(\theta, \theta'), \]

where we have defined the auxiliary quantities,

\[ Q(\theta, \theta') \triangleq \int \log p_\theta(x_{1:T}, y_{1:T}) p_{\theta'}(x_{1:T} \mid y_{1:T}) \, dx_{1:T} \]

\[ = \mathbb{E}_{\theta'}[\log p_\theta(x_{1:T}, y_{1:T}) \mid y_{1:T}] \]  

(2.13)

and \( V(\theta, \theta') \triangleq \mathbb{E}_{\theta'}[\log p_\theta(x_{1:T} \mid y_{1:T}) \mid y_{1:T}] \). From (2.12) it follows that, for any \( (\theta, \theta') \in \Theta^2 \),

\[ \log p_\theta(y_{1:T}) - \log p_{\theta'}(y_{1:T}) = (Q(\theta, \theta') - Q(\theta', \theta')) + (V(\theta', \theta') - V(\theta', \theta')). \]

(2.14)

The difference \( V(\theta', \theta') - V(\theta, \theta') \) can be recognized as the Kullback-Leibler divergence between \( p_{\theta'}(x_{1:T} \mid y_{1:T}) \) and \( p_\theta(x_{1:T} \mid y_{1:T}) \), which is known to be nonnegative (Kullback and Leibler, 1951). Hence, as an implication of (2.14) we get,

\[ Q(\theta, \theta') \geq Q(\theta', \theta') \Rightarrow \log p_\theta(y_{1:T}) \geq \log p_{\theta'}(y_{1:T}). \]

(2.15)

This result implies that the auxiliary quantity (2.13) can be used as a substitute for the log-likelihood function when solving the ML problem (2.4). More precisely, any sequence of iterates which increase the value of the \( Q \)-function, will also increase the value of the log-likelihood. This is exploited in the EM algorithm, which iterates between computing the expectation in (2.13) (the E-step) and maximizing the auxiliary quantity \( Q(\theta, \theta') \) (the M-step).

The auxiliary quantity of the EM algorithm is defined as the expectation of the complete data log-likelihood according to (2.13). The main challenge in using the EM algorithm for learning of general SSMs lies in the computation of this expectation. However, one possibility is to make use of Monte Carlo methods. That is, we generate samples from the latent states \( x_{1:T} \) and approximate the expectation in (2.13) by the sample average. Again, the details of how this simulation can be carried out will be discussed in Paper A.

### 2.5 Online learning

In the previous sections we have considered batch-wise learning. That is, we have assumed that a complete data set \( y_{1:T} \), for some final time point \( T \), is available throughout the learning process. In some applications, it is more natural to do the learning online,
by continuously updating the system model as new observations are obtained (Ljung and Söderström, 1983).

For instance, in the Bayesian, parametric setting, online learning amounts to sequentially computing the posterior PDFs, $p(\theta \mid y_{1:t})$ for $t = 1, 2, \ldots$. Similarly, we can construct a sequence of optimization problems as in (2.4) for online ML learning. Online learning is useful in situations where the properties of the system are changing over time. Since the online learning algorithm continuously receive feedback from the system, it can adapt to situations which are previously unseen. Online learning can also be useful in big data applications. If the data set is very large, it may be more efficient to process it in an online fashion, i.e. one data item at a time.

We will primarily be concerned with batch-wise learning in this thesis. However, in Paper F, an algorithm for online Bayesian learning of a specific class of SSMs is presented.
This chapter provides an introduction to basic Monte Carlo methods, such as rejection sampling and importance sampling. These are the building blocks for the more advanced methods which are studied throughout the thesis. For a thorough elementary treatment of the Monte Carlo idea, see the books by Robert and Casella (2004) and Liu (2001).

3.1 The Monte Carlo idea

The idea of Monte Carlo methods (see Metropolis and Ulam (1949) for an early discussion) is to make use of random simulation to carry out a computation which is otherwise tedious, or intractable, to perform. For instance, consider the problem of evaluating the intermediate quantity of the EM algorithm in (2.13). This amounts to computing an expected value w.r.t. $p_{\theta}(x_{1:T} \mid y_{1:T})$, i.e. to solve a generally high-dimensional integration problem. In many cases, and in particular for learning of nonlinear SSMs, this integration lacks a closed form solution. In such situations, Monte Carlo methods can be used to approximate the expected value with a sample average over samples generated from the underlying random variable.

More generally, let $\gamma(x)$ be a PDF, referred to as the target density, which is defined on some space $X$. Let $x$ be a random variable distributed according to $\gamma(x)$ and assume that we seek the expected value,

$$E[\varphi(x)] = \int \varphi(x) \gamma(x) \, dx,$$

for some test function $\varphi$ (cf. (2.13)). Let us start by making the assumption that we can generate independent samples $\{x^i\}_{i=1}^N$, distributed according to $\gamma(x)$. This is in fact a very restrictive assumption and a large part of this thesis is concerned with strategies for generating realizations from random variables with complicated distributions. Neverthe-
less it is instructive to make this assumption in order to be able to focus on the key idea underlying all Monte Carlo methods. Based on these samples, we can approximate (3.1) by the sample average,

$$
\hat{\gamma}_{MC}^N \triangleq \frac{1}{N} \sum_{i=1}^{N} \phi(x^i).
$$

(3.2)

An equivalent interpretation of this Monte Carlo estimator is to let the samples \( \{x^i\}_{i=1}^{N} \) define an empirical approximation of the target distribution,

$$
\hat{\gamma}_{MC}^N(dx) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^i}(dx),
$$

(3.3)

where \( \delta_{x'}(dx) \) denotes a Dirac point-mass located at \( x' \). Hence, we approximate the target distribution (which may be continuous) with a discrete probability distribution, by placing a point-mass probability of \( 1/N \) at each of the generated samples. Inserting the approximation (3.3) into (3.1) results in

$$
\int \phi(x) \gamma(x) \, dx \approx \int \phi(x) \frac{1}{N} \sum_{i=1}^{N} \delta_{x^i}(dx) = \frac{1}{N} \sum_{i=1}^{N} \phi(x^i),
$$

(3.4)

i.e. we indeed obtain the Monte Carlo estimator (3.2). The idea of letting a collection of samples define an empirical point-mass distribution as in (3.3) is very convenient and it will be frequently used in the sequel.

The Monte Carlo estimator (3.2) comes with many desirable properties, which to a large extent explains the popularity of the Monte Carlo method. First, it is unbiased, i.e. \( \mathbb{E}[\hat{\gamma}_{MC}^N] = \mathbb{E}[\phi(x)] \) where the former expectation is w.r.t. the random realizations \( \{x^i\}_{i=1}^{N} \). Second, the strong law of large numbers implies almost sure convergence, \( \hat{\gamma}_{MC}^N \xrightarrow{a.s.} \mathbb{E}[\phi(x)] \) as \( N \to \infty \). Additionally, if the variance of \( \phi(x) \) is finite, i.e. \( \sigma_{\phi}^2 = \text{Var}[\phi(x)] < \infty \), then a central limit theorem (CLT) holds,

$$
\sqrt{N} \left( \frac{\hat{\gamma}_{MC}^N - \mathbb{E}[\phi(x)]}{\sigma_{\phi}} \right) \xrightarrow{D} \mathcal{N}(0, 1), \quad N \to \infty,
$$

(3.5)

where \( \xrightarrow{D} \) denotes convergence in distribution. In fact, the variance of the estimator (3.2) is explicitly given by \( \text{Var}[\hat{\gamma}_{MC}^N] = \sigma_{\phi}^2 / N \). From (3.5) if follows that the Monte Carlo error decreases as \( O(N^{-\frac{1}{2}}) \). Interestingly, the convergence rate is independent of the dimension of \( X \). This clearly distinguishes Monte Carlo methods from deterministic integration methods, where the latter have an approximation error that grows with the dimension.

As pointed out above, the vanilla Monte Carlo method described above relies on the, often unrealistic, assumption that it is possible to generate independent and identically distributed (i.i.d.) samples from the target distribution. The rest of this chapter, and to a large extent Part II of this thesis, is devoted to strategies for generating samples from complicated target distributions, effectively rendering the use of Monte Carlo methods possible for challenging inference and learning problems.
3.2 Rejection Sampling

An often encountered difficulty is that the target density $\gamma(x)$ can be evaluated only up to proportionality. That is, we can write $\gamma(x) = \bar{\gamma}(x)/Z$, where $\bar{\gamma}(x)$ can be evaluated point-wise, but where the normalization constant $Z$ is unknown. The typical setting is when Bayes’ rule is used to express a posterior PDF in terms of the prior, the likelihood and the (unknown) normalizing constant. For instance, consider the Bayesian learning criteria (2.5). Even in situations when the likelihood function is available, if the model is non-conjugate, then the normalization constant $p(y_1:T)$ is typically unknown.

Rejection sampling (von Neumann, 1951) is a Monte Carlo method which, under these conditions, can be used to generate samples exactly distributed according to the target density $\gamma(x)$. To introduce the idea, let $\bar{\gamma}(z)$ be given by the function shown by the gray area in Figure 3.1. Let the two-dimensional random vector $(x,y)$ be distributed uniformly over the gray area. The area under the graph of $\bar{\gamma}(x)$ is $\int \bar{\gamma}(x) \, dx = Z$ which implies that the PDF of $(x,y)$ is,

$$p(x,y) = \begin{cases} 1/Z & \text{if } 0 \leq y \leq \bar{\gamma}(x), \\ 0 & \text{otherwise.} \end{cases} \quad (3.6)$$

Hence, the marginal PDF of $x$ is,

$$p(x) = \int p(x,y) \, dy = \int_{0}^{\bar{\gamma}(x)} \frac{1}{Z} \, dy = \gamma(x), \quad (3.7)$$

i.e. it holds that $x$ is marginally distributed according to the target distribution.

The problem is that sampling uniformly over the gray area is just as hard as the original problem, i.e. sampling from $\gamma(z)$. However, it leads us to the following idea. Let $q(x)$
Algorithm 1 Rejection sampling

1: \( L \leftarrow \{1, \ldots, N\} \).
2: while \( L \) is not empty do
3: \( n \leftarrow \text{card}(L) \).
4: \( \delta \leftarrow \emptyset \).
5: Sample independently \( \{x'(k)\}_{k=1}^n \sim q(x) \).
6: Sample independently \( \{u(k)\}_{k=1}^n \sim \mathcal{U}([0, 1]) \).
7: for \( k = 1 \) to \( n \) do
8: \( \text{if } u(k) \leq \frac{\tilde{\gamma}(x'(k))}{\rho q(x'(k))} \text{ then} \)
9: \( x^L(k) \leftarrow x'(k) \).
10: \( \delta \leftarrow \delta \cup \{L(k)\} \).
11: end if
12: end for
13: \( L \leftarrow L \setminus \delta \).
14: end while

be a user-chosen PDF which is easy to sample from. Such a distribution is referred to as a proposal distribution. Furthermore, assume that there exists a constant \( \rho \) such that \( \overline{\gamma}(x) \leq \rho q(x) \) for all \( x \in \mathbb{X} \). Now, if we sample independently and uniformly under the graph of \( \rho q(x) \), but only keep the samples that fall under the graph of \( \overline{\gamma}(x) \), then the surviving samples are i.i.d. draws from the target distribution.

More generally, let \( x' \) be sampled from the proposal and let \( u \) be drawn uniformly over the unit interval, i.e.

\[ x' \sim q(x), \quad (3.8a) \]
\[ u \sim \mathcal{U}([0, 1]). \quad (3.8b) \]

The variable \( u \) serves as an indicator on whether we should accept \( x' \) as a valid sample from the target distribution or not. More precisely, if \( \rho q(x') u \leq \overline{\gamma}(x') \) we set \( x = x' \), otherwise we reject \( x' \) and repeat the procedure (3.8) until a sample is accepted. The method is summarized in Algorithm 1, in which \( N \) i.i.d. samples \( \{x^i\}_{i=1}^N \) are generated in parallel.

To see the validity of the algorithm, consider the probability that \( x \) falls in some subset \( A \subset \mathbb{X} \),

\[ \mathbb{P}(x \in A) = \mathbb{P}(x' \in A \mid x' \text{ is accepted}) = \frac{\mathbb{P}(x' \in A \cap x' \text{ is accepted})}{\mathbb{P}(x' \text{ is accepted})}. \quad (3.9) \]

Since \( x' \) is distributed according to \( q(x) \) and \( u \) is uniform on \([0, 1]\), the numerator can be expressed as

\[ \mathbb{P}(x' \in A \cap u \leq \overline{\gamma}(x')/(\rho q(x'))) = \int_A \frac{\overline{\gamma}(x')}{\rho q(x')} q(x') \, dx' = \frac{Z}{\rho} \int_A \overline{\gamma}(x') \, dx'. \quad (3.10) \]

Analogously, the denominator in (3.9) is given by \( \mathbb{P}(x' \text{ is accepted}) = Z/\rho \). Inserting this
3.3 Importance sampling

into (3.9) we get,

$$P(x \in A) = \int_A \gamma(x) \, dx.$$  \hfill (3.11)

Since the set $A$ is arbitrary, we conclude that $x$ is indeed distributed according to the target density $\gamma(x)$.

The choice of proposal density $q(x)$ and the constant $\rho$ are very important from a practical point of view. As noted above, the acceptance probability is given by $P(x' \text{ is accepted}) = Z/\rho$. Consequently, the average number of candidate samples that need to be drawn to generate one sample from the target distribution is $\rho/Z$. It is therefore imperative that this ratio is not too large. However, for the algorithm to work, we also require that the graph of $\tilde{\gamma}(x)$ is completely below the graph of $\rho q(x)$, i.e. $\rho$ is at least as large as the largest discrepancy between the proposal $q(x)$ and the (unnormalized) target $\tilde{\gamma}(x)$. Finding a proposal density in close agreement with the target density is easy for the toy problem considered above. However, as the target density becomes more complicated and, in particular, as the dimension of $X$ increases, this becomes harder.

For the sake of illustration, assume that we wish sample from the $d$-dimensional, standard normal distribution using rejection sampling. As proposal, we use a $d$-dimensional, zero-mean normal distribution with covariance matrix $\sigma_q^2 I_d$. For the ratio between the target and the proposal densities to be bounded, we require that $\sigma_q^2 \geq 1$. The smallest bound is then given by $\rho = \sigma_q^d$. Hence, the acceptance probability decays exponentially as we increase the dimension of the problem. This is referred to as the curse of dimensionality. In high dimensions, what appears to be a small discrepancy between the proposal and the target densities, can in fact have a huge impact, rendering the method impractical.

### 3.3 Importance sampling

Importance sampling (Kahn and Harris, 1951; Marshall, 1956) offers a solution to the problem of evaluating integrals of the form (3.1), but it does not generate exact draws from the target distribution. In the rejection sampler introduced above, we first generate candidate samples from some proposal density $q(x)$. These samples are then either accepted or rejected with certain probabilities, depending on how well they fit the target distribution. Importance sampling proceeds similarly, by generating draws from a proposal distribution. However, rather than discarding some of the simulated values all samples are kept, but they are assigned individual weights depending on how well they fit the target.

Let $x' \sim q(x)$ be an instrumental random variable, distributed according to the proposal. We can then express (3.1) as,

$$\mathbb{E}[\varphi(x)] = \int \varphi(x) \gamma(x) \, dx = \int \frac{\varphi(x) \gamma(x)}{q(x)} q(x) \, dx$$

$$= \int \varphi(x) W(x) q(x) \, dx = \mathbb{E}[\varphi(x') W(x')]$$ \hfill (3.12)
where we have introduced the weight function \( W(x) \triangleq \gamma(x)/q(x) \) and where we have assumed that \( q(x) > 0 \) for all \( x \) where \( \varphi(x)\gamma(x) > 0 \) (i.e. \( \text{supp } \varphi \gamma \subset \text{supp } q \)). By construction, it is easy to generate samples from \( q(x) \). We can thus construct a Monte Carlo estimator for (3.12) by sampling independently \( x^i \sim q(x) \) for \( i = 1, \ldots, N \) and computing,

\[
\bar{\varphi}_{\text{IS}}^N = \frac{1}{N} \sum_{i=1}^{N} W(x^i)\varphi(x^i).
\]  

(3.13)

This estimator is similar to (3.2), but we see that the samples are weighted by so called importance weights, accounting for the discrepancy between the proposal and the target densities. Intuitively speaking, the importance weights contain information about how useful each proposed value \( x^i \) is for computing integrals on the form (3.1).

As mentioned in the previous section, it is common that it is only possible to evaluate the target density up to an unknown normalization constant. That is, we can write \( \gamma(x) = \bar{\gamma}(x)/Z \), where \( \bar{\gamma}(x) \) can be evaluated but the constant \( Z \) is unknown\(^1\). We then have,

\[
\mathbb{E}[\varphi(x)] = \int \varphi(x) \frac{\bar{\gamma}(x)}{Zq(x)} q(x) \, dx = \frac{1}{Z} \int \varphi(x)W(x)q(x) \, dx,
\]  

(3.14)

where (with abuse of notation) we have redefined the weight function as

\[
W(x) \triangleq \frac{\bar{\gamma}(x)}{q(x)}.
\]  

(3.15)

Hence, the importance sampling estimator (3.13) is given by,

\[
\hat{\varphi}_{\text{IS}}^N = \frac{1}{NZ} \sum_{i=1}^{N} \bar{w}^i \varphi(x^i),
\]  

(3.16)

where we have explicitly introduced the weights \( \bar{w}^i = W(x^i) \) for \( i = 1, \ldots, N \). Note that, since \( \bar{w}^i \) is given by a transformation of a random variable, it is itself a random variable. From the above expression it appears as if we have just moved the problem with the unknown normalization constants from one place to another. However, we can make use of the samples \( \{x^i\}_{i=1}^{N} \) to compute an approximation of the unknown constant. Indeed, the normalization constant is given by,

\[
Z = \int \bar{\gamma}(x) \, dx = \int \frac{\bar{\gamma}(x)}{q(x)} q(x) \, dx \approx \frac{1}{N} \sum_{i=1}^{N} \bar{w}^i.
\]  

(3.17)

By inserting this approximation into (3.16), we obtain the normalized importance sampling estimator,

\[
\hat{\bar{\varphi}}_{\text{IS}}^N = \sum_{i=1}^{N} w^i \varphi(x^i),
\]  

(3.18)

where \( \{w^i\}_{i=1}^{N} \) denote the normalized importance weights: \( w^i \triangleq \bar{w}^i/\sum_{i} \bar{w}^i \). Analog-

\(^1\)Similarly, we may assume that the proposal density can only be evaluated up to proportionality, but that is less common in practice.
gously to (3.3), an alternative interpretation of the above is that the importance sampler provides an empirical point-mass approximation of the target distribution, according to,

\[ \hat{\pi}_{IS}^N(dx) = \sum_{i=1}^N w^i \delta_{x^i}(dx). \]  

(3.19)

Hence, even though the importance sampler does not provide samples from the target distribution, the weighted samples \( \{x^i, w^i\}_{i=1}^N \) define an empirical distribution approximating the target. Inserting this empirical distribution into (3.1) straightforwardly results in the estimator (3.18). Note that, even if the constant \( Z \) is known, the importance weights must be normalized for the point-mass approximation (3.19) to be a probability distribution. The above development is summarized in Algorithm 2.

**Algorithm 2** Importance sampling (all operations are for \( i = 1, \ldots, N \))

1. Draw \( x^i \sim q(x) \).
2. Compute \( \bar{w}^i = W(x^i) \).
3. Normalize: set \( w^i = \bar{w}^i / \sum_i \bar{w}^i \).

From the discussion on the rejection sampler in Section 3.2, we recall that the choice of proposal distribution is important in order to obtain a practical algorithm. This holds true also for the importance sampler. A large discrepancy between the target and the proposal densities will lead to high variance in the importance weights, which carries over to the estimator (3.18).

### 3.4 Particle filters and Markov chains

Rejection sampling and importance sampling are important tools in the construction of Monte Carlo inferential methods. However, alone they do not provide satisfactory solutions to the challenging inference problems associated with learning of dynamical systems. When dealing with SSMs, data augmentation schemes typically introduce the state sequence \( x_{1:T} \) as auxiliary variables (see Section 2.4). Hence, the dimensionality of the problem increases with the length of the data record \( T \), which is typically large. For these learning problems, rejection sampling and importance sampling algorithms are often impractical, due to the difficulty of designing efficient proposal distributions in high dimensions. Consequently, there is a need for Monte Carlo methods which are more apt at addressing high-dimensional integration problems.

Two classes of algorithms play a central role in this thesis and, indeed, in the study of Monte Carlo methods as a whole. These are, respectively, methods based on interacting particle systems and on Markov chains. However, despite the relevance of these algorithms for this thesis, this background section only provides a very brief introduction to the concepts. The reason is that both classes of algorithms are introduced and more thoroughly described in Section 1 of Paper A.

The former class of methods are referred to as particle filters or sequential Monte Carlo (SMC) methods (Stewart and McCarty, 1992; Gordon et al., 1993); see also Doucet and
Johansen (2011); Gustafsson (2010); Doucet et al. (2001). Particle filters are useful for approximating a sequence of target distributions. For instance, in the context of SSMs, it is common to target the sequence of joint smoothing densities $p_{\theta}(x_{1:t} \mid y_{1:t})$ for $t = 1, 2, \ldots$. Initially, for $t = 1$, the density $p_{\theta}(x_{1} \mid y_{1})$ is approximated using importance sampling. This is much simpler than targeting, say, $p_{\theta}(x_{1:T} \mid y_{1:T})$ directly, due to the high dimensionality of the latter density for large $T$. Hence, we obtain an empirical point-mass approximation as in (3.19);

$$\hat{p}_{\theta}^{N}(dx_{1} \mid y_{1}) = \sum_{i=1}^{N} w_{i}^{1} \delta_{x_{i}^{1}}(dx_{1}). \quad (3.20)$$

In the SMC literature, the samples $\{x_{i}^{1}\}_{i=1}^{N}$ are called particles and $\{x_{i}^{1}, w_{i}^{1}\}_{i=1}^{N}$ is referred to as a weighted particle system. The particles can be thought of as (random) hypotheses about the state of the system at time $t = 1$. The belief in each of the hypotheses is represented by the corresponding importance weight.

The essence of the particle filter is a systematic procedure for updating these hypotheses to obtain an approximation of the next target density in the sequence. That is, given (3.20) we seek a point-mass approximation of $p_{\theta}(x_{1:2} \mid y_{1:2})$, then of $p_{\theta}(x_{1:3} \mid y_{1:3})$, and so on. Basically, this is accomplished by propagating and reevaluating the belief in the particles according to the model (2.1). By discarding or duplicating particles according to their importance weights, the particle filter is able to put emphasis on high-probability hypotheses, which are more likely to be useful for approximating the next target distribution in the sequence.

The second class of methods are so called Markov chain Monte Carlo (MCMC) samplers. A Markov chain is a memoryless stochastic process. That is, the next state of the chain depends only on the current state and not on the past history of the process. In MCMC, Markov chains are used to represent a sequence of hypotheses about some variable of interest, e.g. the state of a dynamical system at some specific time point or some unknown model parameter. MCMC samplers are thus iterative Monte Carlo methods where each sample (i.e. each hypothesis) is statistically dependent on the previous one. For this approach to be useful for inference, the Markov chain has to be constructed in such a way that, in the long run, the samples are representative for the target distribution. That is, the limiting distribution of the chain should coincide with the target distribution. In MCMC theory, there are systematic ways of constructing Markov chains with this specific property. The Metropolis-Hastings sampler (Metropolis et al., 1953; Hastings, 1970) and the Gibbs sampler (Geman and Geman, 1984) are the most well-known techniques. The latter method was briefly mentioned in the context of data augmentation in Section 2.4.

It is also possible to combine SMC and MCMC to construct composite algorithms, drawing on the strengths of both classes of methods. This thesis puts particular emphasis on a class of methods referred to a particle MCMC (PMCMC) algorithms (Andrieu et al., 2010). PMCMC relies on SMC for generating samples of the often highly auto-correlated state trajectory. Combined with MCMC, this results in powerful Monte Carlo methods which are capable of addressing both inference and learning problems in complex dynamical systems.
3.5 Rao-Blackwellization

In the mid 40’s, Rao (1945) and Blackwell (1947) established a fundamental result in estimation theory, which has later become known as the Rao-Blackwell theorem (see also (Lehmann, 1983, page 50)). Let $\theta$ be an unknown parameter and let $Y$ be some data drawn from a distribution parameterized by $\theta$. Given the data $Y$, we compute an estimator of $\theta$ denoted as $\hat{\theta}(Y)$. Furthermore, let $S$ be a sufficient statistic for $Y$, i.e. informally $S$ contains the same amount of information about $\theta$ as $Y$ does. Then, basically, the Rao-Blackwell theorem states that

$$\hat{\theta}_{RB}(S) = \mathbb{E}[\hat{\theta}(Y) \mid S]$$

(3.21)

is typically a better estimator than $\hat{\theta}(Y)$, and it is never worse. Hence, from a crude estimator $\hat{\theta}(Y)$ we can construct a better estimator $\hat{\theta}_{RB}(S)$, depending only on the sufficient statistic $S$, by computing a conditional expectation. This transformation is known as Rao-Blackwellization.

In this thesis, we are concerned with the implication of the Rao-Blackwell theorem for estimators constructed using Monte Carlo methods. Any Monte Carlo estimator is affected by variance due to the random simulation used in its construction. For instance, consider the vanilla Monte Carlo estimator in (3.2),

$$\hat{\varphi}_{MC}^N(X) = \frac{1}{N} \sum_{i=1}^{N} \varphi(x^i).$$

(3.22)

Here, we have explicitly introduced the dependence on the random samples $X \triangleq \{x^i\}_{i=1}^{N}$ in the notation. As previously mentioned, if the variance of the test function $\varphi(x)$ is finite, we have

$$\operatorname{Var}(\hat{\varphi}_{MC}^N(X)) = \frac{\operatorname{Var}(\varphi(x))}{N}. \quad (3.23)$$

This Monte Carlo variance reveals that there is a random error in the estimator. In fact, since (3.22) is unbiased the mean-squared-error is given by the variance (3.23). Hence, to obtain an accurate estimator it is desirable to keep the variance as small as possible.

By making use of Rao-Blackwellization, it is possible to reduce the Monte Carlo variance and thereby improve upon (3.22). Let the random vector $x$ be split into two components,

$$x = \begin{pmatrix} z \\ \xi \end{pmatrix}. \quad (3.24)$$

The samples $\{x^i\}_{i=1}^{N}$ are split accordingly and we can thus identify $X = \{Z, \Xi\}$, where $Z = \{z^i\}_{i=1}^{N}$ and $\Xi = \{\xi^i\}_{i=1}^{N}$. Furthermore, the test function can be written as $\varphi(x) = \varphi(z, \xi)$ and it follows the estimator (3.22) can be written,

$$\hat{\varphi}_{MC}^N(Z, \Xi) = \frac{1}{N} \sum_{i=1}^{N} \varphi(z^i, \xi^i). \quad (3.25)$$
Now, consider the Rao-Blackwellized estimator,
\[
\hat{\varphi}^N_{RB}(\Xi) = \mathbb{E}[\hat{\varphi}^N_{MC}(Z, \Xi) \mid \Xi] = \frac{1}{N} \sum_{i=1}^{N} \varphi^c(\xi^i),
\]  
(3.26)
where we have introduced the function \( \varphi^c(\xi) = \mathbb{E}[\varphi(z, \xi) \mid \xi] \). Note that (3.26) depends only on \( \Xi \). Similarly to (3.23), the variance of the estimator (3.26) is given by
\[
\text{Var}(\hat{\varphi}^N_{RB}(\Xi)) = \frac{\text{Var}(\varphi^c(\xi))}{N}.
\]  
(3.27)
From the law of total variance, it follows that
\[
\text{Var}(\varphi(z, \xi)) = \text{Var}(\mathbb{E}[\varphi(z, \xi) \mid \xi]) + \mathbb{E}\left[\text{Var}(\varphi(z, \xi) \mid \xi)\right] \geq 0.
\]  
(3.28)
The term on the left-hand-side corresponds to the variance of the vanilla Monte Carlo estimator (3.22) and the first term on the right-hand-side corresponds to the variance of the Rao-Blackwellized estimator (3.26). Since the second term on the right side is non-negative, it follows that (3.26) dominates (3.22), as claimed.

For the Rao-Blackwellized estimator (3.26) to be practical, it is necessary that the conditional expectation,
\[
\varphi^c(\xi) = \mathbb{E}[\varphi(z, \xi) \mid \xi] = \int \varphi(z, \xi) \gamma(z \mid \xi) \, dz,
\]  
(3.29)
can be computed efficiently. Hence, Rao-Blackwellization is useful when, in some sense, part of the integration problem in (3.1) is analytically tractable. In fact, if we set \( Z = X \) (and thus \( \Xi = \emptyset \)), the Rao-Blackwellized estimator (3.26) is given by \( \hat{\varphi}^N_{RB} = \mathbb{E}[\varphi(x)] \).

Clearly, if it is intractable to solve the integration problem in (3.1) in the first place, then computing this “fully Rao-Blackwellized” estimator is also intractable (since they coincide). Hence, there is a trade-off between using Monte Carlo methods to construct randomized estimators, and the application of Rao-Blackwellization to these estimators. The general idea that will be applied in this thesis is to make use of Rao-Blackwellization to an as large degree as possible; see Papers E and F.

In the context of SMC, Rao-Blackwellization is often used to analytically marginalize over part of the state-space. That is, similarly to (3.24), the state is split into two components, \( x_t = (z_t, \xi_t) \). The component \( \xi_t \) is represented using particles, whereas \( z_t \) is marginalized. This results in the so called Rao-Blackwellized particle filter (RBPF) (Chen and Liu, 2000; Doucet et al., 2000; Schön et al., 2005). The most well-known application of the RBPF is for a class of SSMs in which \( z_t \) is conditionally linear Gaussian. The aforementioned marginalization then amounts to running a conditional Kalman filter (Kalman, 1960) for each particle to marginalize \( z_t \).

For the vanilla Monte Carlo method, the variance reduction offered by Rao-Blackwellization is straightforwardly quantified by considering a decomposition of variance as in (3.28). This analysis, however, does not apply to more advanced Rao-Blackwellized Monte Carlo methods, such as the RBPF. This issue is addressed in Paper G, where the RBPF is analysed and a variance reduction expression akin to (3.28) is given.
This chapter concludes the first part of the thesis and points out possible directions for future work. Note, however, that more detailed discussions can be found in the concluding sections of the individual papers in Part II of the thesis.

4.1 Conclusions and future work

The contributions of the thesis can be grouped into two categories – those based on PGAS and those based on Rao-Blackwellization.

The PGAS algorithm has been found to be a useful tool for a range of tasks related to inference and learning of dynamical systems. Various algorithms have been developed around PGAS, addressing both Bayesian and maximum-likelihood-based learning as well as state inference. It has been illustrated how PGAS can be used to solve the classical problem of Wiener system identification in a challenging setting. Basic convergence results have been obtained for PGAS. However, it remains a topic for future work to establish stronger and more explicit ergodicity results, providing informative rates of convergence of the algorithm.

Another direction for future work is to adapt PGAS to certain model classes for which the basic algorithm is not applicable. PGAS relies heavily on so called ancestor sampling. To implement this procedure, it is necessary to evaluate point-wise the transition density function of the model under study (see Papers A and B for details). However, for many models encountered in practice this is not possible. Indeed, the transition density function may not even exist! One option is to study these specific models in more detail and thereby try to modify the PGAS algorithm so that it becomes applicable in these scenarios.

Another possibility, however, is to note that many of the models for which ancestor sam-
pling is problematic can be reformulated as non-Markovian latent variable models (see Section 4.6 of Paper A). This is interesting, since we have shown that PGAS can be useful for inference and learning of precisely such non-Markovian models. Hence, this opens up for using PGAS also for the aforementioned model classes, for which ancestor sampling is not directly possible. This is an encouraging result, since there has not been much progress made in solving inference and learning problems for non-Markovian models. However, it remains to evaluate and to better understand the properties of the PGAS method when applied to these models.

There are, of course, other limitations of the PGAS method as well. In particular: (i) the method may converge slowly when there are strong dependencies between the states and the parameters of the model, and (ii) the method can only be used for batch data (i.e. offline). One interesting direction of future work is to investigate possible ways in which these limitations can be mitigated.

The contributions of the thesis which are based on Rao-Blackwellization include the development of a Rao-Blackwellized particle smoother (RBPS) and a method for online Bayesian parameter estimation. Furthermore, the asymptotic variance of the RBPF has been analysed and compared to that of the standard particle filter. An interesting topic for future work is to establish a similar variance reduction result for the proposed RBPS, i.e. to answer the question of how much we benefit from Rao-Blackwellization when addressing the smoothing problem.

The proposed method for online Bayesian parameter estimation is based on the RBPF. The algorithm can be used for identification of nonlinear SSMs with an affine dependence on the parameters. A Gaussian mixture representation of the posterior parameter distribution \( p(\theta \mid y_{1:t}) \) is maintained. To mitigate the so called path degeneracy problem, which prevents accurate learning of the model parameters using a standard RBPF, a mixing step is incorporated in the algorithm. Unfortunately, this gives rise to a computational complexity which scales quadratically with the number of particles. Future work is needed in order to obtain a computationally more efficient algorithm. Furthermore, it would be interesting to modify the algorithm so that it can be used also for non-affine models. In particular, it should be possible to use the same idea for any model belonging to the exponential family.

Finally, one direction of future work which applies to all the methods discussed throughout the thesis, is to evaluate them in real applications. So far, the methods have primarily been tested in simulation studies. While this is of course very useful as a first level of evaluation, it is not until the methods are used to solve real and relevant problems that their true values can be determined.

### 4.2 Further reading

The learning problem is discussed in a general setting in the textbooks by Hastie et al. (2009) and Barber (2012). The latter contains one part dedicated to dynamical systems. Ljung (1999); Söderström and Stoica (1989) provide a detailed coverage of the problem from a system identification point of view. For readers who are interested in learning
more about Monte Carlo methods, the books by Robert and Casella (2004); Liu (2001) provide a thorough introduction. MCMC and SMC methods are discussed in detail in the collections by Brooks et al. (2011) and Doucet et al. (2001), respectively. See also Part VII of (Crisan and Rozovskii, 2011). Del Moral (2004) provides a extensive collection of convergence results for SMC. Finally, the textbooks by Cappé et al. (2005); Schön and Lindsten (2013) focus on using Monte Carlo methods for inference and learning of dynamical systems.
Bibliography


F. Lindsten, P.-J. Nordlund, and F. Gustafsson. Conflict detection metrics for aircraft sense and avoid systems. In *Proceedings of the 7th IFAC Symposium on Fault Detection,*
Supervision and Safety of Technical Processes (SafeProcess), Barcelona, Spain, July 2009.


T. B. Schön and F. Lindsten. *Computational Learning in Dynamical Systems*. 2013. (forthcoming, draft manuscript is available from the authors).


Paper A

Backward simulation methods for Monte Carlo statistical inference

Authors: Fredrik Lindsten and Thomas B. Schön

Edited version of the paper:
Backward simulation methods for Monte Carlo statistical inference

Fredrik Lindsten and Thomas B. Schön

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
{lindsten,schon}@isy.liu.se

Abstract

Monte Carlo methods, in particular those based on Markov chains and on interacting particle systems, are by now tools that are routinely used in machine learning. These methods have had a profound impact on statistical inference in a wide range of application areas where probabilistic models are used. Moreover, there are many algorithms in machine learning which are based on the idea of processing the data sequentially, first in the forward direction and then in the backward direction. In this monograph we will review a branch of Monte Carlo methods based on the forward-backward idea, referred to as backward simulators. These methods are useful for learning and inference in probabilistic models containing latent stochastic processes. The theory and practice of backward simulation algorithms have undergone a significant development in recent years and the algorithms keep finding new applications. The foundation for these methods is sequential Monte Carlo (SMC). SMC-based backward simulators are capable of addressing smoothing problems in sequential latent variable models, such as general, nonlinear/non-Gaussian state-space models (SSMs). However, we will also clearly show that the underlying backward simulation idea is by no means restricted to SSMs. Furthermore, backward simulation plays an important role in recent developments of Markov chain Monte Carlo (MCMC) methods. Particle MCMC is a systematic way of using SMC within MCMC. In this framework, backward simulation gives us a way to significantly improve the performance of the samplers. We review and discuss several related backward-simulation-based methods for state inference as well as learning of static parameters, both using a frequentistic and a Bayesian approach.

1 Introduction

A basic strategy to address many inferential problems in machine learning is to process data sequentially, first in the forward direction and then in the backward direction. Examples of this approach are the well known forward-backward algorithm for hidden Markov
models (HMMs) and the Rauch-Tung-Striebel smoother (Rauch et al., 1965) for linear Gaussian state-space models. Moreover, two decades of research on sequential Monte Carlo and Markov chain Monte Carlo has enabled inference in increasingly more challenging models. Many developments have been made in order to make use of the forward-backward idea together with these Monte Carlo methods, providing inferential techniques collectively referred to as backward simulation. This monograph provides a unifying view of these methods. In this introductory section we review some relevant background material and also derive a first backward simulator for the special case of linear Gaussian state-space models.

1.1 Background and motivation

For over half a century, Monte Carlo methods have been recognized as potent tools for statistical inference in complex probabilistic models; see Metropolis and Ulam (1949) for an early discussion. A continuous development and refinement of these methods have enabled inference in increasingly more challenging models. A key milestone in this development was the introduction of Markov chain Monte Carlo (MCMC) methods through the inventions of the Metropolis-Hastings algorithm by Metropolis et al. (1953); Hastings (1970) and the Gibbs sampler by Geman and Geman (1984). Parallel to this, sequential importance sampling (Handschin and Mayne, 1969) and sampling/importance resampling (Rubin, 1987) laid the foundation of sequential Monte Carlo (SMC). In its modern form, SMC was first introduced by Stewart and McCarty (1992); Gordon et al. (1993). During the 90’s, several independent developments were made by, among others, Kitagawa (1996) and Isard and Blake (1998). Recently, SMC and MCMC have been combined in a systematic manner through the developments of pseudo-marginal methods (Beaumont, 2003; Andrieu and Roberts, 2009) and particle MCMC (Andrieu et al., 2010).

Backward simulation is a strategy which is useful as a Monte Carlo method for learning of probabilistic models containing latent stochastic processes. In particular, we will consider inference in dynamical systems, i.e. systems that evolve over time. Dynamical systems play a central role in a wide range of scientific fields, such as signal processing, automatic control, epidemiology and econometrics, to mention a few.

One of the most widely used models of a dynamical system is the state-space model (SSM), reviewed in more detail in Sections 1.4–1.6. The structure of an SSM can be seen as influenced by the notion of a physical system. At each time $t$, the system is assumed to be in a certain state $x_t$. The state contains all relevant information about the system; i.e. if we would know the state of the system we would have full insight into its internal condition. However, the state is typically not known. Instead, we measure some quantity $y_t$ which depend on the state in some way. Given a sequence of observations $y_{1:T} \triangleq (y_1, \ldots, y_T)$, we seek to draw inference about the latent state process $x_{1:T}$ (state inference), as well as about unknown static parameters of the model (parameter inference).

The class of SSMs will play a central role in this monograph. Indeed, many of the inferential methods that we will review have been developed explicitly for SSMs. However, as will become apparent in Sections 4 and 5, most of the methods are more general and can be used for learning interesting models outside the class of SSMs.
Backward simulation is based on the forward-backward idea. That is, the data is processed first in the forward direction and then in the backward direction. In the backward pass, the state process is simulated backward in time, i.e. by first simulating $x_T$, then $x_{T-1}$ etc., until a complete state trajectory $x_{1:T}$ is generated. This procedure gives us a tool to address the state smoothing problem in models for which no closed form solution is available. This is done by simulating multiple backward trajectories from the smoothing distribution, i.e. conditionally on the observations $y_{1:T}$, which can then be used for Monte Carlo integration. State smoothing is of key relevance, e.g. to obtain refined state estimates in offline settings. Furthermore, it lies at the core of many parameter inference methods (see Section 1.5) and it can be used to address problems in optimal control (see Section 4.1).

Backward simulation is also useful in MCMC, as a way of grouping variables to improve the mixing of the sampler. A common way to construct an MCMC sampler for an SSM is to sample the state-variables $x_t$, for different $t$, one at a time (referred to as single-state sampling). However, since the states are often strongly dependent across time, this can lead to poor performance. Backward simulation provides a mean of grouping the state variables and sampling the entire trajectory $x_{1:T}$ as one entity. As we will illustrate in Section 1.3, this can lead to a considerable improvement upon the single-state sampler.

In Section 1.7 we will derive a first backward simulator for the class of linear Gaussian state-space (LGSS) models. Apart from LGSS models, exact backward simulation is tractable, basically only for finite state-space HMMs (see also Section 4.1.1). The main focus in this monograph will be on models outside these restricted classes, for which exact backward simulation is not possible. Instead, we will make use of SMC (and MCMC) to enable backward simulation in challenging probabilistic models, such as nonlinear/non-Gaussian SSMs, as well as more general non-Markovian latent variable models.

### 1.2 Notation and definitions

For any sequence $\{x_k\}_{k \in \mathbb{N}}$ and integers $m \leq n$ we write $x_{m:n} \triangleq (x_m, \ldots, x_n)$. We let $\wedge$ be the minimum operator, i.e. $a \wedge b \triangleq \min(a, b)$. For a matrix $A$, the matrix transpose is written as $A^T$. For two probability distributions $\mu_1$ and $\mu_2$, the total variation distance is given by $\|\mu_1 - \mu_2\|_{TV} \triangleq \sup_A |\mu_1(A) - \mu_2(A)|$. A Dirac point-mass located at some point $x'$ is denoted as $\delta_{x'}(dx)$. We write $X \sim \mu$ to mean that the random variable $X$ is either distributed according to $\mu$, or sampled from $\mu$. The uniform probability distribution on the interval $[a, b]$ is written as $\mathcal{U}([a, b])$. $\text{Cat}(\{p_i\}_{i=1}^n)$, with $\sum_{i=1}^n p_i = 1$, is the categorical (i.e. discrete) probability distribution on the set $\{1, \ldots, n\}$, with probabilities $\{p_i\}_{i=1}^n$. Finally, $N(m, \Sigma)$ and $\mathcal{N}(x; m, \Sigma)$ are the Gaussian (i.e. normal) probability distribution and density function, respectively, with mean vector $m$, covariance matrix $\Sigma$ and argument $x$.

### 1.3 A preview example

Before we continue with this section on background theory, we consider an example to illustrate the potential benefit of using backward simulation. A simple stochastic volatility
SSM is given by,

\[
x_{t+1} = ax_t + v_t, \quad v_t \sim \mathcal{N}(0, q), \quad (1a)
\]

\[
y_t = e_t \exp \left( \frac{1}{2} x_t \right), \quad e_t \sim \mathcal{N}(0, 1), \quad (1b)
\]

where the state process \( \{x_t\}_{t \geq 1} \) is latent and observations are made only via the measurement process \( \{y_t\}_{t \geq 1} \). Similar models have been used to generalize the Black-Scholes option pricing equation to allow for the variance to change over time (Chesney and Scott, 1989; Melino and Turnbull, 1990). The same model was used by de Jong and Shephard (1995) to illustrate the poor mixing of a single-state Gibbs sampler; an example which is replicated here.

For simplicity, we assume that the parameters \( a = 0.99 \) and \( q = 0.01 \) are known. We seek the density \( p(x_1:T | y_1:T) \), i.e. the conditional density of the state process \( x_{1:T} \) given a sequence of observations \( y_{1:T} \) for some fixed final time point \( T \). This conditional density is referred to as the joint smoothing density (JSD). For the model under study, the JSD is not available in closed form due to the nonlinear measurement equation \((1b)\). To remedy this, we construct an MCMC method to approximately sample from it. MCMC will be reviewed in more detail in Section 2.2. However, the basic idea is to simulate a Markov chain which is constructed in such a way that it admits the target distribution as limiting distribution. The sample path from the Markov chain can then be used to draw inference about the target density \( p(x_1:T | y_1:T) \).

As an initial attempt, we try a single-state Gibbs sampler. That is, we sample each state \( x_t \) conditionally on \( \{x_1:t-1, x_{t+1}:T\} \) (and the observations \( y_1:T \)). At each iteration of the Gibbs sampler we thus simulate according to,

\[
x'_1 \sim p(x_1 | x_2:T, y_{1:T});
\]

\[
\vdots
\]

\[
x'_t \sim p(x_t | x'_{1:t-1}, x_{t+1}:T, y_{1:T});
\]

\[
\vdots
\]

\[
x'_T \sim p(x_T | x'_{1:T-1}, y_{1:T}).
\]

This procedure will leave \( p(x_{1:T} | y_{1:T}) \) invariant (see Section 2.2 for more on Gibbs sampling) and it results in a valid MCMC sampler. The conditional densities \( p(x_t | x_{1:t-1}, x_{t+1}:T, y_{1:T}) \) are not available in closed form. However, for the model \((1)\), they are log-concave and we can employ the efficient rejection sampling strategy by Wild and Gilks (1993) to sample exactly from these distributions.

The single-state Gibbs sampler will indeed converge to samples from \( p(x_{1:T} | y_{1:T}) \). However, it is well recognized that single-state samplers can suffer from poor mixing, due to the often strong dependencies between consecutive state variables. That is, the convergence can be slow in the sense that we need to iterate the above sampling scheme a large number of times to get reliable samples.

To analyze this, we generate \( T = 100 \) samples from the model \((1)\) and run the Gibbs sampler for 100 000 iterations (in each iteration, we loop over all the state variables for
1. Introduction

Figure 1: (Top left) Part of sample path for the single-state Gibbs sampler; (Top right) Part of sample path for PGBS; (Bottom) Empirical ACF for $x_{50}$ for the single-state Gibbs sampler and for PGBS using $N = 15$ particles.

The first 10,000 iterations are discarded, to avoid transient effects. We then compute the empirical autocorrelation function (ACF) of the state $x_{50}$, which is given in Figure 1. As can be seen, the ACF decreases very slowly, indicating a poorly mixing Gibbs kernel. This simply reflects the fact that, when the state variables are highly correlated, the single-state sampler will be inefficient at exploring the state-space. This is a common and well recognized problem when addressing the state inference problem for SSMs.

One way to remedy this is to group the variables and sample a full state trajectory $x_{1:T}$ jointly. This is what a backward simulator aims to accomplish. Grouping variables in a Gibbs sampler will in general improve upon the mixing of the single-state sampler (Liu, 2001, Section 6.7), and in practice the improvement can be quite considerable.

To illustrate this, we have included the ACF for a backward-simulation-based method in Figure 1. Since the model (1) is nonlinear, exact backward simulation is not possible. Instead, the results reported here are from a backward simulator based on SMC, using (only) $N = 15$ particles. The specific method that we have used is denoted as particle Gibbs with backward simulation (PGBS), and it will be discussed in detail in Section 5.4. For the PGBS, the ACF drops off much more rapidly, indicating a more efficient sampler. Furthermore, a key property of PGBS is that, despite the fact that it relies on a crude SMC
approximation, it does not alter the stationary distribution of the Gibbs sampler, nor does
it introduce any additional bias. That is, PGBS will, just as the single-state Gibbs sampler,
target the exact JSD \( p(x_{1:T} \mid y_{1:T}) \). This property is known as exact
approximation, a concept that we will return to in Section 5.

1.4 State-space models

State-space models (SSMs) are commonly used to model time series and dynamical sys-
tems. Additionally, many models that are not sequential “by nature” can also be written
on state-space form. It is a comprehensive and important class of models, and it serves as
a good starting point for introducing the concepts that will be discussed throughout this
monograph.

We consider discrete-time SSMs on a general state-space \( X \). The system state is a Markov
process \( \{x_t\}_{t \geq 1} \) on \( X \), evolving according to a Markov transition kernel \( F(dx_{t+1} \mid x_t) \)
and with initial distribution \( \nu(dx_1) \). The state \( x_t \) is assumed to summarize all relevant
information about the system at time \( t \). However, the state process is latent and it is
observed only implicitly through the observations \( \{y_t\}_{t \geq 1} \), taking values in some set \( Y \).
Given \( x_t \), the measurement \( y_t \) is conditionally independent of past and future states and
observations, and it is distributed according to a kernel \( G(dy_t \mid x_t) \). A graphical model,
illustrating the conditional dependencies in an SSM, is given in Figure 2.

We shall assume that the observation kernel \( G \) admits a probability density \( g \) w.r.t. some
dominating measure, which we simply denote \( dy \). Such models are referred to as partially
dominated. If, in addition, the transition kernel \( F \) admits a density \( f \) and the initial
distribution \( \nu \) admits a density \( \mu \), both w.r.t. some dominating measure \( dx \), the model is
called fully dominated. In summary, a fully dominated SSM can be expressed as,

\[
\begin{align*}
x_{t+1} &\sim f(x_{t+1} \mid x_t), \\
y_t &\sim g(y_t \mid x_t),
\end{align*}
\]

and \( x_1 \sim \mu(x_1) \). Two examples of SSMs follow below.

Figure 2: Graphical model of an SSM. The white nodes represent latent variables
and the gray nodes represent observed variables.
Example 1: Finite state-space hidden Markov model
A finite state-space HMM, or simply HMM, is an SSM with X = \{1, \ldots, K\} for some finite K. The transition density (w.r.t. counting measure) can be summarized in a K × K transition matrix \(\Pi\), where the \((i,j)\)th entry is given by,
\[
\Pi_{i,j} = \mathbb{P}(x_{t+1} = j \mid x_t = i) = f(j \mid i).
\]
Hence, \(f(j \mid i)\) denotes the probability of moving from state \(i\) at time \(t\), to state \(j\) at time \(t + 1\).

Example 2: Additive noise model
In engineering applications, SSMs are often expressed on functional form with additive noise,
\[
\begin{align*}
x_{t+1} &= a(x_t) + v_t, \\
y_t &= c(x_t) + e_t,
\end{align*}
\]
for some functions \(a\) and \(c\). Here, the noises \(v_t\) and \(e_t\) are commonly referred to as process noise and measurement noise, respectively. If the noise distributions admit densities w.r.t. dominating measures, then the model is fully dominated. The transition density is then given by \(f(x_{t+1} \mid x_t) = p_{v_t}(x_{t+1} - a(x_t))\) and similarly for the observation density.

Throughout this monograph, we will mostly be concerned with fully dominated SSMs and therefore do most of our derivations in terms of probability densities. There are, however, several examples of interesting models that are degenerate, i.e. that are not fully dominated. We will return to this in the sequel and discuss how it affects the methods presented in this monograph.

1.5 Parameter learning in SSMs
The basic inference problem for SSMs is typically that of state inference, i.e. to infer the latent states given measurements from the system. In fact, even when the actual task is to learn a model of the system dynamics, state inference tends to play a crucial role as an intermediate step of the learning algorithm. To illustrate this, assume that the SSM (2) is parameterized by some unknown parameter \(\theta \in \Theta\),
\[
\begin{align*}
x_{t+1} &\sim f_\theta(x_{t+1} \mid x_t), \\
y_t &\sim g_\theta(y_t \mid x_t),
\end{align*}
\]
and \(x_1 \sim \mu_\theta(x_1)\). Given a batch of measurements \(y_{1:T}\), we wish to draw inference about \(\theta\). In the Bayesian setting, a prior distribution \(\pi(\theta)\) is assigned to the parameter and the learning problem amounts to computing the posterior distribution \(p(\theta \mid y_{1:T})\).

A complicating factor is that the likelihood \(p(y_{1:T} \mid \theta)\) in general cannot be computed in closed form. To address this difficulty, it is common to make use of data augmentation (van Dyk and Meng, 2001; Tanner and Wong, 1987). That is, we target the joint state and parameter posterior \(p(\theta, x_{1:T} \mid y_{1:T})\), rather than the marginal posterior \(p(\theta \mid y_{1:T})\). The latent states are thus viewed as auxiliary variables. This opens up for using Gibbs sampling (see Section 2.2), for instance by initializing \(\theta[0] \in \Theta\) and iterating;
(i) Draw $x_{1:T}[r] \sim p(x_{1:T} \mid \theta[r - 1], y_{1:T})$;
(ii) Draw $\theta[r] \sim p(\theta \mid x_{1:T}[r], y_{1:T})$.

Under weak assumptions, this procedure will generate a Markov chain $\{\theta[r], x_{1:T}[r]\}_{r \geq 1}$ with stationary distribution $p(\theta, x_{1:T} \mid y_{1:T})$. Consequently, the stationary distribution of the subchain $\{\theta[r]\}_{r \geq 1}$ will be the marginal parameter posterior distribution $p(\theta \mid y_{1:T})$. Note that Step (ii) of the above sampling scheme requires the computation of the JSD, for a fixed value of the parameter $\theta$. That is, we need to address an intermediate smoothing problem in order to implement this Gibbs sampler.

Data augmentation is commonly used also in the frequentistic setting. Assume that we, instead of the posterior distribution, seek the maximum likelihood estimator (MLE),

$$\hat{\theta}_{\text{ML}} = \arg \max_{\theta \in \Theta} \log p_{\theta}(y_{1:T}),$$

where $p_{\theta}(y_{1:T})$ is the likelihood of the observed data for a given value of the system parameter $\theta$. Again, since the log-likelihood $\log p_{\theta}(y_{1:T})$ is not available in closed form, direct maximization in (4) is problematic. Instead, we can make use of the expectation maximization (EM) algorithm by Dempster et al. (1977) (see also McLachlan and Krishnan (2008)). The EM algorithm is an iterative method, which maximizes $p_{\theta}(y_{1:T})$ by iteratively maximizing an auxiliary quantity,

$$Q(\theta, \theta') = \int \log p_{\theta}(x_{1:T}, y_{1:T}) p_{\theta'}(x_{1:T} \mid y_{1:T}) dx_{1:T}.$$  

The resulting sequence $\{\theta[r]\}_{r \geq 0}$ will, under weak assumptions, converge to a stationary point of the likelihood $p_{\theta}(y_{1:T})$ (Wu, 1983).

Using the conditional independence properties of an SSM, we can write the complete data log-likelihood as

$$\log p_{\theta}(x_{1:T}, y_{1:T}) = \log \mu_{\theta}(x_{1}) + \sum_{t=1}^{T} \log g_{\theta}(y_{t} \mid x_{t}) + \sum_{t=1}^{T-1} \log f_{\theta}(x_{t+1} \mid x_{t}).$$

From (5), we note that the auxiliary quantity is defined as the expectation of the expression (6) under the JSD. Hence, to carry out the E-step of the EM algorithm, we again need to address an intermediate smoothing problem for fixed values of the system parameters.

### 1.6 Smoothing recursions

As noted above, the JSD is a quantity of central interest for learning and inference problems in SSMs. It summarizes all the information about the latent states which is available
Table 1: Filtering and smoothing densities of particular interest.

<table>
<thead>
<tr>
<th>Density</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering</td>
<td>$p(x_t \mid y_{1:t})$</td>
</tr>
<tr>
<td>t ≤ T</td>
<td>$p(x_{1:T} \mid y_{1:T})$</td>
</tr>
<tr>
<td>Marginal smoothing (t ≤ T)</td>
<td>$p(x_t \mid y_{1:T})$</td>
</tr>
<tr>
<td>Fixed-interval smoothing (s &lt; t ≤ T)</td>
<td>$p(x_{s:t} \mid y_{1:T})$</td>
</tr>
<tr>
<td>Fixed-lag smoothing (ℓ fixed)</td>
<td>$p(x_{t-ℓ+1:t} \mid y_{1:t})$</td>
</tr>
</tbody>
</table>

The filtering and fixed-lag smoothing densities are marginals of the JSD at time $t$, $p(x_{1:t} \mid y_{1:t})$. In the observations. Many densities that arise in various state inference problems are given as marginals of the JSD. There are a few that are of particular interest, which we summarize in Table 1. To avoid a cluttered notation, we now drop the (possible) dependence on an unknown parameter $\theta$ from the notation and write the JSD as $p(x_{1:T} \mid y_{1:T})$.

As in (6), the conditional independence properties of an SSM implies that the complete data likelihood can be written as,

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

The JSD is related to the above expression by Bayes’ rule,

$$p(x_{1:T} \mid y_{1:T}) = \int p(x_{1:T}, y_{1:T}) \, dx_{1:T}.$$  

Despite the simplicity of this expression, it is of limited use in practice due to the high-dimensional integration needed to compute the normalization factor in the denominator. Instead, most practical methods (and in particular the ones discussed in this monograph) are based on a recursive evaluation of the JSD.

Again by using Bayes’ rule, we get the following two-step procedure,

$$p(x_{1:t} \mid y_{1:t}) = \frac{g(y_t \mid x_t)p(x_{1:t} \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})},$$  

$$p(x_{1:t+1} \mid y_{1:t}) = f(x_{t+1} \mid x_t)p(x_{1:t} \mid y_{1:t}).$$  

The above equations will be denoted as the forward recursion for the JSD, since they evolve forward in time. Step (9a) is often referred to as the measurement update, since the current measurement $y_t$ is taken into account. Step (9b) is known as the time update, moving the density forward in time, from $t$ to $t + 1$.

An interesting fact about SSMs is that, conditioned on $y_{1:T}$, the state process $\{x_t\}_{t=1}^T$ is an inhomogeneous Markov process. Under weak assumptions (see Cappé et al., 2005, Section 3.3.2 for details), the same holds true for the time-reversed chain, starting at time $T$ and evolving backward in time according to the so called backward kernel,

$$B_t(A \mid x_{t+1}) \triangleq \mathbb{P}(x_t \in A \mid x_{t+1}, y_{1:T}).$$  

Note that the backward kernel is time inhomogeneous. In the general case, it is not possi-
ble to give an explicit expression for the backward kernel. However, for a fully dominated model, this can always be done, and its density is given by

\[ p(x_t \mid x_{t+1}, y_{1:T}) = \frac{f(x_{t+1} \mid x_t)p(x_t \mid y_{1:t})}{\int f(x_{t+1} \mid x_t)p(x_t \mid y_{1:t}) \, dx_t}. \]  

(11)

Furthermore, from the conditional independence properties of an SSM, it also holds that

\[ p(x_t \mid x_{t+1}, y_{1:T}) = p(x_t \mid x_{t+1}, y_{1:t}). \]

Using the backward kernel, we get an alternative recursion for the JSD, evolving backward in time,

\[ p(x_{t:T} \mid y_{1:T}) = p(x_t \mid x_{t+1}, y_{1:t})p(x_{t+1:T} \mid y_{1:T}), \]  

(12)

starting with the filtering density at time \( T \), \( p(x_T \mid y_{1:T}) \). This is known as the backward recursion. At time \( t = 1 \), the JSD for the time interval \( 1, \ldots, T \) is obtained.

The backward kernel density at time \( t \) depends only on the transition density \( f(x_{t+1} \mid x_t) \) and on the filtering density \( p(x_t \mid y_{1:t}) \), a property which is of key relevance. Hence, to utilise the backward recursion (12) for computing the JSD, the filtering densities must first be computed for \( t = 1, \ldots, T \). Consequently, this procedure is generally called forward filtering/backward smoothing.

### 1.7 Backward simulation in linear Gaussian SSMs

An important special case of (2) is the class of linear Gaussian state-space models. A functional representation of an LGSS model is given by,

\[ x_{t+1} = Ax_t + v_t, \quad v_t \sim \mathcal{N}(0, Q), \]  

(13a)

\[ y_t = Cx_t + e_t, \quad e_t \sim \mathcal{N}(0, R). \]  

(13b)

Here, \( y_t \) is an \( n_y \)-dimensional vector of observations, \( x_t \) is an \( n_x \)-dimensional state vector and the system matrices \( A \) and \( C \) are of appropriate dimensions. The process and measurement noises are multivariate Gaussian with zero means and covariances \( Q \) and \( R \), respectively.

---

**Example 3: Partially or fully dominated SSM**

Assume that the measurement noise covariance \( R \) in (13b) is full rank. Then, the observation kernel is Gaussian and dominated by Lebesgue measure. Hence, the model is partially dominated. If, in addition, the process noise covariance \( Q \) in (13a) is full rank, then the transition kernel is also Gaussian and dominated by Lebesgue measure. In this case, the model is fully dominated.

However, for singular \( Q \) the model is degenerate (i.e., not fully dominated). Rank deficient process noise covariances arise in many applications, for instance if there is a physical connection between some of the states (such as between position and velocity).

---

A fully dominated LGSS model can equivalently be expressed as in (2) with,

\[ f(x_{t+1} \mid x_t) = \mathcal{N}(x_{t+1} ; Ax_t, Q), \]  

(14a)

\[ g(y_t \mid x_t) = \mathcal{N}(y_t ; Cx_t, R). \]  

(14b)
LGSS models are without doubt one of the most important and well studied classes of SSMs. There are basically two reasons for this. First, LGSS models provide sufficiently accurate descriptions of many interesting dynamical systems. Second, LGSS models are one of the few model classes, simple enough to allow for a fully analytical treatment.

When addressing inferential problems for SSMs, we are often asked to generate samples from the JSD, typically as part of an MCMC sampler used to learn a model of the system dynamics, as discussed above. For an LGSS model, the JSD is Gaussian and it can be computed using Kalman filtering and smoothing techniques (see e.g. Kailath et al. (2000)). Hence, we can make use of standard results for Gaussian distributions to generate a sample from
\[ p(x_{1:T} \mid y_{1:T}) \]. This is possible for small \( T \), but for increasing \( T \) it soon becomes infeasible due to the large matrix inversions involved.

To address this issue, it was recognized by Carter and Kohn (1994); Frühwirth-Schnatter (1994) that we can instead use the backward recursion (12). It follows that the JSD can be factorized as,
\[
p(x_{1:T} \mid y_{1:T}) = \left( \prod_{t=1}^{T-1} p(x_t \mid x_{t+1}, y_{1:t}) \right) p(x_T \mid y_{1:T})
\]

Initially, we generate a sample from the filtering density at time \( T \),
\[
\tilde{x}_T \sim p(x_T \mid y_{1:T}).
\]
We then, successively, augment this backward trajectory by generating samples from the backward kernel,
\[
\tilde{x}_t \sim p(x_t \mid \tilde{x}_{t+1}, y_{1:t}),
\]
for \( t = T - 1, \ldots, 1 \). After a complete backward sweep, the backward trajectory \( \tilde{x}_{1:T} \) is (by construction) a realization from the JSD (15).

To compute the backward kernel, we first run a forward filter to find the filtering densities \( p(x_t \mid y_{1:t}) \) for \( t = 1, \ldots, T \). For an LGSS model, this is done by a standard Kalman filter (Kalman, 1960). It follows that the filtering densities are Gaussian according to,
\[
p(x_t \mid y_{1:t}) = \mathcal{N}(x_t; \hat{x}_{t|t}, P_{t|t}),
\]
for some tractable sequences of mean vectors \( \{\hat{x}_{t|t}\}_{t \geq 1} \) and covariance matrices \( \{P_{t|t}\}_{t \geq 1} \), respectively. From (14a), we note that the transition density function is Gaussian and affine in \( x_t \). Using (11), (17) and standard results on affine transformations of Gaussian variables, it then follows that
\[
p(x_t \mid x_{t+1}, y_{1:t}) = \mathcal{N}(x_t; \mu_t, M_t),
\]
with
\[
\mu_t = \hat{x}_{t|t} + P_{t|t}A^T(Q + AP_{t|t}A^T)^{-1}(x_{t+1} - A\hat{x}_{t|t}),
\]
\[
M_t = P_{t|t} - P_{t|t}A^T(Q + AP_{t|t}A^T)^{-1}AP_{t|t}.
\]
Note that, if more than one sample is desired, multiple backward trajectories can be generated independently, without having to rerun the forward Kalman filter. We illustrate the backward simulator in the example below.
Figure 3: Histograms of $\{x_t^j\}_{j=1}^M$ for $t = 1$, $t = 25$ and $t = 50$ (from left to right). The true marginal smoothing densities $p(x_t \mid y_{1:T})$ are shown as black lines.

Example 4

To illustrate the possibility of generating samples from the JSD using backward simulation, we consider a first order LGSS model,

$$
\begin{align*}
x_{t+1} &= 0.9x_t + v_t, \\
y_t &= x_t + e_t,
\end{align*}
$$

and $x_1 \sim \mathcal{N}(x_1; 0, 10)$. We simulate $T = 50$ samples $y_{1:T}$ from the model. Since the model is linear Gaussian, the marginal smoothing densities $p(x_t \mid y_{1:T})$ can be computed by running a Kalman filter followed by a Rauch-Tung-Striebel smoother (Rauch et al., 1965). However, we can also generate samples from the JSD $p(x_1:T \mid y_{1:T})$ by running a backward simulator. We simulate $M = 5000$ independent trajectories $\{x^j_{1:T}\}_{j=1}^M$, by first running a Kalman filter and then repeating the backward simulation procedure given by (16) and (18) $M$ times. Histograms over the simulated states at three specific time points, $t = 1$, $t = 25$ and $t = 50$, are given in Figure 3. As expected, the histograms are in close agreement with the true marginal smoothing distributions.

The strategy given by (16), i.e. to sequentially sample (either exactly or approximately) from the backward kernel to generate a realization from the JSD, is what we collectively refer to as backward simulation. We will now leave the world of LGSS models. In the remainder of this monograph we address backward simulation for general nonlinear/non-Gaussian models. In these cases, the backward kernels will in general not be available in closed form. Instead, we will rely on SMC approximations of the kernels to carry out the backward simulation.

Before we leave this section, it should be noted that the backward simulator for LGSS models derived here is provided primarily to illustrate the concept. For LGSS models, more efficient samplers exist, e.g. based on disturbance simulation. See de Jong and Shephard (1995); Durbin and Koopman (2002); Wilkinson and Yeung (2002) for further details and extensions.
1.8 Outline

The rest of this monograph is organized as follows. Section 2 reviews the two main Monte Carlo methods that are used throughout the monograph, SMC and MCMC. The section is self-contained, but for obvious reasons it does not provide an in-depth coverage of these methods. Several references which may be useful for readers with no background in this area are given in Section 2.

Section 3 addresses SMC-based backward simulation for SSMs. The focus in this section is on smoothing in general nonlinear/non-Gaussian SSMs. More precisely, we discuss algorithms for generating state trajectories, approximately distributed according to the joint smoothing distribution. These algorithms can be categorized as particle smoothers. Hence, readers with particular interest in smoothing problems may want to focus their attention on this section. However, smoothing is also addressed in Section 5 (see in particular Section 5.7), and the methods presented there can be useful alternatives to the particle smoothers discussed in Section 3.

Section 4 generalizes the backward simulation idea to latent variable models outside the class of SSMs. A general backward simulator is introduced and we discuss its properties and the type of models for which it is applicable. As a special case of the general backward simulator, we derive a Rao-Blackwellized particle smoother for conditionally linear Gaussian SSMs.

In Section 5, we discuss backward simulation in the context of so called particle MCMC (PMCMC) methods. The focus in this section is on parameter inference, primarily in the Bayesian setting, but we also discuss PMCMC for maximum-likelihood-based inference. As mentioned above, the smoothing problem is also addressed. Finally, in Section 6 we conclude with a discussion about the various methods reviewed throughout this monograph and outline possible directions for future work.

2 Monte Carlo preliminaries

In this section we review the two main Monte Carlo tools on which the subsequent development is based; sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC).

2.1 Sequential Monte Carlo

For general nonlinear non-Gaussian SSMs the backward kernel (10) is not available on closed form. To address this issue, the basic idea that will be employed is to use an SMC approximation of the backward kernel. In this section, we review the basics of SMC applied to SSMs, also referred to as particle filtering.

Since the focus of this monograph is on backward simulation, and not SMC, we will only present a basic SMC sampler and discuss some of its important properties. For a more in-depth treatment, we refer to one of the many tutorials and textbooks on the topic, see e.g. Schön and Lindsten (2013); Doucet and Johansen (2011); Gustafsson (2010); Cappé et al. (2007); Arulampalam et al. (2002); Doucet et al. (2001). A comprehensive collection of
weights are given by,

\[ w_i = \frac{p(x_1^i | y_1:t)}{r_t(x_1^i | y_1:t)}, \]

for \( i = 1, \ldots, N \). The weights can only be computed up to proportionality, since the normalization constant for the target density is unknown. However, this is easily coped with by normalizing the weights to sum to one. That is, we first compute the unnormalized importance weights

\[ \bar{w}_i = \frac{p(x_1^i, y_1:t)}{r_t(x_1^i, y_1:t)}, \]

and then set \( w_i = \bar{w}_i / \sum_{l=1}^{N} \bar{w}_t \).

Now, to allow for a sequential method, we construct the proposal density so that it factorizes according to,

\[
\begin{align*}
r_t(x_{1:t} | y_{1:t}) &= r_t(x_t | x_{t-1}, y_t)r_{t-1}(x_{1:t-1} | y_{1:t-1}) \\
&= r_1(x_1 | y_1) \prod_{s=2}^{t} r_s(x_s | x_{s-1}, y_s).
\end{align*}
\]

Hence, we can draw a sample from the proposal \( x_{1:t}^i \sim r_t(x_{1:t} | y_{1:t}) \) by first generating \( x_1^i \sim r_1(x_1 | y_1) \), then \( x_2^i \sim r_2(x_2 | x_1^i, y_2) \), etc. This implies that the importance weights are given by,

\[
w_i = \frac{p(x_{1:t}^i | y_{1:t})}{r_t(x_{1:t}^i | y_{1:t})} \propto g(y_t | x_t^i)f(x_t^i | x_{t-1}^i) \frac{p(x_{1:t-1}^i | y_{1:t-1})}{p(x_{1:t-1}^i | y_{1:t-1})}.
\]

Hence, if we define the weight function

\[ W_t(x_{t-1}, x_t; y_t) = \frac{g(y_t | x_t)f(x_t | x_{t-1})}{r_t(x_t | x_{t-1}, y_t)}, \]

convergence results for SMC can be found in (Del Moral, 2004). It should be emphasized that the backward simulators that are derived in the subsequent sections indeed rely on SMC approximations of the backward kernels, but not directly on how these approximations are generated. Hence, it is straightforward to replace the basic sampler presented here with a more advanced SMC method, if desired.

### 2.1.1 The particle filter

SMC methods are based on importance sampling and resampling techniques to sequentially sample from a sequence of target densities. They can be seen as combinations of the sequential importance sampling (Handschin and Mayne, 1969) and the sampling/importance resampling (Rubin, 1987) algorithms. The name particle filter (PF) is often used interchangeably with SMC, though here we reserve it for the case when the sequence of target densities is given by \( p(x_{1:t} | y_{1:t}) \) for \( t = 1, 2, \ldots \).

In a standard importance sampler, targeting \( p(x_{1:t} | y_{1:t}) \), we generate \( N \) independent samples \( \{ x_{1:t}^i \}_{i=1}^{N} \) from some proposal density \( r_t(x_{1:t} | y_{1:t}) \). These samples are then assigned importance weights,

\[ w_i = \frac{p(x_1^i | y_1:t)}{r_t(x_1^i | y_1:t)}, \]

interchangeably with SMC, though here we reserve it for the case when the sequence of target densities is given by \( p(x_{1:t} | y_{1:t}) \) for \( t = 1, 2, \ldots \).
the importance weights can be computed as
\[ \tilde{w}_t^i = W_t(x_{t-1}^i, x_t^i; y_t)w_{t-1}^i, \]
and \( w_t^i = \frac{\tilde{w}_t^i}{\sum_{l=1}^N \tilde{w}_t^l} \). We thus obtain a sequential updating formula also for the importance weights. The initial weight function at time \( t = 1 \) is given by \( W_1(x_1; y_1) = g(y_1 | x_1)p(x_1)/r_1(x_1 | y_1) \). As indicated by the notation, the proposal at time \( t \) is allowed to depend on both the previous state \( x_{t-1} \) and on the current observation \( y_t \). This is important in practice, to be able to make good use of the available information when designing the proposal.

The samples \( \{x_{1:t}^i\}_{i=1}^N \) are called particles and the collection \( \{x_{1:t}^i, w_t^i\}_{i=1}^N \) is referred to as a weighted particle system. This weighted sample defines an empirical point-mass distribution on \( X_t \),
\[ \hat{p}_N(dx_{1:t} | y_{1:t}) \triangleq \sum_{i=1}^N w_t^i \delta_{x_{1:t}^i}(dx_{1:t}), \] (25)
which is an approximation of the target distribution. Equivalently, for any test function \( \varphi : X_T \rightarrow \mathbb{R} \) we can construct the estimator
\[ \hat{\varphi}_{t|t}^N \triangleq \sum_{i=1}^N w_t^i \varphi(x_{1:t}^i) \approx \mathbb{E}[\varphi(x_{1:t}) | y_{1:t}]. \] (26)
This estimator is consistent, as \( N \to \infty \), and a central limit theorem holds (see e.g. Del Moral (2004); Künsch (2005); Douc and Moulines (2008)).

The procedure resulting from sampling according to (21) and computing the weights (24) is the sequential importance sampler (Handschin and Mayne, 1969), dating back to the late 60’s. Though applicable for short data lengths, a serious drawback with this method is that the weight update (24) is unstable. More precisely, the variance of the normalized importance weights increases over time (Cappé et al., 2005, p. 232), typically at an exponential rate. This has the effect that all but one of the weights decreases to zero, and all emphasis is thus put on one of the particles (recall that the weights are normalized to sum to one). This does not come as a surprise, since we are in fact applying an importance sampler in a space with dimension increasing with \( t \). In high dimensions, even what appears to be a small discrepancy between the proposal and the target densities, will result in poor performance of the sampler (see e.g. Doucet and Johansen (2011) for an illustrative example).

A way to mitigate this problem was proposed by Stewart and McCarty (1992); Gordon et al. (1993), resulting in the first functional SMC sampler. The idea is to rejuvenate the sample by replicating particles with high weights and discarding particles with low weights. This is done by resampling the particle system, similarly to the sampling/importance resampling method (Rubin, 1987). Since the weighted particle system \( \{x_{1:t}^i, w_t^i\}_{i=1}^N \) approximates the target according to (25), we can generate a new, unweighted set of particles by sampling independently from (25). That is, we set \( \tilde{x}_{1:t}^j = x_{1:t}^i \) with probability \( w_t^i \), for \( j = 1, \ldots, N \). The equally weighted particle system \( \{\tilde{x}_{1:t}^j, \frac{1}{N}\}_{j=1}^N \) can then be
used to construct an estimator similarly to (26),

$$\hat{\varphi}^N_{1:t} \triangleq \frac{1}{N} \sum_{j=1}^{N} \varphi(\hat{x}^j_{1:t}) \approx \mathbb{E}[\varphi(x_{1:t}) \mid y_{1:t}].$$ (27)

It should be noted that the resampling introduces some additional variance, so (27) will be dominated by (26). However, when applied sequentially, the resampling is critical since it allows us to put focus on the promising particles and discard the improbable ones. We summarize the PF in Algorithm 1.

**Algorithm 1** Particle filter (all operations are for \(i = 1, \ldots, N\))

1: Draw \(x^i_1 \sim r_1(x_1 \mid y_1)\).
2: Compute \(\bar{w}^i_1 = W_1(x^i_1; y_1)\).
3: Normalize: set \(w^i_1 = \bar{w}^i_1 / \sum_{i=1}^{N} \bar{w}^i_1\).
4: for \(t \geq 2\) do
5: \hspace{1em} Resample with \(\mathbb{P}(\hat{x}^i_{t-1} = x^j_{t-1}) = w^j_{t-1}\).
6: \hspace{1em} Draw \(x^i_t \sim r_t(x_t \mid \hat{x}^i_{t-1}, y_t)\) and set \(x^i_{1:t} = \{\hat{x}^i_{1:t-1}, x^i_t\}\).
7: \hspace{1em} Compute \(\bar{w}^i_t = W_t(x^i_t, \hat{x}^i_{t-1}; y_t)\).
8: \hspace{1em} Normalize: set \(w^i_t = \bar{w}^i_t / \sum_{i=1}^{N} \bar{w}^i_t\).
9: end for

A common choice in practice is to run the PF with \(r_t(x_t \mid x_{t-1}, y_t) = f(x_t \mid x_{t-1})\), i.e. we propose samples according to the transition density function. This results in a vanilla method referred to as the bootstrap filter. If possible, it is recommended to use a proposal density which takes the current observation \(y_t\) into account. The optimal proposal is given by the conditional density \(r_t(x_t \mid x_{t-1}, y_t) = p(x_t \mid x_{t-1}, y_t)\). This choice of proposal density will minimize the variance of the incremental importance weights at time \(t\) (Doucet and Johansen, 2011). Unfortunately, it is often the case that this density is not available in practice, but various approximations can be used instead.

The resampling scheme outlined above is known as multinomial resampling. Alternative methods that introduce less variance exist, e.g. residual resampling (Whitley, 1994; Liu and Chen, 1998) and stratified resampling (Kitagawa, 1996). Furthermore, instead of resampling at every iteration, it is recommended to do so only when there is need for it. For this cause, it is common to introduce a measure of imbalance of the importance weights, such as the effective sample size (ESS) (Liu, 2001, Section 2.5). Resampling is then only carried out when the ESS drops below some prespecified threshold.

Another modification of the resampling step is used in the auxiliary particle filter (APF) by Pitt and Shephard (1999). The idea is to use an adjusted proposal distribution in the resampling step. This is accomplished by computing a set of weights \(\{w^i_t \nu^i_t\}_{i=1}^{N}\), and using these (after normalization) as probabilities in the resampling. Here, the adjustment weights \(\nu^i_t\) are user defined. They allow us to, for instance, take the observation \(y_{t+1}\) into account when resampling the particles at time \(t\). The fact that we modify the proposal for the resampling is compensated for in the importance weight computation. If the APF is run with an optimal proposal kernel and with adjustment weights given by \(\nu^i_t = p(y_{t+1} \mid x^i_t)\), the importance weights will be identically equal to one. In this case, the filter is said
to be fully adapted. See Pitt and Shephard (1999); Johansen and Doucet (2008); Douc et al. (2009) for more information about the APF and how the adjustment weights can be chosen.

2.1.2 Path degeneracy

As pointed out in the previous section, the PF targets the sequence of JSDs, \( p(x_{1:t} \mid y_{1:t}) \). By keeping track of the genealogy of the particle filter, i.e. the ancestral dependence of the particles, we obtain weighted particle trajectories \( \{x_{1:t}^i, w_t^i\}_{i=1}^N \). These provide an approximation, not only of the filtering distribution, but of the complete joint smoothing distribution as in (25) (Kitagawa, 1996). It thus appears as if the PF solves the general state inference problem, since any smoothing density can be attained from \( p(x_{1:t} \mid y_{1:t}) \) by marginalization. However, this is not truly the case.

To see why, let \( s \) be some fixed time point and assume that we apply a PF to approximate the marginal smoothing density \( p(x_s \mid y_{1:t}) \) for \( t \geq s \). This is given straightforwardly from (25) by simply discarding everything except the \( s \)th time point from the particle trajectories \( \{x_{1:t}^i\}_{i=1}^N \). Now, if \( t = s \) (i.e., we are in fact looking at the filtering density) we have \( N \) unique particles in the point-mass approximation. However, each time we resample the particle system, the unique number of particles at time \( s \) will decrease. This is in fact the purpose of the resampling, to remove particles with small weights and duplicate particles with large weights. Consequently, for large enough \( t \gg s \), all the particles \( \{x_{1:t}^i\}_{i=1}^N \) will share a common ancestor at time \( s \), due to the consecutive resampling steps. This problem, known as path degeneracy, is further illustrated in the example below.

--- Example 5: Path degeneracy
---

A PF with \( N = 30 \) particles is applied to a one-dimensional Gaussian random walk process measured in Gaussian noise. At time \( t = 50 \) the JSD is targeted by a weighted particle system \( \{x_{1:50}^i, w_{50}^i\}_{i=1}^{30} \). Figure 4 depicts the particle trajectories. As can be seen, all particles share a common ancestor at time \( s = 32 \).

Assume, for instance, that we are interested in the smoothed estimate of the initial state, \( \mathbb{E}[x_1 \mid y_{1:50}] \). We thus construct an estimator,

\[
\hat{x}_{1|50} = \sum_{i=1}^{30} w_{50}^i x_1^i,
\]

but since \( x_1^i \) are identical for all \( i = 1, \ldots, 30 \), this is in effect a Monte Carlo integration using a single sample. Hence, due to the path degeneracy, we can not expect to obtain an accurate estimate of \( \mathbb{E}[x_1 \mid y_{1:50}] \) from the PF.

Path degeneracy arises as an effect of resampling, but it should rather be thought of as a manifestation of a deeper problem, namely the degeneracy of the importance weights. In the sequential importance sampler, all weight will be put on a single sample for large enough \( t \), having a similar effect as path degeneracy. This issue is in fact to a large extent mitigated by introducing resampling.

Due to path degeneracy, the PF is in practice used mostly for filtering (hence the name) and fixed-lag smoothing. At time \( t \) or \( s \lesssim t \), the particle diversity is in general high,
leading to accurate approximations of the filtering and fixed-lag smoothing distributions. How SMC can be used to address the marginal and joint smoothing problems will be the topic of Section 3.

2.2 Markov chain Monte Carlo

Another comprehensive class of Monte Carlo methods is MCMC. In this section, we review some of the basic concepts of MCMC that are of key relevance. For further reading, see the tutorial papers by Andrieu et al. (2003); Spall (2003) or one of the many textbooks, e.g. Brooks et al. (2011); Robert and Casella (2004); Liu (2001).

2.2.1 Markov chains and limiting distributions

MCMC methods allow us to approximately generate samples from an arbitrary target distribution $f(z)$ on some space $Z$. More precisely, an MCMC sampler will simulate a Markov chain, or Markov process, $\{z[r]\}_{r \geq 0}$ which is constructed in such a way that the limiting distribution of the chain is given by $f(z)$. The sample path of the Markov chain can than be used to draw inference about the target distribution. For instance, the target can be the JSD $p(x_{1:T} \mid y_{1:T})$, which means that MCMC can be used as an alternative to SMC for addressing the state inference problem. However, we will primarily make use of MCMC in a different way, either as a component of SMC or vice versa. Hence, in this section, the methods are presented in a general setting, which is specialized to different scenarios in the consecutive sections.

A Markov chain $\{z[r]\}_{r \geq 0}$ is completely specified by its initial distribution $\mu(z)$ and its transition kernel $K(z' \mid z)$ (for simplicity, we assume that there exists a dominating measure and express all distributions in terms of their probability densities). Here, $K(z' \mid z)$
encodes the probability density for the consecutive state of the chain, denoted as \( z' \), given that the given that the current state is \( z \). That is, the chain can be simulated according to \( z[0] \sim \mu(z) \) and \( z[r] \sim K(z' \mid z[r - 1]) \) for \( r = 1, 2, \ldots \). As an example, the state process \( \{x_t\}_{t \geq 1} \) of the SSM (2) is a Markov chain. However, the view on this process is quite different, as \( \{x_t\}_{t \geq 1} \) is thought of as the internal state of a physical system evolving over time. On the contrary, the Markov chain \( \{z[r]\}_{r \geq 0} \) generated in an MCMC sampler is simulated as part of an inferential method, and it does not have a physical interpretation in the same sense.

The Markov chains encountered in MCMC have, by construction, a special stability property, namely that of a stationary distribution. That is, there exist some distribution \( g \) such that, if \( z[r] \) is marginally distributed according to \( g \), then so is \( z[r + 1] \) (and therefore all \( z[m] \) for \( m \geq r \)). If \( g \) is a stationary distribution for the Markov kernel \( K \), then we say that \( K \) leaves \( g \) invariant. Furthermore, if the marginals of the chain approaches the stationary distribution (in total variation), it is referred to as the limiting distribution of the chain. An important consequence of this convergence property is that the sample path average,

\[
\frac{1}{R} \sum_{r=0}^{R} \varphi(z[r]),
\]

converges almost surely to the expectation \( \mathbb{E}_g[\varphi(z)] = \int \varphi(z) g(z) \, dz \). This result, known as the ergodic theorem (see e.g. (Robert and Casella, 2004, Section 6.6)), ensures that the Markov chain can be used to compute Monte Carlo estimators of expectations under the stationary distribution.

There are two properties of the chain that are sufficient for a stationary distribution to also be a limiting distribution; irreducibility and aperiodicity. The Markov chain is said to be \( g \)-irreducible if, for any initial state, it has positive probability of entering any set which has a positive probability under \( g \). The chain is periodic if, with probability 1, certain subsets only can be visited at regularly spaced intervals. If the chain is not periodic, it is said to be aperiodic. For an irreducible and aperiodic chain with stationary distribution \( g \), the marginals converge to the stationary distribution.

**Theorem 1.** Let \( \{z[r]\}_{r \geq 0} \) be a \( g \)-irreducible and aperiodic Markov chain with stationary distribution \( g \). Then, for \( g \)-almost all starting points,

\[
\|\mathcal{L}(z[r] \in \cdot) - g(\cdot)\|_{TV} \to 0,
\]

as \( r \to \infty \), where \( \mathcal{L}(z[r] \in \cdot) \) is the marginal distribution of \( z[r] \) and \( \|\cdot\|_{TV} \) is the total variation norm.

**Proof:** See e.g. (Tierney, 1994, Theorem 1).

Stronger forms of convergence are also common in the MCMC literature, such as geometric ergodicity and uniform ergodicity. It is also possible to establish central limit theorems for the estimator (28). There is by now a well developed theory on Markov chains, safeguarding the theoretical validity of MCMC methods, see e.g. Tierney (1994); Robert and
A sample path from a Markov chain with a specific limiting distribution can be used to compute estimators according to (28). However, since we are interested in computing expectations under a given target distribution $f(z)$, it remains to construct the chain so that $f(z)$ is its limiting distribution. To accomplish this, we have freedom in the choice of transition kernel $K(z' \mid z)$, as long as the kernel is chosen in such a way that it is possible to simulate the Markov chain.

A property which is sufficient for $f(z)$ to be a stationary distribution of the chain is that the kernel satisfies detailed balance,

$$f(z)K(z' \mid z) = f(z')K(z \mid z') \quad \text{for} \ (z, z') \in \mathbb{Z}^2. \quad (29)$$

Based on this criterion, it is possible to construct a Markov kernel with the correct stationary distribution using an accept/reject procedure. This results in the Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970), which is the main tool in the MCMC toolbox.

The idea behind the MH algorithm is to generate samples from some arbitrary proposal kernel $q(z' \mid z)$. The simulation is followed by an accept/reject decision to make sure that detailed balance holds for the given target distribution. The MH acceptance probability, for a move from $z$ to $z'$, is given by

$$\rho(z, z') = 1 \wedge \frac{f(z')}{f(z)} \frac{q(z \mid z')}{q(z' \mid z)},$$

where $a \wedge b = \min(a, b)$. That is, with $z[r - 1]$ being the previous state of the Markov chain at iteration $r$, we proceed by sampling a candidate value $z' \sim q(z' \mid z[r - 1])$. With probability $\rho(z[r - 1], z')$, the sample is accepted and we set $z[r] = z'$. Otherwise, the sample is rejected and we set $z[r] = z[r - 1]$. It is interesting to note that the acceptance probability depends on the target density only through the ratio $f(z')/f(z)$. Hence, it is possible to run the MH algorithm even when the normalizing constant of the target density is unknown, which is key to its applicability in practice. The MH sampler is summarized in Algorithm 2.

**Algorithm 2** Metropolis-Hastings sampler

1. Set $z[0]$ arbitrarily.
2. for $r \geq 1$ do
3. Draw $z' \sim q(z' \mid z[r - 1])$.
4. With probability
   $$\rho(z[r - 1], z') = 1 \wedge \frac{f(z')}{f(z[r - 1])} \frac{q(z[r - 1] \mid z')}{q(z' \mid z[r - 1])},$$
   set $z[r] = z'$, otherwise set $z[r] = z[r - 1]$.
5. end for
The transition kernel given by the sampling procedure above is,
\[ K(dz' \mid z) = \rho(z, z')q(dz' \mid z) + (1 - \eta(z))\delta_z(dz'), \]  
(31)
with \( \eta(z) = \int \rho(z, z')q(z' \mid z)\,dz' \). It can be verified that this kernel satisfies the detailed balance condition (29). Consequently, \( f(z) \) is a stationary distribution of the chain. Under additional, mild, assumptions (basically, positivity of the proposal kernel), it can also be verified that the chain is irreducible and aperiodic, which by Theorem 1 implies that \( f(z) \) is the limiting distribution of the Markov chain.

In practice, the choice of the proposal kernel is critical. If the proposal is not chosen appropriately, the average acceptance probability can be very low. This results in that the chain gets “stuck” at the same value for many iterations. We say that the chain suffers from poor mixing. Furthermore, a high acceptance probability does not necessarily imply good mixing. Indeed, a proposal kernel which only makes small, local moves can yield a high acceptance probability, but at the same time a very poor exploration of the state space. To construct a proposal kernel with good mixing properties is particularly challenging in high dimensions. To address this issue, sophisticated proposal mechanisms have been developed for the MH algorithm. One approach is to construct a proposal kernel based on the simulation of a Hamiltonian dynamical system, leading to the so called Hamiltonian Monte Carlo method (Duane et al., 1987) (see also Neal (2011)). See Girolami and Calderhead (2011); Ishwaran (1999); Schmidt (2009) for applications and extensions of this method. Another approach is to make use of SMC within MCMC, which is the idea that underlies the particle MCMC framework by Andrieu et al. (2010). These methods will be discussed in more detail in Section 5.

### 2.2.3 Gibbs sampling

The Gibbs sampler (Geman and Geman, 1984; Gelfand and Smith, 1990) is an MCMC method of particular interest. It can be seen as a special case of the MH algorithm. However, it differs both in terms of basic methodology and historical motivation. Assume that the random variable of interest can be partitioned according to \( z = \{z_1, \ldots, z_p\} \), where the \( z_i \)s can be either uni- or multidimensional. A Markov kernel is then constructed by sampling each component from its full conditional under \( f \), i.e.
\[ z_i' \mid z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_p \sim f_i(z_i \mid z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_p). \]  
(32)
The sampling procedure given above leaves \( f(z) \) invariant, i.e. if \( z \) is distributed according to \( f \), then so is \( \{z_1, \ldots, z_{i-1}, z_i', z_{i+1}, \ldots, z_p\} \). Hence, sampling each of the components in this manner, either in a deterministic or in a random order, defines a Markov kernel on \( \mathbb{Z} \), with stationary distribution \( f(z) \). Since only the conditionals \( f_1, \ldots, f_p \) are used in the sampling procedure, all of the simulations can be univariate (or low-dimensional), even for high-dimensional problems, which is a key property of the Gibbs sampler. We summarize the method in Algorithm 3.

The specific parameterization that is used in the construction of a Gibbs sampler can have very significant effects on its performance, see e.g. Roberts and Sahu (1997); Hills and Smith (1992). If the individual components \( z_1, \ldots, z_p \) are strongly dependent, sampling from the full conditionals can lead to insignificant updates of the components, resulting in a poorly mixing Gibbs sampler. An example of this phenomena, which is of particular
Algorithm 3: Gibbs sampler (deterministic scan)

1: Set $z[0]$ arbitrarily.
2: for $r \geq 1$ do
3: Sample according to
   \[ z_1[r] \sim f_1(z_1 \mid z_2[r - 1], \ldots, z_p[r - 1]). \]
   \[ \vdots \]
   \[ z_i[r] \sim f_i(z_i \mid z_1[r], \ldots, z_{i-1}[r], z_{i+1}[r - 1], \ldots, z_p[r - 1]). \]
   \[ \vdots \]
   \[ z_p[r] \sim f_p(z_p \mid z_1[r], \ldots, z_{p-1}[r]). \]
4: end for

interest to the present work, is single-state Gibbs sampling for SSMs. Assume that the target distribution is the JSD $p(x_{1:T} \mid y_{1:T})$. A natural parameterization for a Gibbs sampler is to draw the individual states at different time points from their full conditional, i.e.

\[ x'_t \sim p(x_t \mid x_{1:t-1}, x_{t+1:T}, y_{1:T}), \quad (33) \]

for $t = 1, \ldots, T$. This leads to the single-state sampler which was used in the illustrating example in Section 1.3. As was experienced in this example, the resulting Gibbs sampler can suffer from very poor mixing. The reason is that there is often a strong dependence between consecutive states in an SSM. Hence, when $x'_t$ is sampled, the resulting value will be in close agreement with the conditioning on $x_{t-1}$ and $x_{t+1}$, resulting in small updates at each iteration.

Two important techniques, used to improve the mixing of the Gibbs sampler, are grouping and collapsing. To illustrate these concepts, assume that the variable of interest is partitioned into three parts, $z = \{z_1, z_2, z_3\}$. The basic Gibbs sampler would then, at each iteration, simulate according to

\[ z'_1 \sim f_1(z_1 \mid z_2, z_3), \quad z'_2 \sim f_2(z_2 \mid z'_1, z_3), \quad z'_3 \sim f_3(z_3 \mid z'_1, z'_2). \]

Grouping, or blocking, amounts to sample a group of variable, say $\{z_2, z_3\}$, jointly. That is, we simulate according to

\[ z'_1 \sim f_1(z_1 \mid z_2, z_3), \quad \{z'_2, z'_3\} \sim f_{2,3}(z_2, z_3 \mid z'_1). \]

This corresponds to a reparameterization of the Gibbs sampler, with only two components instead of three. If $z_2$ and $z_3$ are strongly dependent, sampling them jointly can result in much larger updates, improving the mixing of the basic Gibbs sampler.

Collapsing is another word for marginalization. In the simplest case, one component of the model, say $z_3$, is marginalized in all the steps of the sampler. That is, we simulate according to

\[ z'_1 \sim \tilde{f}_1(z_1 \mid z_2), \quad z'_2 \sim \tilde{f}_2(z_2 \mid z'_1), \]
where $\bar{f}_i$, $i = 1, 2$, are the full conditionals under the marginal distribution $\bar{f}(z_1, z_2) = \int f(z_1, z_2, z_3) \, dz_3$. This corresponds to a standard Gibbs sampler, targeting the marginal distribution $\bar{f}$ instead of $f$.

Perhaps more interestingly, the Gibbs sampler does not have to be fully collapsed, as above, to attain an improvement in mixing. In a partially collapsed Gibbs sampler, we marginalize a certain variable only in some of the Gibbs steps. For instance, we could simulate according to

$$z'_1 \sim f_1(z_1 \mid z_2, z_3), \quad z'_2 \sim \bar{f}_2(z_2 \mid z'_1), \quad z'_3 \sim f_3(z_3 \mid z'_1, z'_2).$$

This partially collapsed Gibbs sampler results in a valid MCMC method, with stationary distribution $f(z_1, z_2, z_3)$. Again, if $z_2$ and $z_3$ are strongly dependent, sampling $z_2$ unconditionally on $z_3$ enables a larger update, which in turn results in a larger update also of $z_3$.

However, care needs to be taken when constructing a partially collapsed sampler, since the order of the Gibbs steps in fact can affect the stationary distribution. If, for instance, we would collapse all the Gibbs steps to a maximum degree and sample each of the variables $z_1, z_2$ and $z_3$ from their respective marginals under $\bar{f}$, then we would loose all correlation between the variables. A more intricate example of an improperly collapsed sampler is,

$$z'_1 \sim \bar{f}_1(z_1 \mid z_2), \quad z'_2 \sim f_2(z_2 \mid z'_1, z_3), \quad z'_3 \sim f_3(z_3 \mid z'_1, z'_2).$$

This sampling scheme is seemingly similar to the one above, but this latter scheme does not leave $f(z_1, z_2, z_3)$ invariant. Again, the reason is that the sampler does not take the dependences between the variables into account in a correct way. To see why this is the case and how to construct a properly collapsed Gibbs sampler, see Dyk and Park (2008).

Grouping and collapsing will in general improve the mixing properties of the Gibbs sampler (see (Liu, 2001, Section 6.7)). In practice, the gain can be quite considerable. In the motivating example in Section 1.3, we pointed out that a backward simulator aims at sampling the state sequence $x_{1:T}$ jointly from the JSD, instead of resorting to single-state sampling of the individual state components. In light of the above discussion, we see that the backward simulator thus is a way of grouping the state variables to alleviate the strong dependencies between consecutive states.

A different technique which can also be useful is to reparameterize the model by making a change of variables. If the new variables are chosen in a way which decreases the posterior interdependence, then the mixing of the Gibbs sampler can be improved; see Papaspiliopoulos et al. (2003) and Section 4.6.3.

To be able to implement the Gibbs samplers described above, we need to be able to sample from all the involved conditionals under the target distribution $f$. While this is indeed possible for many interesting problems, it is not always the case. To alleviate this, one possibility is to use a mixed strategy, in which Gibbs steps are used whenever possible and MH steps are used for the intractable conditionals. Since the MH sampler leaves its target distribution invariant, each step of the mixed sampler will leave $f$ invariant, resulting in a

---

1In fact, in this specific example, the partially collapsed Gibbs sampler is equivalent to the grouped Gibbs sampler.
valid MCMC kernel. This approach is sometimes referred to as \textit{Hastings-within-Gibbs}.

3 Backward simulation for state-space models

SMC provides an approximation of the joint smoothing distribution, given by (25). However, as discussed in Section 2.1.2, path degeneracy makes this approximation unreliable for anything but filtering or fixed-lag smoothing with a short enough lag. In the present section, we will see how the joint smoothing problem can be addressed by complementing the forward filter with a second recursion, evolving in the time-reversed direction. In particular, we will make use of SMC to enable backward simulation for general SSMs, allowing us to generate samples approximately distributed according to the JSD.

A related approach is the two-filter smoother (Bresler, 1986), which is based on one filter moving forward in time and one filter moving backward in time. When the two filters meet “in the middle”, the information is merged, enabling computation of smoothed estimates. This approach will not be considered here, and for further reading on SMC implementations of the two-filter smoother we refer to Briers et al. (2010); Fearnhead et al. (2010).

3.1 Forward filter/backward simulator

As was recognized in Section 1.7, it is possible to make use of the backward recursion (12) to generate samples from the JSD. The key ingredient is the backward kernel. Let us assume that the model under study is fully dominated. The backward kernel density is then given by (11). As previously pointed out, this expression depends explicitly on the filtering density \( p(x_t \mid y_{1:t}) \). The basic idea, underlying the particle-based forward filter/backward simulator (FFBSi) (Godsill et al., 2004; Doucet et al., 2000b), is to make use of a PF to approximate the backward kernel.

Assume that we have recorded a sequence of observations \( y_{1:T} \) up to some final time point \( T \). Assume also that we have applied a PF to this batch of data. For each time \( t = 1, \ldots, T \) we thus have a weighted particle system \( \{x_i^t, w_i^t\}_{i=1}^N \) approximating the filtering distribution at time \( t \),

\[
\hat{p}^N(dx_t \mid y_{1:t}) \triangleq \sum_{i=1}^N w_i^t \delta_{x_i^t}(dx_t). \tag{34}
\]

These approximations are given by marginalization of (25), or, put differently, by discarding everything from the particle trajectories except for the last time point. Using (34) in the expression (11), we obtain an approximation of the backward kernel,

\[
\hat{B}^N_t(dx_t \mid x_{t+1}^i) \triangleq \sum_{i=1}^N \frac{w_i^t f(x_{t+1} \mid x_i^t)}{\sum_{i=1}^N w_i^t f(x_{t+1} \mid x_i^t)} \delta_{x_i^t}(dx_t). \tag{35}
\]

It is important to note that the particles \( \{x_i^t, w_i^t\}_{i=1}^N \) that are used to approximate the filtering distribution in (34), are those that were generated at time \( t \) in the PF. Hence, there is in general a rich diversity of particles in this system, and the approximations (34) and (35) do not suffer (directly) from path degeneracy. This is also the reason for why FFBSi
can succeed where the PF fails. Even though the PF results in a degenerate approximation of the JSD, it can generally provide accurate approximations of the filtering distributions, which is all that is needed to compute an approximation of the backward kernel.

We can now make use of (34) and (35) to generate a backward trajectory by sampling (cf. (16)),

\[ \tilde{x}_T \sim \tilde{p}^N(dx_T \mid y_{1:T}), \]  
\[ \tilde{x}_t \sim \tilde{B}^N_t(dx_t \mid \tilde{x}_{t+1}), \]

for \( t = T - 1, \ldots, 1 \). The backward trajectory \( \tilde{x}_{1:T} \) is an approximate realization from \( p(x_{1:T} \mid y_{1:T}) \). How close the distribution of \( \tilde{x}_{1:T} \) is to the JSD, clearly depends on how accurate the PF approximations (34) and (35) are. The convergence properties of FFBSi will be the topic of Section 3.2.2.

Sampling according to (36a) simply consists of drawing among the filter particles at time \( T \) with \( P(\tilde{x}_T = x_i^T) = w_i^T \). Similarly, the empirical backward kernel has finite support and it can be written as,

\[ \tilde{B}^N_t(dx_t \mid \tilde{x}_t+1) = \sum_{i=1}^{N} \tilde{w}_i^{t|T} \delta_{x_i^t}(dx_t), \]

where we have defined the smoothing weights

\[ \tilde{w}_i^{t|T} = \frac{w_i^t f(\tilde{x}_{t+1} \mid x_i^t)}{\sum_{l=1}^{N} w_l^t f(\tilde{x}_{t+1} \mid x_l^t)}. \]

Hence, to sample according to (36b), we evaluate the smoothing weights \( \{ \tilde{w}_i^{t|T} \}_{i=1}^{N} \) and draw among the forward filter particles at time \( t \) with \( P(\tilde{x}_t = x_i^t) = \tilde{w}_i^{t|T} \). Note that, in general, \( \tilde{x}_t \) will differ from the ancestor particle of \( \tilde{x}_{t+1} \), resulting in an increased particle diversity.

We emphasize that the expression (35) relies on the assumption that the model under study is fully dominated. Applying backward simulation to degenerate (non fully dominated) models, turns out to be much more tricky. We will return to this issue in Section 4.

When using FFBSi to address the smoothing problem, we typically generate multiple backward trajectories \( \{ \tilde{x}_{1:T}^j \}_{j=1}^{M} \), by repeating the backward simulation (36) \( M \) times. Hence, conditionally on the forward filter particles, the collection \( \{ \tilde{x}_{1:T}^j \}_{j=1}^{M} \) are i.i.d. samples. The conditional distribution of these samples will be given explicitly in Section 3.2.1. The backward trajectories define an unweighted point-mass approximation of the joint smoothing distribution,

\[ \tilde{p}^M(dx_{1:T} \mid y_{1:T}) \triangleq \frac{1}{M} \sum_{j=1}^{M} \delta_{\tilde{x}_{1:T}^j}(dx_{1:T}). \]

Clearly, this approximation can also be used to approximate any marginal or fixed-interval smoothing distribution.

We summarize the FFBSi in Algorithm 4, and illustrate the backward simulation proce-
**Algorithm 4** FFBSi (Godsill et al., 2004)

**Input:** Forward filter particle systems \( \{x^i_t, w^i_t\}_{i=1}^N \) for \( t = 1, \ldots, T \).

**Output:** Backward trajectories \( \{\tilde{x}^j_{1:T}\}_{j=1}^M \).

1. Sample independently \( \{b_T(j)\}_{j=1}^M \sim \text{Cat} \left( \{w^i_T\}_{i=1}^N \right) \).
2. Set \( \tilde{x}^j_T = x^b_T(j) \) for \( j = 1, \ldots, M \).
3. for \( t = T-1 \) to 1 do
   4. for \( j = 1 \) to \( M \) do
      5. Compute \( \tilde{w}^{t,j}_{i|T} \propto w^i_t f(\tilde{x}^j_{t+1} | x^i_t) \) for \( i = 1, \ldots, N \).
      6. Normalize the smoothing weights \( \{\tilde{w}^{t,j}_{i|T}\}_{i=1}^N \) to sum to one.
      7. Draw \( b_t(j) \sim \text{Cat} \left( \{\tilde{w}^{t,j}_{i|T}\}_{i=1}^N \right) \).
      8. Set \( \tilde{x}^j_t = x^b_t(j) \) and \( \tilde{x}^j_{t:T} = \{\tilde{x}^j_t, \tilde{x}^j_{t+1:T}\} \).
   9. end for
10. end for

---

**Figure 5:** Particle trajectories for \( N = 4 \) particles over \( T = 5 \) time steps after a completed forward filtering pass. The sizes of the dots represent the particle weights.

---

**Example 6: Backward simulation**

We illustrate the backward simulation process on a toy example. In Figure 5, we show the particle trajectories generated by a forward PF in a one-dimensional problem. The dots show the particle positions for the \( N = 4 \) particles over \( T = 5 \) time steps and their sizes represent the particle weights. The dots are connected, to illustrate the ancestral dependence of the particles. All particles at time \( t = 5 \) share a common ancestor at time \( t = 3 \), i.e. the particle paths are degenerate.

In Figure 6 we illustrate the simulation of one backward trajectory. In the upper left plot, the backward trajectory is initialized by sampling from the forward filter particles at time \( t = 5 \). The probability of sampling a particle \( x^i_T \) is given by its importance weight \( w^i_T \). The initialized backward trajectory is shown as a square. The particle weights at \( t = 4 \) are thereafter recomputed according to (38). The smoothing weights \( \{\tilde{w}^{t,j}_{i|T}\}_{i=1}^N \) are shown as circles, whereas the filter weights are illustrated with dots. Another particle is then
drawn and appended to the backward trajectory. In the upper right and lower left plots, the trajectory is augmented with new particles at $t = 3$ and $t = 2$, respectively. Finally, in the lower right plot, a final particle is appended at $t = 1$, forming a complete backward trajectory $\tilde{x}_{1:5}$. Observe that the generated backward trajectory differs from the ancestral line of the forward filter particle as shown in Figure 5. The procedure can be repeated as many times as needed (using the same forward filter particles), to generate a collection of backward trajectories.

The backward trajectories are conditionally i.i.d. and the particle diversity among the trajectories $\{\tilde{x}_{1:T}^j\}_{j=1}^M$ will thus in general be larger than that among the forward filter trajectories $\{x_{1:T}^i\}_{i=1}^N$, since the latter suffer from path degeneracy. However, an interesting question to ask is the following. Assume that we generate only a single backward trajectory $\tilde{x}_{1:T}$. Assume also that we extract a single trajectory $x_{1:T}^k$ from the PF, by sampling once from the empirical distribution (25) at time $T$. Then, will the distribution of $\tilde{x}_{1:T}$ be closer to the true JSD, than the distribution of $x_{1:T}^k$?

The answer to this question is not obvious in the general case. Intuitively, we might think that the answer is yes, since $\tilde{x}_{1:T}$ is sampled by a forward/backward smoothing procedure, whereas $x_{1:T}^k$ is sampled by a forward-only procedure. However, at least when the underlying SMC sampler is a bootstrap PF, this turns out not to be the case, as is stated in the following proposition due to Olsson and Rydén (2011).

**Proposition 1.** Assume that the weighted particle system $\{x_{1:T}^i, w_{1:T}^i\}_{i=1}^N$ is generated by a bootstrap particle filter. Let $k$ be sampled with $\mathbb{P}(k = i) = w_{1:T}^i$, and let $\tilde{x}_{1:T}$ be generated by a backward simulator. Then, $\mathbb{P}(x_{1:T}^k \in A) = \mathbb{P}(\tilde{x}_{1:T} \in A)$ for any set $A$.

**Proof:** See (Olsson and Rydén, 2011, Proposition 5).
Hence, the backward simulation procedure should be thought of as a way to reduce the correlation between the degenerate forward filter trajectories, rather than as a way to actually improve the quality of the individual trajectories.

As can be seen in Algorithm 4 (page 72), to generate $M$ backward trajectories, the smoothing weights need to be computed for index $i$ ranging from 1 to $N$ and index $j$ ranging from 1 to $M$. Hence, the computational complexity of the algorithm is $O(NM)$. The number of backward trajectories $M$, generated by Algorithm 4 is arbitrary. However, to obtain accurate approximations of the smoothing distributions of interest, it is clear that $M$ needs to be fairly large. The computational complexity of Algorithm 4 may therefore be prohibitive. One option is to make use of parallel architectures, e.g. graphics processing units. Algorithm 4, and in fact most backward simulators, is particularly well suited for parallel implementation, since the backward trajectories can be generated independently. In Section 3.3, we will discuss alternative methods for reducing the computational complexity of the algorithm.

The FFBSi is illustrated in the example below, which also provides some additional insight into how to reason about the design parameters $N$ and $M$.

---

**Example 7: FFBSi**

We consider a standard nonlinear time-series model previously used by, among others, Andrade Netto et al. (1979); Gordon et al. (1993); Godsill et al. (2004). The model is given by,

\[ x_{t+1} = 0.5 x_t + 25 \frac{x_t}{1+x_t^2} + 8 \cos(1.2t) + v_t, \]

\[ y_t = 0.05 x_t^2 + e_t, \]

with $v_t \sim \mathcal{N}(0, \sigma_v^2)$, $e_t \sim \mathcal{N}(0, \sigma_e^2)$ and $x_1 \sim \mathcal{N}(0, 5)$. We take $\sigma_v^2 = 10$ and $\sigma_e^2 = 1$ and generate $T = 100$ observations $y_{1:T}$ from the system. The smoothing distribution for this model is distinctively bimodal when $\sigma_e < \sigma_v$, rendering state inference problematic.

To address the joint smoothing problem, we apply the FFBSi of Algorithm 4 to generate $M$ backward trajectories. A bootstrap PF with $N$ particles is used in the forward direction. To see how the performance is affected by different numbers of particles and backward trajectories, we run the FFBSi several times on the same data, with different values for $N$ and $M$.

To evaluate the performance, we compute the posterior mean estimates

\[ \tilde{x}_{N,M}^{N,M} = \frac{1}{M} \sum_{j=1}^{M} \tilde{x}_t^j \]

for each smoother, for $t = 1, \ldots, T$. We then compute the root-mean-square error
Figure 7: Averaged RMSE values (log-scale), for different combinations of $N$ and $M$.

$\varepsilon(N, M) = \sqrt{\frac{1}{T} \sum_{t=1}^{T} \left(\bar{x}_{t,T}^{N,M} - \mathbb{E}[x_t | y_{1:T}]\right)^2}$.

Since the RMSE is defined relative to the true posterior mean, we expect $\varepsilon(N, M) \to 0$ as $N \to \infty$ and $M \to \infty$ (see Section 3.2.2). We run the smoothers on a grid of values of $N$ and $M$, with $N$ ranging from 10 to 3000 and with $M$ ranging from 10 to 1000. Furthermore, we use 500 independent runs at each grid point and average the RMSE values, to reduce the effects of randomness. The results are reported in Figure 7.

As expected, the error is decreasing with both $N$ and $M$. It is interesting to note, however, that the dependence on $N$ is much more pronounced than the dependence on $M$. Even for a very modest number of backward trajectories ($M = 10$), the error can be reduced a fair bit by just increasing $N$. This is not the case if we instead fix $N = 10$ and only increase $M$.

These results can be explained by the fact that the FFBSi relies heavily on the empirical backward kernel (35). Since this only depends on the forward filter, it is clear that we need to take $N$ large to obtain an accurate approximation of the kernel. Conditionally on the forward filter particles, Algorithm 4 generates $M$ i.i.d. backward trajectories. If the distribution of these trajectories, which only depends on $N$, is “close enough” to the true JSD, a modest number of samples can be sufficient to obtain accurate posterior estimates.

The results in Figure 7 suggests that we can try to find an optimal trade-off between $M$

\footnote{To obtain a ground truth estimate, we run an FFBSi with 100,000 forward filter particles and 50,000 backward trajectories, yielding very accurate results.}
and $N$ for a given computational time. However, the results can vary between different models. Also, in this example, we have only considered estimating the posterior means $\mathbb{E}[x_t \mid y_{1:T}]$. For other test functions, the results may also differ. Furthermore, for the FFBSi of Algorithm 4, the computational cost scales rather predictively as $MN$. However, as we will see in Section 3.3, there are alternative implementations of the FFBSi with smaller computational costs, which on the other hand are harder to predict. Hence, it can be difficult to find an optimal trade-off, even if a graph such as in Figure 7 would be available. However, as a rule-of-thumb, it is better to put most effort on the forward filter to obtain accurate backward kernel approximations, i.e. to take $N > M$.

3.2 Analysis and convergence

In this section we discuss the relationship between the FFBSi and a method referred to as forward filter/backward smoother (FFBSm). As we will see, analyzing the FFBSm is useful in order to assess convergence properties for FFBSi and to better understand the properties of the method.

3.2.1 Forward filter/backward smoother

So far, we have seen how the empirical backward kernel (35) can be used to simulate backward trajectories, approximately distributed according to the JSD. From these trajectories, we obtained an approximation of the joint smoothing distribution given by (39). However, an alternative way to address the smoothing problem is to simply plug the approximation (35) into the backward recursion (12). Contrary to the backward simulator, this will result in a deterministic approximation of the smoothing distribution, conditionally on the forward filter particles. This method is referred to as the forward filter/backward smoother (FFBSm).

FFBSm was introduced for marginal smoothing by Doucet et al. (2000a) and thus predates the FFBSi. The computational complexity for FFBSm is at least quadratic in the number of particles. For FFBSi, it is possible to obtain a lower computational cost (see Section 3.3), making it the preferred method of choice (among these two) for most models. However, the two methods are closely related and, as noted above, by studying the FFBSm we can gain additional insight into the properties of FFBSi.

We start by reviewing the marginal FFBSm by Doucet et al. (2000a). This method aims at computing the marginal smoothing densities $p(x_t \mid y_{1:T})$ for $t = 1, \ldots, T$. These densities are given recursively, backward in time, by

$$p(x_t \mid y_{1:T}) = \int p(x_t \mid x_{t+1}, y_{1:t})p(x_{t+1} \mid y_{1:T}) \, dx_{t+1}. \tag{40}$$

Assume that we have obtained a weighted particle system $\{x_{t+1}^i, \omega_{t+1|T}^i\}_{i=1}^N$, targeting $p(x_{t+1} \mid y_{1:T})$. At the final time point, this is given by the PF since the filtering distribution and the marginal smoothing distribution coincide at time $T$, i.e. $\omega_{t|T}^i = w_T^i$ for $i = 1, \ldots, N$. Plugging the empirical distribution defined by this particle system, and
the approximation of the backward kernel (35), into (40) results in

\[ \tilde{p}^{N}_{\text{FFBSm}}(dx_t \mid y_1:T) \triangleq \sum_{j=1}^{N} \omega_{t+1|T}^j \sum_{i=1}^{N} w_t^i f(x_{t+1}^j \mid x_t^i) \delta_{x_t^i}(dx_t) \]

\[ = \sum_{i=1}^{N} \omega_{t|T}^i \delta_{x_t^i}(dx_t), \]  

(41)

where we have defined the smoothing weights,

\[ \omega_{t|T}^i \triangleq w_t^i \sum_{j=1}^{N} \omega_{t+1|T}^j f(x_{t+1}^j \mid x_t^i) \sum_{l=1}^{N} w_l^i f(x_{t+1}^l \mid x_t^i), \]  

(42)

for \( i = 1, \ldots, N \). The smoothing weights defined above are self-normalized, in the sense that they sum to one by construction.

We have added 'FFBSm' explicitly in the notation to emphasize that the approximation of the marginal smoothing distribution given by (41) is not the same as we would get by marginalization of the PF approximation (25). Note, however, that the FFBSm reuses the forward filter particles \( \{x_i^t\}_{i=1}^{N} \), but updates the importance weights of these particles to target the marginal smoothing distribution, rather than the filtering distribution.

The computational complexity of the marginal FFBSm is \( O(N^2) \), which can be seen from the weight expression (42). It is possible to apply the same approach to approximate any fixed interval distribution or the joint smoothing distribution, albeit at an increased computational cost. For the joint smoothing distribution, we make use of the following factorization of the JSD,

\[ p(x_1:T \mid y_1:T) = \left( \prod_{t=1}^{T-1} p(x_t \mid x_{t+1}, y_{1:t}) \right) p(x_T \mid y_1:T), \]  

(43)

following from (12). By plugging the empirical filter and backward kernel approximations into this expression, we get

\[ \tilde{p}^{N}_{\text{FFBSm}}(dx_1:T \mid y_1:T) \]

\[ \triangleq \sum_{i_1=1}^{N} \cdots \sum_{i_T=1}^{N} \left( \prod_{t=1}^{T-1} \frac{w_t^{i_t} f(x_{t+1}^{i_t+1} \mid x_t^{i_t})}{\sum_{i=1}^{N} w_t^i f(x_{t+1}^i \mid x_t^i)} \right) w_T^{i_T} \delta_{x_1^{i_1} \cdots x_T^{i_T}}(dx_1:T). \]  

(44)

The expression above defines a point-mass distribution on \( X^T \), and the cardinality of its support is \( N^T \). The meaning of the distribution can be understood in the following way. For each time \( t = 1, \ldots, T \), the particles \( \{x_t^i\}_{i=1}^{N} \) generated by the PF is a set in \( X \) of cardinality \( N \). By picking one particle from each time point, we obtain a particle trajectory, i.e. a point in \( X^T \),

\[ (x_1^{i_1}, \ldots, x_T^{i_T}) \in X^T. \]  

(45)

By letting all of the indices \( i_1, i_2, \ldots, i_T \) range from 1 to \( N \), we get in total \( N^T \) such trajectories. The empirical distribution (44) assigns, to each such trajectory, an importance
weight \( \omega_{1:T|T}(i_1, \ldots, i_T) \).

Clearly, the approximation (44) is impractical for any real problem of interest, since the computation of the weights is an \( O(N^T) \) operation, both in terms of computational complexity and storage. However, the expression (44) provides an interesting connection between FFBSm and FFBSi.

From the construction of the backward simulator (36), it follows that FFBSi generates i.i.d. draws from the distribution (44), conditionally on the forward filter particles. We thus make the following observation. Let \( \varphi : X_T \rightarrow \mathbb{R} \) be some test function, of which we seek to compute the expectation under the JSD, \( \mathbb{E}[\varphi(x_{1:T}) \mid y_{1:T}] \). By (44), the FFBSm estimator is given by,

\[
\hat{\varphi}^N_{\text{FFBSm}} = \sum_{i_1=1}^{N} \cdots \sum_{i_T=1}^{N} \omega_{1:T|T}(i_1, \ldots, i_T) \varphi(x_{i_1}^{i_T}, \ldots, x_T^{i_T}).
\] (46)

This estimator is, unfortunately, also intractable. However, the FFBSi approximation (39) provides an unbiased estimator of (46), i.e.

\[
\tilde{\varphi}^N_{\text{FFBSi}} = \mathbb{E}\left[ \varphi^M_{\text{FFBSi}} \mid \{x_t^i, w_t^i\}_{i=1}^{N}, t = 1, \ldots, T \right],
\] (47)

where \( \varphi^M_{\text{FFBSi}} = \frac{1}{M} \sum_{j=1}^{M} \varphi(\tilde{x}_{1:T}^j) \). Here, the expectation is taken w.r.t. the random components of the backward simulation, conditionally on all the random variables generated by the forward filter. That is, the expectation in (47) is given by a summation over the backward trajectory indices, corresponding exactly to the expression (46).

From (47), we see that the FFBSm estimator can be seen as a Rao-Blackwellization of the FFBSi estimator (see e.g. Lehmann and Casella (1998)). Put the other way around, FFBSi is an “anti-Rao-Blackwellization” of FFBSm. Rao-Blackwellization usually aims at reducing the variance of an estimator, but generally at the cost of increased computational complexity. Here, we go the other way, and (significantly) reduce the complexity of the FFBSm estimator by instead employing FFBSi, albeit at the cost of a slight increase in variance.

### 3.2.2 Convergence of FFBSm and FFBSi

The close connection between FFBSm and FFBSi is useful in order to transfer convergence results from the former method to the latter. In this section, a few key results from Douc et al. (2011) are reviewed. For simplicity, we assume \( M = N \) throughout this section. Let \( \|\varphi\|_{\infty} = \sup_x |\varphi(x)| \) and \( \text{osc}(\varphi) = \sup_{(x,x')} |\varphi(x) - \varphi(x')| \) be the supremum and oscillator norms, respectively. Let \( \xrightarrow{D} \) denote convergence in distribution. We make the following assumption on bounds on the likelihood and the weight function.

\textbf{(A1)} For any \( 1 \leq t \leq T \),

- \( g(y_t \mid \cdot) > 0 \) and \( \|g(y_t \mid \cdot)\|_{\infty} < \infty \).
- \( \|W_t(\cdot, \cdot; y_t)\|_{\infty} < \infty \) where \( W_t(x_t, x_{t-1}; y_t) \) is defined in (23).

We start with a deviation inequality for FFBSm.
**Theorem 2 (Exponential deviation inequality for FFBSm).** Assume (A1). Then there exists constants $c_{1,T}$ and $c_{2,T} > 0$ such that for all $N, \varepsilon > 0$ and all bounded measurable functions $\varphi : X^T \to \mathbb{R}$,

$$P(\left| \hat{\varphi}^{N}_{FFBSm} - \mathbb{E}[\varphi(x_{1:T}) \mid y_{1:T}] \right| \geq \varepsilon) \leq c_{1,T} \exp(-c_{2,T} N^2 \text{osc}^2(\varphi)).$$

**Proof:** See (Douc et al., 2011, Theorem 5).

The exponential deviation inequality of Theorem 2 can be more or less directly extended to FFBSi by using the identity (47).

**Corollary 1 (Exponential deviation inequality for FFBSi).** Assume (A1). Then there exists constants $c_{1,T}$ and $c_{2,T} > 0$ such that for all $N, \varepsilon > 0$ and all bounded measurable functions $\varphi : X^T \to \mathbb{R}$,

$$P(\left| \tilde{\varphi}^{N}_{FFBSi} - \mathbb{E}[\varphi(x_{1:T}) \mid y_{1:T}] \right| \geq \varepsilon) \leq c_{1,T} \exp(-c_{2,T} N^2 \text{osc}^2(\varphi)).$$

**Proof:** See (Douc et al., 2011, Corollary 6).

Hence, when using FFBSi to compute an expectation, the probability that the Monte Carlo error exceeds some value $\varepsilon$ is bounded by a function decreasing exponentially fast with the number of particles. Among other things, this non-asymptotic result implies almost sure convergence of the estimator $\tilde{\varphi}^{N}_{FFBSi}$.

In the asymptotic regime, it is also possible to establish a central limit theorem (CLT) with rate $\sqrt{N}$. To do so, we make an additional assumption on the transition density function and the proposal density.

(A2) $\|f(\cdot \mid \cdot)\|_{\infty} < \infty$ and, for any $1 \leq t \leq T$, $\|r_t(\cdot \mid \cdot, y_t)\|_{\infty} < \infty$.

A CLT can now be stated for FFBSm.

**Theorem 3 (CLT for FFBSm).** Assume (A1) and (A2). Then, for any bounded measurable function $\varphi : X^T \to \mathbb{R}$,

$$\sqrt{N}(\hat{\varphi}^{N}_{FFBSm} - \mathbb{E}[\varphi(x_{1:T}) \mid y_{1:T}]) \overset{D}{\to} \mathcal{N}(0, \Gamma_{1:T|T}[\varphi]),$$

where $\Gamma_{1:T|T}[\varphi]$ is defined in (Douc et al., 2011, Equation (48)).

**Proof:** See (Douc et al., 2011, Theorem 8).

Similarly to above, we can extend the CLT to the FFBSi by using the relation (47).

**Corollary 2 (CLT for FFBSi).** Assume (A1) and (A2). Then, for any bounded measurable function $\varphi : X^T \to \mathbb{R}$,

$$\sqrt{N}(\tilde{\varphi}^{N}_{FFBSi} - \mathbb{E}[\varphi(x_{1:T}) \mid y_{1:T}]) \overset{D}{\to} \mathcal{N}(0, \text{Var}(\varphi(x_{1:T}) \mid y_{1:T}) + \Gamma_{1:T|T}[\varphi]),$$

where $\Gamma_{1:T|T}[\varphi]$ is defined in (Douc et al., 2011, Equation (48)).
**Proof:** See (Douc et al., 2011, Corollary 9).

The asymptotic variance for the FFBSi estimator contains an additional term, compared to the FFBSm estimator. This is in agreement with what we can expect from (47). However, as pointed out above, the increase in variance is compensated for by a significant reduction in computational complexity.

Additional convergence results are derived in (Douc et al., 2011), e.g. time uniform bounds for marginal smoothing estimators. Non-asymptotic deviation inequalities for FFBSi estimators of smoothed additive functionals are given in (Dubarry and Le Corff, 2012).

### 3.3 Backward simulation with rejection sampling

The particle smoothers considered above have quadratic computational complexity, which can be prohibitive for many practical applications. FFBSi requires $O(MN)$ operations and the computational cost of FFBSm is (at least) $O(N^2)$.

There exist several different approaches to reduce the computational complexity of various particle smoothers, based on both numerical approximations and algorithmic modifications. Klaas et al. (2006) use so called $N$-body methods to derive a marginal FFBSm with $O(N \log N)$ complexity. The same approach can be used also for FFBSi. These methods impose additional approximations, though the tolerance can usually be specified beforehand. Similar $N$-body methods have also been used by Klaas et al. (2005); Lang and de Freitas (2005) to compute exact joint maximum a posteriori estimates of the state trajectory in $O(N \log N)$ computational complexity.

There have also been developments based on the two-filter algorithm (Briers et al., 2010). In its original formulation, this method is quadratic in the number of particles. However, Fearnhead et al. (2010) have proposed a modified two-filter algorithm with linear complexity. In (Fearnhead, 2005), quasi-random numbers are used for particle smoothing resulting in a method, albeit with quadratic complexity, but at the same time with a quadratic decrease in variance (at least for one-dimensional systems).

In this section, we will see how the computational complexity of FFBSi can be reduced, without introducing any further approximations, by using rejection sampling (RS).

#### 3.3.1 Rejection sampling

The basic idea that we will explore is to make use of a rejection sampler within the FFBSi algorithm, as suggested by Douc et al. (2011). The key insight is that we do not need to evaluate all the smoothing weights $\{\tilde{w}_i | T\}_{i=1}^N$ to be able to sample from the empirical backward kernel (37). To convince ourselves that there indeed is room for improvement, note that in the FFBSi in Algorithm 4 on page 72 we evaluate $N$ smoothing weights on line 5, draw a single sample from the categorical distribution on line 7 and then discard all the weights. Instead of making an exhaustive evaluation of the categorical distribution, we can sample from it using RS. For this to be applicable, we assume that the transition density function is bounded from above,

$$f(x_{t+1} | x_t) \leq \rho, \quad (x_t, x_{t+1}) \in X^2,$$

(48)
At time $t$, which is true for many models arising in practical applications.

Algorithm 5 Rejection sampling FFBSi (Douc et al., 2011)

**Input:** Forward filter particle systems $\{x_t^i, w_t^i\}_{i=1}^N$ for $t = 1, \ldots, T$.

**Output:** Backward trajectories $\{\tilde{x}_{1:T}^j\}_{j=1}^M$.

1. Sample independently $\{b_T(j)\}_{j=1}^M \sim \text{Cat} \left(\{w_T^i\}_{i=1}^N\right)$.
2. Set $\tilde{x}_{T}^j = x_{T}^{b_T(j)}$ for $j = 1, \ldots, M$.
3. for $t = T - 1$ to 1 do
   4. $L \leftarrow \{1, \ldots, M\}$.
   5. while $L$ is not empty do
      6. $n \leftarrow \text{card}(L)$.
      7. $\delta \leftarrow \emptyset$.
      8. Sample independently $\{I(k)\}_{k=1}^n \sim \text{Cat}\left(\{w_{t}^i\}_{i=1}^N\right)$.
      9. Sample independently $\{U(k)\}_{k=1}^n \sim U([0, 1])$.
   10. for $k = 1$ to $n$ do
        11. if $U(k) \leq f(\tilde{x}_{t+1}^{I(k)} | x_{t}^{I(k)}) / \rho$ then
            12. $b_t(L(k)) \leftarrow I(k)$.
            13. $\delta \leftarrow \delta \cup \{L(k)\}$.
        end if
   14. end for
   15. $L \leftarrow L \setminus \delta$.
   16. end while
    17. Set $\tilde{x}_t^j = x_t^{b_t(j)}$ and $\tilde{x}_{1:T}^j = \{\tilde{x}_t^j, \tilde{x}_{t+1:T}^j\}$ for $j = 1, \ldots, M$.
18. end for

which is true for many models arising in practical applications.

At time $t$, we wish to sample an index $b_t(j)$, corresponding to the forward filter particle which is to be appended to the $j$th backward trajectory. The target distribution is categorical over the index space $\{1, \ldots, N\}$, with probabilities $\{w_{t}^i\}_{i=1}^N$ (which we have not computed yet). As proposal, we take another categorical distribution over the same index space, with (known) probabilities $\{w_{t}^i\}_{i=1}^N$. That is, we propose samples based on the filter weights, rather than on the smoothing weights.

Assume that a sample index $I(j)$ is proposed for the $j$th backward trajectory. To compute the acceptance probability, we consider the ratio between the target and the proposal distributions. Using the definition of the smoothing weights (38) we get,

$$
\frac{\tilde{w}_{t|T}^{I(j), j}}{w_t^{I(j)}} = \frac{f(\tilde{x}_{t+1}^j | x_{t}^{I(j)})}{\sum_{i=1}^N w_t^i f(\tilde{x}_{t+1}^i | x_{t}^{I(j)})} \propto f(\tilde{x}_{t+1}^j | x_{t}^{I(j)}) \leq \rho. \tag{49}
$$

This implies that the sample should be accepted with probability $f(\tilde{x}_{t+1}^j | x_{t}^{I(j)}) / \rho$ (see e.g. (Bishop, 2006, Section 11.1.2)). The rejection-sampling-based FFBSi (RS-FFBSi) is given in Algorithm 5. We also provide MATLAB code in Listing A.1.

The rationale for using RS is that in the limit $N \to \infty$, the computational cost will be linear in the number of particles. This is formalized in the following proposition, due to Douc et al. (2011).
INPUT:

x_pf — nx * N * T array with forward filter particles.
w_pf — I * N * T array with forward filter weights.

OUTPUT:

x_ffbsi — nx * M * T array with backward trajectories.

Listing A.1: MATLAB code for RS-FFBSi. We have assumed that a function transition_density_function(x_t1, x_t) is available, where x_t1 and x_t are nx × N matrices (nx being the state dimension). The function computes the transition density function value f(x_{t+1} | x_t) for each pair of columns in the two matrices, and returns the result as a 1 × N row vector.
Proposition 2. Assume (48). Let $M = N$ and let $C_t^N$ be the total number of simulations required in the accept-reject procedure at time $t$ in Algorithm 5. Assume that $\int g(y_t \mid x_t) \, dx_t < \infty$ for all $t = 1, \ldots, T$. Then, for a bootstrap PF or a fully adapted PF, $C_t^N / N$ converges in probability to a finite constant.

Proof: See ( Douc et al., 2011, Proposition 1 ).

Proposition 2 implies that for large $N$, RS-FFBSi will have close to linear computational complexity. However, it is worth to note that there is no upper bound on the number of times that the while-loop at line 5 may be executed. It has been observed in practice that different backward trajectories can get very different acceptance probabilities. This has the effect that most of the time required by Algorithm 5 is spent on just a few trajectories. Furthermore, RS-FFBSi has been found to be sensitive to the dimension of the state-space $X$ (Taghavi et al., 2013; Bunch and Godsill, 2012). Technically, the rejection sampling in Algorithm 5 is done over the finite space $\{1, \ldots, N\}$. However, the underlying idea is to sample from the backward kernel using rejection sampling, with the filtering distribution as a proposal. That is, the rejection sampling is in effect done in the space $X$. Hence, it is natural to expect that the acceptance probability is diminished as the dimension of $X$ increases, due to the curse of dimensionality.

Due to the effects mentioned above, RS-FFBSi can in fact require more computational time than the standard FFBSi (Algorithm 4) for many models. To alleviate this, Taghavi et al. (2013) have proposed a hybrid strategy which switches between RS-FFBSi and FFBSi. This approach will be reviewed in the next section.

Finally, as a practical detail, we note that the sampling at line 8 should be conducted prior to the for-loop at line 10, for Algorithm 5 to reach linear complexity. That is, when proposing indices $\{I(k)\}_{k=1}^N$ from the categorical distribution with probabilities $\{w^t_i\}_{i=1}^N$, we draw the samples all at once. This allows us to use the efficient multinomial sampler by ( Douc et al., 2011, Algorithm 2 ), which generates $N$ i.i.d. samples from a categorical distribution with support at $N$ points in $O(N)$ time.

### 3.3.2 Early stopping

Since the expression (48) depends on $x_{t+1}$, different backward trajectories will get different acceptance probabilities in RS-FFBSi. The trajectories with high probabilities are typically accepted early in the process, whereas the trajectories with low probabilities can remain for many iterations. This has the effect that the cardinality of the set $L$ (see Algorithm 5, page 81) decreases fast in the beginning, but it can linger for a long time close to zero. Consequently, most of the time required by RS-FFBSi is spent on just a few trajectories.

To speed up the algorithm, a hybrid strategy has been proposed by Taghavi et al. (2013). The idea is to run the rejection-sampling-loop for a certain number of iterations, and then switch to standard FFBSi for the remaining backward trajectories (i.e. by making an exhaustive evaluation of the remaining weights). The general method based on this idea, RS-FFBSi with early stopping, is given in Algorithm 6.
Algorithm 6 RS-FFBSi with early stopping (Taghavi et al., 2013)

**Input:** Forward filter particle systems \( \{x_t^i, w_t^i\}_{i=1}^N \) for \( t = 1, \ldots, T \).

**Output:** Backward trajectories \( \{\tilde{x}_{1:T}^j\}_{j=1}^M \).

1: Sample independently \( \{b_T(j)\}_{j=1}^M \sim \text{Cat} \left( \{w_T^i\}_{i=1}^N \right) \).
2: Set \( \tilde{x}_{T}^j = x_{T}^{b_T(j)} \) for \( j = 1, \ldots, M \).
3: for \( t = T - 1 \) to 1 do
4:   \( L \leftarrow \{1, \ldots, M\} \).
5:   while Stopping criterion is not met and \( L \) is not empty do
6:     Run one RS iteration (lines 6 to 16 in Algorithm 5, page 81).
7:   end while
8:   if \( L \) is not empty then
9:     Sample \( b_t(j) \) for \( j \in L \) using FFBSi (Algorithm 4, page 72).
10: end if
11: Set \( \tilde{x}_{t}^j = x_t^{b_t(j)} \) and \( \tilde{x}_{t:T}^j = \{\tilde{x}_{t}^j, \tilde{x}_{t+1:T}^j\} \) for \( j = 1, \ldots, M \).
12: end for

Two stopping criteria are proposed by Taghavi et al. (2013), one deterministic and one adaptive. The first is a simple stopping rule, in which a maximum number of \( K \) rejection-sampling-iterations are allowed. If \( L \) is not empty after \( K \) iterations, an exhaustive evaluation of the backward sampling weights is made for the remaining trajectories. The second stopping criterion is an adaptive rule, which monitors the number of acceptances at each rejection-sampling-iteration. Based on this information, the average acceptance probability is estimated and used to make a decision about when to stop.

The benefit of using the adaptive rule is that it avoids a hard (and possibly difficult) choice for the design parameter \( K \). Instead, it has the ability to automatically adapt the stopping time to the properties of the model, depending on the acceptance probabilities. As we will see in Example 8 below, early stopping can be quite useful in reducing the computational cost compared to both FFBSi and RS-FFBSi.

--- Example 8: RS-FFBSi with early stopping ---

This example is taken from Taghavi et al. (2013). We consider a second-order LGSS model, previously used to evaluate the two-filter smoother presented by Fearnhead et al. (2010),

\[
x_{t+1} = Ax_t + v_t, \quad v_t \sim \mathcal{N}(0, Q),
\]

\[
y_t = Cx_t + e_t, \quad e_t \sim \mathcal{N}(0, \sigma^2),
\]

with

\[
C = \begin{pmatrix} 1 & 0 \\ \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ \end{pmatrix}, \quad Q = \begin{pmatrix} \frac{1}{3} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{pmatrix}.
\]

We let \( \sigma \in \{0.1, 1, 10\} \). The reason to choose different values for \( \sigma \) is to obtain different acceptance probabilities in the rejection sampler, where \( \sigma = 0.1 \), correspond to a fairly high acceptance probability and \( \sigma = 10 \) to a very low acceptance probability. We choose \( N = 5000 \) forward filter particles and \( M = 1000 \) backward trajectories. For the deter-
ministic stopping rule, we consider three different thresholds: $K \in \{ \frac{M}{5}, \frac{M}{10}, \frac{M}{20} \}$. We also use the adaptive stopping rule from (Taghavi et al., 2013).

We emphasize that all the backward simulators considered here are equivalent in terms of accuracy. Hence, they are evaluated only in terms of computational time. We clock the CPU times using the `tic` and `toc` commands in Matlab. All the simulations are done on a standard laptop Intel(R) Core(TM) i7-3720Qm 2.60GHz platform with 8GB of RAM. For each value of $\sigma$, we generate five unique datasets, each consisting of $T = 100$ samples. For each dataset, we run the algorithms ten times and average the results, to reduce the effects of randomness.

The results for different choices of $\sigma$ are shown in Table 2. For high acceptance probabilities ($\sigma = 0.1$), there is a gain in using RS-FFBSi instead of FFBSi. However, as $\sigma$ is increased, some backward trajectories get very small acceptance probabilities with many rejections as a result. By using early stopping, a large improvement in computation time is attained for both small and large acceptance probabilities, compared to both FFBSi and RS-FFBSi.

### Table 2: Average CPU times in seconds for RS-FFBSi with early stopping.

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>FFBSi</th>
<th>RS-FFBSi</th>
<th>$K = \frac{M}{5}$</th>
<th>$K = \frac{M}{10}$</th>
<th>$K = \frac{M}{20}$</th>
<th>Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>44.65</td>
<td>19.50</td>
<td>2.03</td>
<td>1.94</td>
<td>2.36</td>
<td>1.92</td>
</tr>
<tr>
<td>1</td>
<td>45.28</td>
<td>77.71</td>
<td>3.49</td>
<td>3.65</td>
<td>5.33</td>
<td>3.79</td>
</tr>
<tr>
<td>10</td>
<td>48.63</td>
<td>355.70</td>
<td>14.83</td>
<td>20.21</td>
<td>25.86</td>
<td>15.25</td>
</tr>
</tbody>
</table>

#### 3.4 Backward simulation with MCMC moves

In this section, we review a few related methods in which MCMC moves are used in the context of backward simulation.

##### 3.4.1 Metropolis-Hastings FFBSi

The bottleneck of the FFBSi algorithm is the computation of the backward sampling weights (38). In RS-FFBSi, rejection sampling is used to avoid an exhaustive evaluation of the weights, which is possible since the weights can be evaluated up to proportionality in constant time. However, under the same conditions, it is also possible to apply MCMC. An alternative is thus to run a Metropolis-Hastings sampler for a fixed number of iterations, say $R$, to approximately sample from the empirical backward kernel. This method, proposed by Bunch and Godsill (2012), is referred to as Metropolis-Hastings FFBSi (MH-FFBSi).

Given a partial backward trajectory $\tilde{x}_{t+1:T}$, the backward weights are given by,

$$\tilde{w}_{i|T}^i \propto w_{i|T}^i f(\tilde{x}_{t+1} | x_t^i),$$

(50)

for $i = 1, \ldots, N$. Let $\{\nu_i\}_{i=1}^N$ be probabilities on the index set $\{1, \ldots, N\}$, defining a proposal distribution for an independent Metropolis-Hastings sampler. To propose a move
Algorithm 7 Metropolis-Hastings FFBSi (Bunch and Godsill, 2012)

Input: Forward filter particle systems \( \{x_t^i, w_t^i\}_{i=1}^N \) for \( t = 1, \ldots, T \).

Output: Backward trajectories \( \{\tilde{x}_t^j\}_{j=1}^M \).

1: Sample independently \( \{I(j)\}_{j=1}^M \sim \text{Cat}\left(\{w_t^j\}_{i=1}^N\right) \).
2: Set \( \tilde{x}_T^j = x_T^{I(j)} \) for \( j = 1, \ldots, M \).
3: for \( t = T - 1 \) to 1 do
4: \hspace{0.5cm} For \( j = 1, \ldots, M \), set \( I(j)[0] \) to the index of the ancestor of \( \tilde{x}_{t+1}^j \).
5: \hspace{0.5cm} for \( r = 1 \) to \( R \) do
6: \hspace{1cm} Sample independently \( \{C(j)\}_{j=1}^M \sim \text{Cat}\left(\{\nu_t^j\}_{i=1}^N\right) \).
7: \hspace{1cm} for \( j = 1 \) to \( M \) do
8: \hspace{1.5cm} With probability
9: \hspace{2cm} \begin{align*}
10: & 1 \wedge \frac{w_t^{C(j)} \nu_t^{I(j)[r-1]} f(x_{t+1}^{\tilde{x}_{t+1}^j} | x_t^{C(j)})}{w_t^{I(j)[r-1]} \nu_t^{C(j)} f(x_{t+1}^{\tilde{x}_{t+1}^j} | x_t^{I(j)[r-1]})}, \\
11: & \text{set } I(j)[r] = C(j) \text{ otherwise set } I(j)[r] = I(j)[r-1].
\end{align*}
12: \hspace{1cm} end for
13: \hspace{0.5cm} end for
14: \hspace{0.5cm} Set \( \tilde{x}_t^r = x_t^{I(j)[R]} \) and \( \tilde{x}_{t:T}^j = \{\tilde{x}_t^j, \tilde{x}_{t+1:T}^j\} \) for \( j = 1, \ldots, M \).
15: end for

from \( I \) to \( C \), we thus draw \( C \sim \text{Cat}\left(\{\nu_t^j\}_{i=1}^N\right) \) and accept this sample with probability

\begin{align*}
1 \wedge \frac{w_t^C f(\tilde{x}_{t+1} | x_t^{\tilde{x}_{t+1}^j}) \nu_t^I}{w_t^I f(\tilde{x}_{t+1} | x_t^{I[j]}) \nu_t^C}.
\end{align*}

(51)

Note that the acceptance probability can be evaluated in constant time (independent of \( N \)). After \( R \) iterations, the state of the Markov chain is used as an approximate sample from the empirical backward kernel.

To generate \( M \) backward trajectories, we have to run \( M \) independent chains. Hence, the computational complexity of MH-FFBSi is \( O(RM) \). However, Bunch and Godsill (2012) argue that a small number of iterations, say \( R = 10–30 \), are enough to get good performance in many cases. To avoid burn-in, they propose to initialize the Markov chains according to the genealogy of the forward filter. This means that, for \( R = 0 \), the MH-FFBSi reduces to a degenerate smoother relying on the forward filter particle trajectories. At the other extreme, as \( R \) tends to infinity, it approaches the FFBSi. The parameter \( R \) thus gives a trade-off between performance and computational cost. We summarize the MH-FFBSi sampler in Algorithm 7.

The use of MCMC within FFBSi opens up for additional modifications of the basic FFBSi, which is also recognized by Bunch and Godsill (2012). The backward simulators considered so far are limited by the fact that the states are only selected from those which appear in the collection of forward filter particles. If there is a significant discrepancy between the filtering and the smoothing distributions, then the forward filter particles are unlikely to be in the right locations of the state-space to represent the smoothing distribution well.

For this reason, Bunch and Godsill (2012) propose a second method geared toward this
issue. The method, referred to as Metropolis-Hastings forward filtering/backward proposing (MH-FFBP), proceeds in a similar manner as MH-FFBSi. However, at time \( t \), rather than running a Markov chain on the index set \( \{1, \ldots, N\} \) to select one of the forward filter particles \( \{x^i_t\}_{i=1}^N \), the chain runs on the product space \( \{1, \ldots, N\} \times X \) to sample jointly \( \{x^i_{t-1}, x^i_t\} \). That is, we draw one of the forward filter particles at time \( t - 1 \) and, given this particle, we propose a new value for \( x_t \) from some continuous proposal distribution on \( X \).

This approach is enabled by the fact that the backward kernel is given by a marginal of the joint density,

\[
p(x_{t-1:t} \mid x_{t+1}, y_{1:t}) \propto f(x_{t+1} \mid x_t)g(y_t \mid x_t)f(x_t \mid x_{t-1})p(x_{t-1} \mid y_{1:t-1}). \tag{52}
\]

Hence, for a fixed backward trajectory \( \tilde{x}_{t+1:T} \), if we obtain a sample \( x'_{t-1:t} \) from the joint kernel (52), we can simply discard \( x'_{t-1} \) and augment the backward trajectory according to \( \tilde{x}_{t:T} = \{x'_t, \tilde{x}_{t+1:T}\} \). By using the forward filter particles at time \( t - 1 \), (52) can be approximated by

\[
p(dx_{t-1:t} \mid \tilde{x}_{t+1}, y_{1:t}) \approx \sum_{i=1}^N \frac{w_{t-1}f(\tilde{x}_{t+1} \mid x_t)g(y_t \mid x_t)f(x_t \mid x_{t-1})dx_t}{\sum_{i=1}^N \frac{w_{t-1}p(\tilde{x}_{t+1}, y_t \mid x'_{t-1})}{\delta_{x'_{t-1}}(dx_{t-1})}}. \tag{53}
\]

To sample from (53) using MCMC, we choose a proposal distribution on \( \{1, \ldots, N\} \times X \),

\[
\nu_{t-1}q(x_t \mid \tilde{x}_{t+1}, y_t, x'_{t-1}), \tag{54}
\]

for \( i = 1, \ldots, N \). To propose a move from \( \{I, x_t\} \) to \( \{C, x'_t\} \), we first draw \( C \sim \text{Cat}(\{\nu_{t-1}^i\}_{i=1}^N) \) and then \( x'_t \sim q(x_t \mid \tilde{x}_{t+1}, y_t, x_{t-1}^C) \). The sample is accepted with probability

\[
1 \wedge \frac{w_{t-1}^Cf(\tilde{x}_{t+1} \mid x'_t)g(y_t \mid x'_t)f(x'_t \mid x_{t-1}^C)\nu_{t-1}^Cq(x_t \mid \tilde{x}_{t+1}, y_t, x_{t-1}^C)}{w_{t-1}f(\tilde{x}_{t+1} \mid x_t)g(y_t \mid x_t)f(x_t \mid x_{t-1})\nu_{t-1}q(x_t \mid \tilde{x}_{t+1}, y_t, x'_{t-1})}. \tag{55}
\]

As for MH-FFBSi, the acceptance probability can be evaluated in constant time. Hence, the computational complexity of MH-FFBP is of the same order, \( O(RM) \), though, the overhead is clearly larger. The benefit of MH-FFBP, as pointed out above, is that it is able to sample new positions for the particles when generating the backward trajectories, instead of just recycling the particles from the forward filter. The MH-FFBP is given in Algorithm 8. Note that a straightforward modification is needed for the proposal mechanism at time \( t = 1 \), but, for brevity, we have not made this explicit in the algorithm.

### 3.4.2 Metropolis-Hastings improved particle smoother

A method which is related to MH-FFBP is the Metropolis-Hastings improved particle smoother (MH-IPS), suggested by Dubarry and Douc (2011). The method is also reminiscent of the resample-move algorithm (Gilks and Berzuini, 2001). Instead of making use of the intermediate filtering distributions, as has been the common theme for the backward simulators considered so far, they start from the (degenerate) paths \( \{\tilde{x}^i_{1:T}\}_{i=1}^N \), obtained by resampling the forward filter trajectories at time \( T \). To increase the diversity among these paths, the particle positions are updated by running \( N \) independent, single-state MCMC samplers, one for each particle trajectory.
Algorithm 8 Metropolis-Hastings FFBP (Bunch and Godsill, 2012)

**Input:** Forward filter particle systems \( \{x_t^i, w_t^i\}_{i=1}^N \) for \( t = 1, \ldots, T \).

**Output:** Backward trajectories \( \{\tilde{x}_{1:T}^j\}_{j=1}^M \).

1. Sample independently \( \{I(j)\}_{j=1}^M \sim \text{Cat}\left(\{w_T^i\}_{i=1}^N\right) \).
2. Set \( \tilde{x}_{T}^j = x_T^{I(j)} \) for \( j = 1, \ldots, M \).
3. for \( t = T - 1 \) to 1 do
   4. For \( j = 1, \ldots, M \), set \( \tilde{x}_{t}^j[0] \) to the ancestor of \( \tilde{x}_{t+1}^j \) and set \( I(j)[0] \) to the index of the ancestor of \( \tilde{x}_{t}^j[0] \).
   5. for \( r = 1 \) to \( R \) do
      6. Sample independently \( \{C(j)\}_{j=1}^M \sim \text{Cat}\left(\{\nu_{t-1}^i\}_{i=1}^N\right) \).
      7. for \( j = 1 \) to \( M \) do
         8. Sample \( x_t^{I,j} \sim q(x_t \mid \tilde{x}_{t+1}^j, y_t, x_{t-1}^{C(j)}) \).
         9. With probability \( (55) \), set \( \{I(j)[r], \tilde{x}_{t}^j[r]\} = \{C(j), x_t^{I,j}\} \), otherwise set \( \{I(j)[r], \tilde{x}_{t}^j[r]\} = \{I(j)[r-1], \tilde{x}_{t}^j[r-1]\} \).
      10. end for
   11. end for
   12. Set \( \tilde{x}_{t:T}^j = \{\tilde{x}_{t}[R], \tilde{x}_{t+1:T}^j\} \) for \( j = 1, \ldots, M \).
   13. end for

If possible, the variables \( x_t \) are sampled from their full conditionals,

\[
p(x_t \mid x_{1:t-1}, x_{t+1:T}, y_{1:T}) \propto f(x_{t+1} \mid x_t) g(y_t \mid x_t) f(x_t \mid x_{t-1}). \tag{56}
\]

In the general case, however, these conditionals are not available. Instead, Hastings-within-Gibbs moves are used to update the state trajectories. Let \( \tilde{x}_{1:T} \) be the current state of one of the Markov chains (i.e. one of the current particle trajectories). To update the \( t \)th component, we simulate from some proposal density \( x_t^j \sim q(x_t \mid \tilde{x}_{t+1}, y_t, \tilde{x}_{t-1}) \), targeting (56). With probability

\[
1 \wedge \frac{f(\tilde{x}_{t+1} \mid x_t^j) g(y_t \mid x_t^j) f(x_t^j \mid \tilde{x}_{t-1}) q(\tilde{x}_t \mid \tilde{x}_{t+1}, y_t, \tilde{x}_{t-1})}{f(\tilde{x}_{t+1} \mid x_t^j) g(y_t \mid \tilde{x}_t) f(\tilde{x}_t \mid \tilde{x}_{t-1}) q(x_t^j \mid \tilde{x}_{t+1}, y_t, \tilde{x}_{t-1})}, \tag{57}
\]

the proposed sample is accepted and a new state trajectory is constructed according to \( \{\tilde{x}_{1:t-1}, x_t^j, \tilde{x}_{t+1:T}\} \). If not, the sample is rejected and the current trajectory \( \tilde{x}_{1:T} \) is retained. Obvious modifications to the proposal density and the acceptance probability are needed at times \( t = T \) and \( t = 1 \).

Dubarry and Douc (2011) suggest to sample the state variables in a deterministic order, backward in time, for \( t = T, \ldots, 1 \), which makes the MH-IPS reminiscent of a backward simulator. The incentive for this is to propagate the, in general, rich particle diversity at time points close to \( T \), down to time points far from \( T \). The MH-IPS is given in Algorithm 9, where we use a fixed number of \( R \) MCMC iterations. That is, we sweep through the data \( R \) times, from time \( t = T \) to time \( t = 1 \). The algorithm produces an (unweighted) collection of backward trajectories \( \{\tilde{x}_{1:T}^i[R]\}_{i=1}^N \) which can be seen as approximate draws from the JSD.

In the simulation studies conducted by Dubarry and Douc (2011), only a few MCMC
Algorithm 9 Metropolis-Hastings improved particle smoother (Dubarry and Douc, 2011)

**Input:** Forward filter particle trajectories \(\{x_{1:T}^i, w_{T}^i\}_{i=1}^N\).

**Output:** Improved particle trajectories \(\{\tilde{x}_{1:T}^i\}_{i=1}^N\).

1. Resample the forward filter particle system at time \(T\) to obtain an equally weighted particle system \(\{\tilde{x}_{1:T}^i\}_{i=1}^N\).
2. Initialize: set \(\tilde{x}_{1:T}^i[0] = x_{1:T}^i\) for \(i = 1, \ldots, N\).
3. for \(r = 1\) to \(R\) do
   4. for \(t = T\) to \(1\) do
      5. for \(i = 1\) to \(N\) do
         6. Sample \(x_{t}^{i,r} \sim q_t(x_t \mid \tilde{x}_{t+1}^i[r], y_t, \tilde{x}_{t-1}^i[r - 1])\).
         7. With probability given by (57), set \(\tilde{x}_{t}^i[r] = x_{t}^{i,r}\), otherwise set \(\tilde{x}_{t}^i[r] = \tilde{x}_{t}^i[r - 1]\).
     8. end for
   9. end for
10. end for
11. Set \(\tilde{x}_{1:T}^i = \tilde{x}_{1:T}^i[R]\) for \(i = 1, \ldots, N\).

iterations, say \(R = 5–10\), improve the particle diversity considerably. The computational complexity of MH-IPS is \(O(RN)\). Hence, MH-IPS can be an interesting, and computationally cheaper, alternative to FFBSi, much like the MH-FFBP. It should be noted, however, that the method relies on single-state updates, and it might thus to a larger extent than a backward simulator be subject to strong dependencies between the state variables.

The main differences between MH-FFBP and MH-IPS are: (i) MH-FFBP uses all the intermediate filtering approximations from the forward filter, whereas MH-IPS only makes use of the degenerate particle trajectories at time \(T\); (ii) MH-FFBP loops over the time indices \(t = T, \ldots, 1\) and runs \(R\) MCMC iterations for each time \(t\), whereas MH-IPS runs \(R\) MCMC iterations, looping over the time indices \(t = T, \ldots, 1\) in each iteration.

Finally, as pointed out by Dubarry and Douc (2011), the MH-IPS procedure does not have to be initialized by the forward filter trajectories. In fact, it can be used as an add-on to any particle smoother, to increase the diversity among the particle trajectories. Hence, it is possible to think of various combinations, such as FFBSi combined with MH-IPS or even MH-FFBP combined with MH-IPS.

### 3.5 Backward simulation for maximum likelihood inference

As noted in Section 1.5, state smoothing lies at the core of many common learning algorithms for SSMs. The particle smoothers discussed throughout this section can thus be useful for parameter inference in nonlinear and/or non-Gaussian models. To illustrate this, we consider the problem of maximum likelihood parameter inference in the parameterized SSM (3).

To compute the MLE (4), we make use of the EM algorithm as discussed in Section 1.5. However, computing the auxiliary quantity in the E-step amounts to solving a smoothing problem, which cannot be done in closed form for a general nonlinear/non-Gaussian
**Algorithm 10** Backward-simulation-based PSEM

1: Set $\theta[0]$ arbitrarily.
2: for $k \geq 1$ do
3: Run Algorithm 1 with $N_r$ particles, targeting $p_{\theta[k-1]}(x_{1:T} \mid y_{1:T})$.
4: Run a backward simulator to generate $M_r$ trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^{N_r}$.
5: Compute $\hat{Q}_r(\theta)$ according to (58).
6: Compute $\theta[r] = \arg \max_{\theta \in \Theta} \hat{Q}_r(\theta)$.
7: if convergence criterion is met then
8: break
9: end if
10: end for
11: return $\hat{\theta}_{\text{PSEM}} = \theta[r]$.

SSM. Instead, we run one of the FFBSi (e.g. Algorithm 6 on page 84, RS-FFBSi with early stopping) in the E-step. This results in an SMC-analogue of the well known Monte Carlo EM algorithm by Wei and Tanner (1990). Similar particle smoother EM (PSEM) algorithms have previously been successfully applied to maximum likelihood inference in challenging scenarios by, among others, Cappé et al. (2005); Olsson et al. (2008); Schön et al. (2011); Wills et al. (2013).

More precisely, given a collection of backward trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^{M_r}$ targeting the distribution $p_{\theta[k-1]}(x_{1:T} \mid y_{1:T})$, we compute an estimate of the auxiliary quantity (5) according to

$$\hat{Q}_r(\theta) = \frac{1}{M_r} \sum_{j=1}^{M_r} \log p_{\theta}(\widetilde{x}_{1:T}^j, y_{1:T}),$$

(58)

where the summand is given by (6). This Monte Carlo approximation is then maximized with respect to $\theta$ in the M-step, to obtain the next parameter iterate. To obtain convergence of PSEM, we require the accuracy of the approximation (58) to increase with the iteration number (Cappé et al., 2005; Fort and Moulines, 2003). Consequently, we need to let the number of particles $N_r$ and the number of backward trajectories $M_r$ increase with the iteration number $r$. One way to circumvent this issue will be presented in Section 5.6.

The backward-simulation-based PSEM algorithm is summarized in Algorithm 10. We illustrate the method in Example 9 below.

---

**Example 9: PSEM**

Consider again the nonlinear time-series model studied in Example 7. We let the process noise variance be given by $\sigma_v^2 = 1$ and the measurement noise variance be given by $\sigma_e^2 = 0.1$, and assume that these parameters are unknown. Given a batch of $T = 1500$ observations $y_{1:T}$, we wish to infer the unknown noise variances and thus set $\theta = (\sigma_v^2, \sigma_e^2)$. We apply Algorithm 10 using an RS-FFBSi sampler with early stopping in the E-step. The parameter estimates are initialized at $\theta[0] = (2, 2)$. We run the algorithm for 2000 iterations and let the number of particles $N_r$ and the number of backward trajectories $M_r$ increase cubically with the iteration number $r$, from 500 to 5000 and from 50 to 500,
respectively. The resulting parameter estimates $\theta[r]$ are shown in Figure 8. As can be seen, the estimates converge to values close to the true parameters. However, as pointed out above, it is necessary to increase the number of particles and backward trajectories with the iteration number to kill the Monte Carlo variance and obtain a convergent sequence of estimates.

**Figure 8:** Estimates of $\sigma_v^2$ (left) and $\sigma_e^2$ (right) vs. the iteration number $k$. The true parameter values are shown as dashed lines.

### 4 Backward simulation for general sequential models

So far, we have been looking closely at SSMs. In the previous sections, we saw how SMC and backward simulation can be used to sample approximately from the JSD $p(x_{1:T} \mid y_{1:T})$. One of the strengths of SMC, however, is that it is applicable to a much wider range of models than SSMs. The same is true for backward simulation. However, contrary to SMC, the application of backward simulation will in general be less straightforward when we leave the class of SSMs. In this section, we will derive a general backward simulator and discuss how this can be applied for inference in various models of interest. We will also highlight some pitfalls that might limit the applicability of backward simulation for certain types of models.

#### 4.1 Motivating examples

Let us start by considering a few examples of sequential inference problems which fall outside the class of SSMs. These examples are included as motivation for the development of the present section, but also to outline possible directions for future work. We will return to these examples in Section 6, when discussing possible extensions of the methods presented throughout this monograph. Readers not interested in these examples can safely skip directly to Section 4.2.
Figure 9: (Left) MRF with white nodes representing latent variables and small gray nodes representing observations; (Right) partitioning of the latent variables into groups $x_{I_1}$ and $x_{I_2}$ used in the tree sampling algorithm (for clarity, the nodes corresponding to the observations are not shown).

4.1.1 Blocked Gibbs sampling in Markov random fields

Markov random fields (MRFs) are a class of undirected graphical models. They play an important role in spatial statistics and computer vision, see e.g. Blake et al. (2011); Wainwright and Jordan (2008). Let $\{x_t\}_{t=1}^T$ be a collection of latent variables and let $\{y_t\}_{t=1}^T$ be a collection of observations. In this setting, $t$ is just an index variable for the data set and it has no temporal meaning. The conditional independence properties among these variables are specified in terms of an undirected graph with edges $E$ and vertices $V$ as illustrated in Figure 9 (left). The complete data likelihood can be factorized according to

$$p(x_{1:T}, y_{1:T}) = \frac{1}{Z} \prod_{i \in V} \phi(x_i, y_i) \prod_{(i,j) \in E} \psi(x_i, x_j),$$

(59)

where $Z$ is a normalization constant (referred to as the partition function) and the functions $\phi$ and $\psi$ are referred to as the observation potential and the interaction potential, respectively.

As in the case of SSMs, Gibbs samplers that sample the latent variables one at a time tend to be slow to converge, due to strong dependencies among the sampled variables. To alleviate this, a Gibbs sampler which exploits the structure of the MRF has been proposed by Hamze and de Freitas (2004). The method, referred to as a tree sampling algorithm, is based on the fact that the MRF graph can be partitioned into two disjoint chains, or more generally, trees; see Figure 9 (right). With $I_1$ and $I_2$ representing the vertex indices for the two trees, respectively, a two-stage Gibbs sampler is constructed according to:

(i) Draw $x'_{I_1} \sim p(x_{I_1} | x_{I_2}, y_{1:T});$
(ii) Draw $x'_{I_2} \sim p(x_{I_2} | x'_{I_1}, y_{1:T}).$
By block sampling the latent variables in each tree, a considerable improvement in mixing can be obtained compared to a one-at-a-time sampler (Hamze et al., 2006; Hamze and de Freitas, 2004). This is in agreement with our previous findings for SSMs.

Conditionally on $x_{I_2}$, the variables $x_{I_1}$ form an acyclic graph. Hamze and de Freitas (2004) assume that the latent variables are discrete valued (cf. a finite state-space HMM). Hence, it is possible to compute their exact marginal distributions by using belief propagation (Pearl, 1988). Based on these marginals, exact backward simulation can be employed to sample from the conditional densities in the above Gibbs scheme.

By using SMC it is possible to generalize this idea to models for which the exact marginals are not available. In particular, in combination with PMCMC (see Section 5), SMC-based backward simulation can be used within Gibbs sampling in a systematic manner. This approach would thus generalize the tree sampling algorithm to, for instance, models with continuous latent variables. This requires a backward simulator for the chain structures in the MRF in Figure 9. The same approach can also be applied to other types of graphical models, such as factor graphs (Hamze et al., 2006).

### 4.1.2 Inference strategies for optimal control problems

An interesting aspect of parameter inference is that it is useful, not only for learning predictive models, but also for policy optimization in control problems. Consider a dynamical system with state $x_t$, affected by a controlled input signal $u_t$,

$$x_{t+1} \sim f(x_{t+1} \mid x_t, u_t) \tag{60}$$

and $x_1 \sim \mu(x_1)$. For simplicity, we assume that the state is completely observed. We consider a state-feedback policy (i.e. a control law) $u_t = \pi_\theta(x_t)$, parameterized by some parameter $\theta$. Furthermore, let $r_t = r(x_t, u_t)$ be the reward, or return, obtained at time $t$. In optimal control problems we wish to find the policy which maximizes the expected future return,

$$V_\pi^\mu (\theta) = \mathbb{E} \left[ \sum_{t=1}^{\infty} \alpha^t r_t \right], \tag{61}$$

where $\alpha < 1$ is a discount factor. It has been recognized by Toussaint and Storkey (2006); Doucet et al. (2010); Hoffman et al. (2009b) that it is possible to view (61) as the normalization constant for an artificial trans-dimensional probability distribution, defined on $\bigcup_{T\in\mathbb{N}} \{T\} \times X_T$,

$$\tilde{p}_\theta(T, x_{1:T}) \triangleq \frac{r(x_T, u_T)\alpha^T}{V_\pi^\mu (\theta)} \mu(x_1) \prod_{t=1}^{T-1} f(x_{t+1} \mid x_t, u_t). \tag{62}$$

What is interesting with this reformulation is that maximization of the normalization constant is completely analogous to maximization of the marginal likelihood in the parameterized SSM (3). It follows that the inference strategies discussed in Section 1.5 can be used also for policy optimization. Indeed, EM algorithms have been used by Toussaint and Storkey (2006) and Hoffman et al. (2009a) for discrete and linear Gaussian models, respectively. In (Hoffman et al., 2009b; Doucet et al., 2010) the problem is addressed in a general setting by using trans-dimensional MCMC, such as reversible jump
samplers (Green, 1995). By factorizing \( \tilde{p}_\theta(T, x_{1:T}) = \tilde{p}_\theta(x_{1:T} \mid T)\tilde{p}_\theta(T) \), we note that \( \tilde{p}_\theta(x_{1:T} \mid T) \) takes the role of a joint smoothing density. Hence, the intermediate state inference step of these algorithms requires us to address a nonstandard “smoothing problem” for the artificial distribution \( \tilde{p}_\theta(x_{1:T} \mid T) \).

### 4.1.3 Gaussian process regression

The Markovian assumption in SSMs can be limiting for many applications of interest. Consider a regression problem with regressors \( \xi_t \in \mathbb{R}^n_\xi \) and observations \( y_t \in \mathbb{R}^n_y \). We observe a batch of data \( \{\xi_t, y_t\}_{t=1}^T \) and wish to find a predictive model for \( y_t \) given \( \xi_t \). As in Section 4.1.1, \( t \) is simply an index variable with no temporal meaning.

To model the dependency between the regressors and the observations, we use a nonparametric Gaussian process (GP) model. GPs are widely used for both classification and regression problems; see Rasmussen and Williams (2006) for a general introduction. In the regression setting, we have

\[
\begin{align*}
  f(\cdot) &\sim \mathcal{GP}(m(\xi), \kappa(\xi, \xi')) , \\
  x_t &= f(\xi_t) , \\
  y_t \mid x_t &\sim g(y_t \mid x_t),
\end{align*}
\]

where \( m(\cdot) \) is a mean function and \( \kappa(\cdot, \cdot) \) is a covariance kernel. Hence, the function \( f \) is modeled as a sample path from a GP, i.e. such that any finite collection of variables \( \{f(\xi_t)\}_{t \in \mathcal{I}} \) (where \( \mathcal{I} \) is an index set) has a joint Gaussian distribution,

\[
\{f(\xi_t)\}_{t \in \mathcal{I}} \sim \mathcal{N}(m, K),
\]

where \( m_t = m(\xi_t), \quad t \in \mathcal{I}, \quad K_{s,t} = \kappa(\xi_s, \xi_t), \quad s, t \in \mathcal{I} \).

The \( x_t \)'s can be seen as noise free sample points from the GP. These variables are latent, but observed indirectly through the measurement likelihood \( g(y_t \mid x_t) \). The model (63) is reminiscent of (2). However, for the GP model, the \( x_t \)'s are not given by a Markovian process and the model thus falls outside the class of SSMs. We return to this problem in Examples 11 and 12 below.

### 4.2 SMC revisited

The examples provided in the previous section are examples of latent variable models which are not given by SSMs. There are numerous other models of this kind for which SMC has been successfully applied, e.g. Dirichlet process mixture models (MacEachern et al., 1999; Fearnhead, 2004), phylogenetic trees (Bouchard-Côté et al., 2012) and agglomerative clustering models (Teh et al., 2008), to mention a few.

We thus seek a framework for backward simulation which is more generally applicable than the one presented in Section 3. For this cause, let \( \gamma_t(x_{1:t}) \) for \( t = 1, \ldots, T \) be a sequence of target densities on some increasing space \( X^t \). We assume that these densities can be written as

\[
\gamma_t(x_{1:t}) = \frac{\bar{\pi}_t(x_{1:t})}{Z_t},
\]
where the unnormalized density $\bar{\gamma}_t(x_{1:t})$ can be evaluated point-wise, whereas the normalization constant $Z_t$ is possibly unknown. SMC can be used to target any such sequence, in the same way as was done for the sequence of JSDs in Section 2. In that case, we had $\gamma_t(x_{1:t}) = p(x_{1:t} \mid y_{1:t})$ and $\bar{\gamma}_t(x_{1:t}) = p(x_{1:t}, y_{1:t})$. We will continue to refer to the index $t$ as time, even though it might not at all be a temporal index. Del Moral et al. (2006) employ SMC samplers for a sequence of target distributions on a common, fixed-dimensional space. However, by making use of auxiliary variables, they transform these problems into a form which coincides with the SMC framework discussed here.

Assume that \( \{x^i_{1:t-1}, w^i_{t-1}\}_{i=1}^N \) is a weighted particle system, targeting $\gamma_{t-1}(x_{1:t-1})$. This particle system is propagated to time $t$ by resampling and sequential importance sampling, in the same way as was done in Section 2. However, we now take a slightly different view on this procedure, which will prove to be convenient in the sequel.

Let $a_t^i$ be the index of the ancestor at time $t-1$, of particle $x_t^i$. That is, if $\{\tilde{x}^i_{1:t-1}\}_{i=1}^N$ is the collection of resampled particle trajectories at time $t-1$, we have $\tilde{x}^i_{1:t-1} = x^a_t^i_{1:t-1}$. The ancestor indices are auxiliary variables in the SMC sampler, and they were (among other things) used by Pitt and Shephard (1999) to derive the auxiliary particle filter (APF). When we write $x^k_{1:t}$, this should be interpreted as the ancestral path of the particle $x^k_t$.

Since the concepts of ancestor indices and ancestral paths will be of importance later on, we illustrate them with an example.

--- Example 10: Ancestral paths ---

Figure 10 shows the evolution of three particles for $t = 1, 2, 3$. At time $t = 1$, particle $x_1^2$ is resampled twice and particle $x_1^3$ is resampled once. At time $t = 2$, the ancestors are thus given by $a_1 = 2, a_2 = 2$ and $a_3 = 3$. Similarly, at time $t = 3$, the ancestors are given by $a_1 = 2, a_2 = 3$ and $a_3 = 3$. The ancestral path of particle $x_1^3$, which we denote $x_{1:3}$, is shown as a thick line in the figure. This path is given recursively from the ancestor indices,

\[
x_{1:3} = (x_1^{a_1}, x_2^{a_2}, x_3^{a_3}) = (x_1^2, x_2^2, x_3^3).
\]

Let us introduce a sequence of proposal kernels on the product space $\{1, \ldots, N\} \times X$,

\[
M_t(a_t, x_t) = w_{t-1}^{a_t} r_t(x_t \mid x_{1:t-1}^{a_t}),
\]

(65)
Algorithm 11 SMC (all operations are for $i = 1, \ldots, N$)

1: Draw $x_1^i \sim r_1(x_1)$.
2: Compute $\bar{w}_1^i = W_1(x_1^i)$.
3: Normalize: set $w_1^i = \bar{w}_1^i / \sum_{i=1}^{N} \bar{w}_1^i$.
4: for $t = 2$ to $T$ do
5: Draw $\{a_t^i, x_t^i\} \sim M_t(a_t, x_t)$.
6: Set $x_{1:t}^i = \{x_{1:t-1}^i, x_t^i\}$.
7: Compute $\bar{w}_t^i = W_t(x_{1:t}^i)$.
8: Normalize: set $w_t^i = \bar{w}_t^i / \sum_{i=1}^{N} \bar{w}_t^i$.
9: end for

The particle system $\{x_{1:t-1}^i, w_{t-1}^i\}_{i=1}^{N}$ can now be propagated to time $t$ by sampling independently from (65),

$$\{a_t^i, x_t^i\} \sim M_t(a_t, x_t), \quad (66)$$

for $i = 1, \ldots, N$. The particles $x_{1:t}^i = \{x_{1:t-1}^i, x_t^i\}$ are thereafter assigned (unnormalized) importance weights $\bar{w}_t^i = W_t(x_{1:t}^i)$, where the weight function is given by

$$W_t(x_{1:t}) = \frac{\bar{\gamma}_t(x_{1:t})}{\bar{\gamma}_{t-1}(x_{1:t-1}) r_t(x_t | x_{1:t-1})}. \quad (67)$$

As before, since the weights are only known up to proportionality, they are normalized to sum to one, $w_t^i = \bar{w}_t^i / \sum_{i=1}^{N} \bar{w}_t^i$. This results in a new weighted particle system $\{x_{1:t}^i, w_t^i\}_{i=1}^{N}$ targeting $\gamma_t(x_{1:t})$. In this formulation, the resampling step is implicit and corresponds to sampling the ancestor indices $a_t$ in (66).

The procedure is initialized by targeting $\gamma_1(x_1)$ using importance sampling. We thus sample from some proposal density $x_1^i \sim r_1(x_1)$ and compute importance weights $\bar{w}_1^i = W_1(x_1)$, where the weight function is given by $W_1(x_1) = \gamma_1(x_1) / r_1(x_1)$. The SMC sampler is summarized in Algorithm 11.

4.3 A general backward simulator

In many inferential problems for which SMC is applied, the actual quantity of interest is the final target density $\gamma_T(x_{1:T})$, or some marginal thereof. The densities $\gamma_t(x_{1:t})$ are then used as intermediate quantities, as a way to sequentially construct this target. The SMC procedure outlined above generates a sequence of weighted particle systems $\{x_{1:t}^i, w_t^i\}_{i=1}^{N}$ targeting the densities $\gamma_t(x_{1:t})$ for $t = 1, \ldots, T$. These systems define
empirical point-mass distributions according to
\[
\hat{\gamma}_t^N(dx_{1:t}) = \sum_{i=1}^{N} w_i^t \delta_{x_{1:t}^i}(dx_{1:t}).
\]  

Hence, an approximation of \(\gamma_T(x_{1:T})\) is given for \(t = T\). However, this approximation will be inaccurate due to path degeneracy (see Section 2.1.2). To mitigate this issue, we seek a way to increase the diversity of the particle trajectories used to approximate \(\gamma_T(x_{1:T})\). We will strive to do this in a similar way as for the FFBSi in the SSM setting, i.e. by reusing information from the intermediate approximations (68).

Consider the conditional density
\[
\gamma_T(x_{1:t} \mid x_{t+1:T}) = \frac{\gamma_T(x_{1:T})}{\int \gamma_T(x_{1:T}) \, dx_{1:t}}.
\]  

This density defines a transition kernel (the backward kernel) of invariant density \(\gamma_T\). Hence, if \(x_{1:T}\) is distributed according to \(\gamma_T\), we can generate a new sample with the same distribution by sampling from this kernel, much in the same way as in a Gibbs sampler. That is, we draw \(x'_{1:t} \sim \gamma_T(x_{1:t} \mid x_{t+1:T})\) and construct the trajectory \(\{x'_{1:t}, x_{t+1:T}\}\). This suggests that we can use the following strategy;

- Draw \(x'_{1:T} \sim \gamma_T(x_{1:T})\), set \(\tilde{x}_T = x'_T\) and discard \(x'_{1:T-1}\);
- For \(t = T-1, \ldots, 1\):
  - Draw \(x'_{1:t} \sim \gamma_T(x_{1:t} \mid \tilde{x}_{t+1:T})\);
  - Set \(\tilde{x}_{t:T} = \{x'_t, \tilde{x}_{t+1:T}\}\) and discard \(x'_{1:t-1}\).

The resulting trajectory \(\tilde{x}_{1:T}\) is distributed according to \(\gamma_T\). At first, this backward simulator might appear superfluous, since it is initialized by sampling directly from the target density of interest. However, in the case of SMC, it allows us to make use of the intermediate quantities (68) to improve upon the degenerate particle trajectories resulting from an initial SMC sweep.

The backward kernel for this simulator is given by
\[
\gamma_T(x_{1:t} \mid x_{t+1:T}) \propto \frac{\gamma_T(x_{1:T})}{\hat{\gamma}_t^N(x_{1:t})}. \tag{70}
\]  

By plugging (68) into this expression, we obtain an approximation of the backward kernel according to
\[
\hat{\gamma}_t^N(dx_{1:t} \mid \tilde{x}_{t+1:T}) = \sum_{i=1}^{N} \tilde{w}_i^t \delta_{x_{1:t}^i}(dx_{1:t}), \tag{71}
\]  

with
\[
\tilde{w}_i^t \propto w_i^t \frac{\hat{\gamma}_T(\{x_{1:t}^i, \tilde{x}_{t+1:T}\})}{\hat{\gamma}_t^N(x_{1:t}^i)}, \tag{72}
\]  

and where the weights are normalized to sum to one. Here, \(\{x_{1:t}^i, \tilde{x}_{t+1:T}\}\) should be understood as a point in \(X_T\) formed by concatenating the two partial trajectories. As ex-
Algorithm 12 Backward simulator

**Input:** Forward filter particle systems \( \{x^i_{1:t}, w^i_t\}_{i=1}^N \) for \( t = 1, \ldots, T \).

**Output:** Backward trajectories \( \{\tilde{x}^j_{t:1:T}\}_{j=1}^M \).

1. Sample independently \( \{b_T(j)\}_{j=1}^M \sim \text{Cat} \left( \{w^i_T\}_{i=1}^N \right) \).
2. Set \( \tilde{x}^j_T = x^b_T(j) \) for \( j = 1, \ldots, M \).
3. For \( t = T - 1 \) to 1 do
   4. For \( j = 1 \) to \( M \) do
      5. Compute
         \[
         \tilde{w}^{i,j}_t \propto w^i_t \frac{\gamma_T(\{x^i_{1:t}, \tilde{x}^j_{t+1:T}\})}{\gamma_t(x^i_{1:t})},
         \]
         for \( i = 1, \ldots, N \).
      6. Normalize the smoothing weights \( \{\tilde{w}^{i,j}_t\}_{i=1}^N \) to sum to one.
      7. Draw \( b_t(j) \sim \text{Cat} \left( \{\tilde{w}^{i,j}_t\}_{i=1}^N \right) \).
      8. Set \( \tilde{x}^j_t = x^{b_t(j)}_t \) and \( \tilde{x}^j_{t:T} = \{\tilde{x}^j_t, \tilde{x}^j_{t+1:T}\} \).
   9. End for
10. End for

Expected, for an SSM, with \( \gamma_t(x_{1:t}) = p(x_{1:t}, y_{1:t}) \), we obtain the weight expression (38). Using the empirical backward kernel, we construct an SMC-based backward simulator as in Algorithm 12. The algorithm generates \( M \) independent trajectories, similarly to the FF-BSi in Algorithm 4 (page 72). We illustrate the backward simulation on the nonparametric regression problem from Section 4.1.3.

---

**Example 11: Gaussian process regression**

Let us return to the GP regression problem considered in Section 4.1.3. We observe a heteroscedastic data set \( \{y_{1:T}, \xi_{1:T}\} \) for \( T = 1000 \) and wish to infer the momentaneous volatility. The data is illustrated in Figure 11. We put a zero-mean Gaussian process prior on the log-volatility, i.e. the model is given by

\[
\begin{align*}
    f(\cdot) & \sim \mathcal{GP}(0, \kappa(\xi, \xi')), \\
    x_t & = f(\xi_t), \\
    y_t & = e_t \exp \left( \frac{1}{2} x_t \right), \quad e_t \sim \mathcal{N}(0, 1).
\end{align*}
\tag{73-75}
\]

To complete the model, we use a covariance function in the Matérn class of kernels, namely,

\[
\kappa(\xi, \xi') = \left( 1 + \sqrt{3}|\xi - \xi'|/\ell \right) \exp \left( -\sqrt{3}|\xi - \xi'|/\ell \right).
\tag{76}
\]

For simplicity, we fix the length scale to \( \ell = 0.1 \). To infer the latent process \( x_{1:T} \), we employ an SMC sampler with \( N = 500 \) particles to the data set\(^3\). The results are given in the left panel of Figure 11. The SMC sampler clearly suffers from path degeneracy, resulting

---

\(^3\)For clarity of illustration, we sort the data points in increasing order of the regressors. However, this is not necessary and the algorithms can be applied using any ordering of the data points.
in poor regression results and overconfidence in the estimated volatility. To mitigate these issues, we complement the SMC sampler with a run of the backward simulator given in Algorithm 12. We generate $M = 100$ backward trajectories. The results are illustrated in the right panel of Figure 11. The backward simulator mitigates the degeneracy problem to a large extent and results in more reliable credibility intervals.

Figure 11: Regression results using SMC; (left) applied only in the forward direction; (right) complemented with a backward simulation pass. The dots show the observations and the blue lines the generated trajectories, representing the log-volatility.

The generality of Algorithm 12 suggests that the method can be used for a very wide range of problems. While this is indeed true, it should be noted that its usefulness will depend heavily on the properties of the problem at hand.

To start with, the computational complexity of Algorithm 12 is $O(MN)$, just as for the original formulation of the FFBSi. Unfortunately, for many models outside the class of SSMs, the rejection sampling technique applied in Section 3.3 is of limited use. The reason is that Algorithm 12 samples complete trajectories at each iteration. Hence, the sampling is done in a high-dimensional space and a rejection sampler will suffer from the curse of dimensionality, with very low acceptance probabilities as a result. Furthermore, the computation of the weights (72) will for many models scale with $T$ (see Section 4.5). This differs from the SSM setting, where the weights (38) are independent of $T$. Whether or not this is prohibitive clearly depends on the application.

Another pitfall to look out for is that the backward sampling weights can get badly skewed when there are strong and long-ranging dependencies among the variables $x_{1:T}$. If this is the case, there is a high probability that the backward simulator does not alter the trajectories generated in the forward pass to any considerable degree. The problem is illustrated in the example below.
Figure 12: Regression results using SMC; (left) applied only in the forward direction; (right) complemented with a backward simulation pass. The dots show the observations and the blue lines the generated trajectories, representing the log-volatility.

---

Example 12: Gaussian process regression, cont’d

Consider again the GP regression problem of Example 11. Assume that we want to impose further smoothness constraints on the latent process $x_t$. This can be done by modifying the covariance kernel of the GP prior. For instance, we can use a squared exponential kernel,

$$
\kappa(\xi, \xi') = \exp\left(-\frac{(\xi - \xi')^2}{2\ell^2}\right),
$$

which results in a very smooth process (Rasmussen and Williams, 2006). The same fixed length scale as before is used, $\ell = 0.1$. We apply an SMC sampler and a backward simulator to the data with $N = 500$ particles and $M = 100$ backward trajectories. The results are given in Figure 12. For this choice of covariance kernel, the results are quite different from what we experienced in Example 11. Most notably, the backward simulator does not provide any significant improvement over just running SMC in the forward direction, and the path degeneracy problem is still present. To generate a backward trajectory, at each time $t$ we consider the union of the partial backward trajectory $\tilde{x}_{t+1:T}$ with partial forward trajectories $x_{1:t}$. These unions, $\{x_{1:t}, \tilde{x}_{t+1:T}\}$, are used to compute the weights $\tilde{w}_{t+1:T}$; see (72). In order to improve the particle diversity in the backward pass, we rely on having several possible candidates among the partial forward trajectories, i.e. on having several weights $\tilde{w}_{t+1:T}$ which are significantly larger than zero. However, the long-ranging dependencies imposed by the squared exponential kernel tend to result in weights close to zero, except for the one particle $x_{1:t}$ which was the ancestor of $\tilde{x}_{t+1}$ in the forward pass. Hence, the backward simulator will with high probability just recover the ancestral paths from the SMC sampler, and consequently suffer from the same degeneracy problems.

The problem illustrated in the example above is caused by the strong dependence between the partial backward and partial forward trajectories. It was illustrated on the GP regression problem with a squared exponential kernel, because the long-ranging dependencies
imposed by this kernel (with this specific length scale) made the problem very distinct. However, it should be noted that the same effect can be experienced when there is a strong dependence between consecutive states, even when the dependence is not long-ranging. In fact, even in a Markovian SSM, we might get skewed weights when the transition density $f$ is “peaky”. The extreme case is of course when the transition kernel is degenerate, a case which is discussed in more detail in Section 4.6.1.

4.4 Rao-Blackwellized FFBSi

To illustrate the applicability of the general backward simulator, we consider a special case which is of particular interest. A popular approach to increase the efficiency of SMC samplers for SSMs is to marginalize over one component of the state, and apply an SMC sampler in the lower dimensional marginal space. This leads to what is known as the Rao-Blackwellized particle filter (RBPF) (Chen and Liu, 2000; Doucet et al., 2000a; Schön et al., 2005). It has been shown that the RBPF always will have lower asymptotic variance than the corresponding non-marginalized filter (Chopin, 2004; Lindsten et al., 2011). Intuitively, this can be understood by the fact that in an RBPF, the particles are spread in a lower dimensional space and will thus provide a denser point-mass approximation of the target distribution. The RBPF has been successfully applied to a range of applications, e.g. simultaneous localization and mapping (Montemerlo and Thrun, 2007; Montemerlo et al., 2002), aircraft positioning (Schön et al., 2005), underwater navigation (Karlsson and Gustafsson, 2003), communications (Wang et al., 2002; Chen et al., 2000) and audio source separation (Andrieu and Godsill, 2000).

The same approach has also been applied to state smoothing by Lindsten et al. (2013a); Särkkä et al. (2012); Whiteley et al. (2010); Fong et al. (2002), but it turns out that Rao-Blackwellization is less straightforward in this case. The reason is that the marginal state-process will be non-Markov. As an example, consider the following conditionally linear Gaussian state-space (CLGSS) model,

$$
x_{t+1} \sim f(x_{t+1} \mid x_t),
$$

$$
z_{t+1} = A(x_t)z_t + F(x_t)v_t,
\quad v_t \sim \mathcal{N}(0, I),
$$

$$
y_t = C(x_t)z_t + e_t,
\quad e_t \sim \mathcal{N}(0, R(x_t)).
$$

(76a)

(76b)

(76c)

If the state-variable $x_t$ can take only a finite number of values, the above model reduces to a jump Markov linear system where $x_t$ is a mode-variable determining which linear mode that is active at time $t$. The CLGSS model is thus a generalization of a switching system, with a (possibly) continuous mode variable.

The RBPF exploits the tractable substructure of the model through the factorization

$$
p(z_t, x_{1:t} \mid y_{1:t}) = p(z_t \mid x_{1:t}; y_{1:t})p(x_{1:t} \mid y_{1:t}).
$$

(77)

Since the model is CLGSS, the first factor of this expression is Gaussian and it can be evaluated by running a conditional Kalman filter (see e.g. Schön et al. (2005)). That is,

$$
p(z_t \mid x_{1:t}; y_{1:t}) = \mathcal{N}(z_t; \tilde{z}_{t|t}(x_{1:t}), P_{t|t}(x_{1:t})),
$$

(78)

for some tractable sequences of mean and covariance functions. The factor $p(x_{1:t} \mid y_{1:t})$ is targeted by an SMC sampler and it is approximated by a weighted particle system
\{x_{1:t}^i, w_{t|t}^i\}_{i=1}^N$. Consequently, we obtain the Gaussian mixture approximation,
\[
p(dz_t, dx_{1:t} \mid y_{1:t}) \approx \sum_{i=1}^N w_t^i \mathcal{N}(dz_t; \bar{z}_{t|t}^i, P_{t|t}^i) \delta_{x_{1:t}^i}(dx_{1:t}),
\]
where $\bar{z}_{t|t}^i = \bar{z}_{t|t}(x_{1:t}^i)$ and $P_{t|t}^i = P_{t|t}(x_{1:t}^i)$. Hence, the RBPF is an SMC sampler in which each particle is equipped with a Kalman filter, tracking the sufficient statistics for the conditional Gaussian densities (78).

Equivalently, we can view the marginalization of the $z$-process as a way to reduce the model (76) to,
\[
x_{t+1} \sim f(x_{t+1} \mid x_t),
\]
\[
y_t \sim p(y_t \mid x_{1:t}, y_{1:t-1}).
\]
Similarly to (78), the conditional density (80b) is Gaussian and it can be evaluated for any fixed marginal state trajectory $x_{1:t}$ by running a Kalman filter. The RBPF is then simply an SMC sampler targeting the sequence of conditional densities $p(x_{1:t} \mid y_{1:t})$ for the reduced latent variable model (80). This model shares many of the properties with the SSM (2). However, as an effect of the marginalization of the $z$-process, the measurement model (80b) depends on the complete history $x_{1:t}$ and the backward simulators derived in Section 3 are not applicable.

To construct a Rao-Blackwellized FFBSi particle smoother, we instead make use of the general backward simulator in Algorithm 12 (page 98). With $T$ being some final time point, the target distribution is given by $p(x_{1:T} \mid y_{1:T})$. To compute the weights (72), we need to evaluate the ratio
\[
\frac{\bar{\gamma}_T(x_{1:T})}{\bar{\gamma}_t(x_{1:t})} = \frac{p(x_{1:T}, y_{1:T})}{p(x_{1:t}, y_{1:t})} = p(x_{t+1:T}, y_{t+1:T} \mid x_{1:t}, y_{1:t}) \propto p(y_{t+1:T} \mid x_{1:T}, y_{1:t}) f(x_{t+1} \mid x_t).
\]
It remains to find an expression for the first factor of this expression (up to proportionality). In fact, this predictive density can be computed straightforwardly by running a conditional Kalman filter from time $t$ up to $T$. However, using this approach to calculate the weights at time $t$ would require $N$ separate Kalman filters to run over $T - t$ time steps, resulting in a total computational complexity scaling quadratically with $T$. For this specific model, it is possible to do better.

The idea is to propagate a set of statistics backward in time, as the backward trajectory $\bar{x}_{1:T}$ is generated. This approach has been used by Gerlach et al. (2000) for MCMC sampling and it has been adapted to Rao-Blackwellized backward simulation by Whiteley et al. (2010); Särkkä et al. (2012). Lindsten et al. (2013a) derive a similar method for a different type of CLGSS model, in which there is a dependence on $z_t$ in the updating equation (76a).

For the derivation below, we use the notation $\|\mu\|_\Omega^2 = \mu^T \Omega \mu$, where $\mu$ is a vector and
\( \Omega \succeq 0 \) is a positive semidefinite matrix. To compute (81) we consider the expression,

\[
p(y_{t+1:T} \mid x_{1:T}, y_{1:t}) = \int p(y_{t+1:T} \mid z_{t+1}, x_{t+1:T}) p(z_{t+1} \mid x_{1:t}, y_{1:t}) \, dz_{t+1}. \tag{82}
\]

The second factor of the integrand is given by a one-step prediction of the RBPF, similarly to (78),

\[
p(z_{t+1} \mid x_{1:t}, y_{1:t}) = \mathcal{N}(z_{t+1}; \tilde{z}_{t+1|t}(x_{1:t}), P_{t+1|t}(x_{1:t})). \tag{83}
\]

For later reference, we introduce a Cholesky factorization of the predictive covariance, \( P_{t+1|t}(x_{1:t}) = \Gamma_{t+1|t}(x_{1:t}) \Gamma_{t+1|t}(x_{1:t})^T \).

The key quantity is the first factor of the integrand in (82). We will show, by induction, that for any \( t = 1, \ldots, T - 1 \),

\[
p(y_{t+1:T} \mid z_{t+1}, x_{t+1:T}) \propto \exp \left( -\frac{1}{2} \left( \| z_{t+1} \|^2_{\Omega_{t+1}} - 2 \lambda_{t+1}^T z_{t+1} \right) \right) \tag{84}
\]

for some \( \Omega_{t+1} \succeq 0 \) and \( \lambda_{t+1} \) that only depend on \( y_{t+1:T} \) and \( x_{t+1:T} \).

First, we give a technical lemma. The proof is omitted for brevity, but follows straightforwardly by carrying out the integration.

**Lemma 1.** Let \( z = m + \Gamma \xi \) with \( \xi \sim \mathcal{N}(0, I) \) and let \( \Omega \succeq 0 \). Then

\[
\mathbb{E} \left[ \exp \left( -\frac{1}{2} \left( \| z \|^2_{\Omega} - 2 \lambda^T z \right) \right) \right] = |M|^{-\frac{1}{2}} \exp \left( -\frac{1}{2} \left( \| m \|^2_{\Omega^{-1}} - 2 \lambda^T m - \| \Gamma^T (\lambda - \Omega m) \|^2_{M^{-1}} \right) \right),
\]

with \( M = \Gamma^T \Omega \Gamma + I \).

To simplify the notation, we will write \( A_t \) for \( A(x_t) \) and similarly for other functions. First, we note that, by (76c),

\[
p(y_t \mid z_t, x_t) \propto \exp \left( -\frac{1}{2} \left( z_t^T \Gamma_t^{-1} z_t + 2 \Gamma_t^{-1} y_t \right) \right). \tag{85}
\]

Hence, at time \( T \), (84) holds with

\[
\Omega_T = C_T^T R_T^{-1} C_T, \tag{86a}
\]

\[
\lambda_T = C_T^T R_T^{-1} y_T. \tag{86b}
\]

Assume that (84) holds at time \( t + 1 \) and consider the factorization,

\[
p(y_{t:T} \mid z_t, x_{1:T}) = p(y_{t+1:T} \mid z_{t}, x_{t:T}) p(y_t \mid z_t, x_t). \tag{87}
\]

The first factor in (87) is given by

\[
p(y_{t+1:T} \mid z_t, x_{t:T}) = \int p(y_{t+1:T} \mid z_{t+1}, x_{t+1:T}) p(z_{t+1} \mid z_t, x_{t:T}) \, dz_{t+1}
\]

\[
= \int p(y_{t+1:T} \mid z_{t+1}, x_{t+1:T}) p(z_{t+1} \mid z_t, x_t) \, dz_{t+1}. \tag{88}
\]

Under the induction hypothesis and using Lemma 1 and (76b),

\[
p(y_{t+1:T} \mid z_t, x_{t:T}) \propto \exp \left( -\frac{1}{2} \xi_t \right), \tag{89a}
\]
Algorithm 13: Rao-Blackwellized FFBSi

**Input:** Forward filter particle systems \( \{x_t^i, w_t^i\}_{i=1}^N \) for \( t = 1, \ldots, T \) and linear state statistics \( \{\tilde{x}_{t+1|t}^i, \Gamma_{t+1|t}^i\}_{i=1}^N \) for \( t = 1, \ldots, T - 1 \).

**Output:** Backward trajectories \( \{\tilde{x}_{1:T}^j\}_{j=1}^M \).

1. Sample independently \( \{b_T(j)\}_{j=1}^M \sim \text{Cat}\left(\{w_T^i\}_{i=1}^N\right) \).
2. Set \( \tilde{x}_T^j = x_T^{b_T(j)} \) for \( j = 1, \ldots, M \).
3. Compute \( \{\Omega_T^j, \lambda_T^j\} \) according to (86) for \( j = 1, \ldots, M \).
4. for \( t = T - 1 \) to 1 do
5. for \( j = 1 \) to \( M \) do
6. for \( i = 1 \) to \( N \) do
7. Compute \( \{M_t^{i,j}, \eta_t^{i,j}\} \) according to (91).
8. Compute \( \tilde{w}_t^{i,j} \propto w_t^i | M_t^{i,j}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \eta_t^{i,j}\right) f(\tilde{x}_{t+1} | x_t^i) \).
9. end for
10. Normalize the smoothing weights \( \{\tilde{w}_t^{i,j}\}_{i=1}^N \) to sum to one.
11. Draw \( b_t(j) \sim \text{Cat}\left(\{\tilde{w}_t^{i,j}\}_{i=1}^N\right) \).
12. Set \( \tilde{x}_t^j = x_t^{b_t(j)} \) and \( \tilde{x}_{t:T}^j = \{\tilde{x}_T^j, \tilde{x}_{t+1:T}^j\} \).
13. Compute \( \{\Omega_t^j, \lambda_t^j\} \) according to (90).
14. end for
15. end for

with
\[
\xi_{t+1} = \|A_t z_t\|_{\Omega_{t+1}}^2 - 2\lambda_{t+1}^T A_t z_t - \|F_t^T (\lambda_{t+1} - \Omega_{t+1} A_t z_t)\|_{A_t^{-1}}^2, \tag{89b}
\]
and \( \Lambda_t = F_t^T \Omega_{t+1} F_t + I \). Combining (85) and (89) in (87) we get
\[
p(y_t: T \mid z_t, x_{t:T}) \propto \exp\left(-\frac{1}{2} \left(\|z_t\|_{\Omega_t}^2 - 2\Lambda_t^T z_t\right)\right)
\]
with
\[
\begin{align*}
\Omega_t &= A_t^T (I - \Omega_{t+1} F_t A_t^{-1} F_t^T) \Omega_{t+1} A_t + C_t^T R_t^{-1} C_t, \tag{90a} \\
\lambda_t &= A_t^T (I - \Omega_{t+1} F_t A_t^{-1} F_t^T) \lambda_{t+1} + C_t^T R_t^{-1} y_t, \tag{90b}
\end{align*}
\]
which completes the induction.

Finally, applying Lemma 1 to (82), with the integrands given by (83) and (84) results in,
\[
p(y_{t+1:T} \mid x_{1:T}, y_{1:t}) \propto |M_t|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \eta_t\right), \tag{91a}
\]
with
\[
M_t = \Gamma_{t+1|t}^T \Omega_{t+1} \Gamma_{t+1|t} + I, \eta_t = \|\tilde{z}_{t+1|t}\|_{\Omega_{t+1}}^2 - 2\Lambda_{t+1}^T \tilde{z}_{t+1|t} - \|\Gamma_{t+1|t}^T (\lambda_{t+1} - \Omega_{t+1} \tilde{z}_{t+1|t})\|_{M_t^{-1}}^2. \tag{91b}
\]
Using the above expression we can compute the backward sampling weights through (81). We present the adaption of the general backward simulator (Algorithm 12) to Rao-Blackwellized particle smoothing in Algorithm 13.
4.5 Non-Markovian latent variable models

By marginalization of the CLGSS model (76) we obtained the formulation (80). This model is in fact a special case of an important generalization of SSMs, the class of non-Markovian latent variable models given by

\[ x_{t+1} \sim f(x_{t+1} \mid x_{1:t}), \]
\[ y_t \sim g(y_t \mid x_{1:t}). \]  

Similarly to the SSM (2), this model is characterized by a latent process \( x_t \in X \) and an observed process \( y_t \in Y \). However, it does not share the conditional independence properties that are central to SSMs. Instead, both the transition density \( f \) and the measurement density \( g \) may depend on the entire past history of the state trajectory.

In Section 4.6 we will see additional examples in which non-Markovian models arise by some manipulation of an SSM, similarly to the marginalization discussed in the previous section. Another example is the GP regression model considered in Section 4.1.3, which also can be written on the form (92).

With the target density being \( \gamma_t(x_{1:t}) = p(x_{1:t} \mid y_{1:t}) \), this model fits into the framework presented in Section 4.3. As we saw in the previous section, to compute the backward simulation weights (72) we need to evaluate the ratio

\[ \frac{\tilde{\gamma}_T(x_{1:T})}{\tilde{\gamma}_t(x_{1:t})} = \frac{p(x_{1:T}, y_{1:T})}{p(x_{1:t}, y_{1:t})} = \prod_{s=t+1}^{T} g(y_s \mid x_{1:s}) f(x_s \mid x_{1:s-1}). \]  

For the CLGSS model (76), we found a backward recursion for a set of statistics, which enabled the evaluation of this expression in constant time. However, in the general case, the computational cost of evaluating (93) will increase with \( T \). For instance, if the functions \( f \) and \( g \) can be evaluated in constant time, then the computational cost of evaluating (93) scales linearly with \( T \). This implies that the cost of generating a full backward trajectory is \( O(T^2) \). Since \( T \) is typically large, an \( O(T^2) \) computational complexity can be prohibitive for many applications.

One way to mitigate this issue has been proposed by Lindsten et al. (2012). They consider non-Markovian models in which there is a decay in the influence of the past on the present, akin to that in SSMs but without the strong Markovian assumption. It is thus possible to obtain a useful approximation of (93) by truncating the product to a smaller number of factors. Lindsten et al. (2012) propose an adaptive method, in which the factors of (93) are computed sequentially until some criterion is met, after which the product is truncated. They also propose a specific backward-simulation-based inference method, particle Gibbs with ancestor sampling, which is suitable for inference in this type of models. This method will be reviewed in detail in Section 5.5.

4.6 From state-space models to non-Markovian models

Rao-Blackwellization, as considered in Section 4.4, resulted in a non-Markovian latent variable model. To provide further insight into this model class, we present a few additional examples of SSMs for which backward simulation is problematic in the original formulation. A transformation to a non-Markovian model is then useful in order to enable...
backward simulation, by using the general sampler provided in Algorithm 12 on page 98.

### 4.6.1 Degenerate state-space models

As was pointed out in Section 3, the approximation of the backward kernel (35) relies on the assumption that the model under study is fully dominated. Hence, the backward simulators for SSMs derived in Section 3 are not applicable for degenerate models. This can be understood by noting that, if the model is degenerate, then so is the backward kernel. Consequently, the support of the backward kernel is limited to some low-dimensional subspace or manifold, embedded in the state-space. Due to this, it cannot be approximated in a natural way by the forward filter particles. We illustrate the problem in the following example.

---

**Example 13: Degenerate backward kernel**

To see why backward simulation is problematic for degenerate models, we consider a simple toy model. Let the state be given by \( x_t = (x_{1,t}, x_{2,t})^\top \in \mathbb{R}^2 \). The initial state is distributed according to \( x_1 \sim \mathcal{N}(0, I) \) and the evolution of the state process is given by \( x_{t+1} = x_t + v_t \) with,

\[
v_t \sim \mathcal{N}\left( \begin{pmatrix} 10 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \right).
\]

Since the process noise covariance is singular, the model is degenerate (see Section 1.4). Still, it is straightforward to apply SMC to this model. Assume that we apply a bootstrap PF with \( N = 10 \) particles. The particles generated at times \( t = 1 \) and \( t = 2 \) are shown in Figure 13.

From the definition (10), the backward kernel at time \( t = 1 \) is given by \( B_1(A \mid \tilde{x}_2, y_1) = \mathbb{P}(x_1 \in A \mid \tilde{x}_2, y_1) \). Since the transition kernel is degenerate, it holds that \( B_1(\{x_1 \in \mathbb{R}^2 : x_{1,2} \neq \tilde{x}_{2,2}\} \mid \tilde{x}_2, y_1) = 0 \). That is, for a given particle \( \tilde{x}_2 \) there is, with probability 1, only one particle at time \( t = 1 \) which is contained in the support of the backward kernel \( B_1(\cdot \mid \tilde{x}_2, y_1) \), namely the ancestor particle of \( \tilde{x}_2 \).

The same argument can be applied for any time \( t \). If we apply a backward simulator to this model, we will only recover the genealogy of the forward PF and the backward simulator.
is not able to improve upon the degenerate paths of the forward filter. For this specific model, the $x_{2,t}$-state is completely deterministic given the initial state. However, the same problem will arise whenever the transition kernel is degenerate, but in general with more complicated constraints on the individual state components as a result.

This restriction of backward simulators for SSMs is unfortunate, since many interesting dynamical systems are most naturally modelled as degenerate. For instance, consider a nonlinear system with additive noise on the form,

$$\xi_{t+1} = f_\xi(\xi_t) + G \nu_t,$$

$$y_t = h_\xi(\xi_t) + e_t,$$

where $G$ is a tall matrix and, consequently, $\text{rank}(G) < \text{dim}(\xi_t)$. Hence, the process noise covariance matrix $GG^T$ is singular. SMC samplers can straightforwardly be applied to this type of models, but as pointed out above, smoothing via backward simulation is more problematic.

To alleviate the problem, we can recast the degenerate SSM as a non-Markovian model in a lower dimensional space. For the model considered in Example 13, the state $x_{2,t}$ is superfluous. Hence, the model can be reformulated by removing $x_{2,t}$ and running the backward simulator only for the $x_{1,t}$-state. A similar approach can be applied to the additive noise model (94). Let $G = U \Sigma V^T$ with unitary $U$ and $V$ be a singular value decomposition of $G$. Furthermore, let,

$$\begin{bmatrix} x_{t+1} \\ z_{t+1} \end{bmatrix} \triangleq U^T \xi_{t+1} = U^T f(UU^T \xi_t) + \begin{bmatrix} \Sigma V^T \nu_t \\ 0 \end{bmatrix}.$$  

(95)

Hence, with $\nu_t \triangleq \Sigma V^T \nu_t$ and by appropriate definitions of the functions $f_x$, $f_z$ and $h$, the model (94) can be rewritten as,

$$x_{t+1} = f_x(x_t, z_t) + v_t,$$

$$z_{t+1} = f_z(x_t, z_t),$$

$$y_t = h(x_t, z_t) + e_t.$$  

(96a, 96b, 96c)

For simplicity, assume that $z_1$ is known. If this is not the case, $z_1$ can be included in the system state or treated as an unknown static parameter of the model. Hence, the sequence $z_{1:t}$ is $\sigma(x_{1:t-1})$-measurable and we can write $z_t = z_t(x_{1:t-1})$. We can then further rewrite the model as,

$$x_{t+1} = f_x(x_{1:t}) + v_t,$$

$$y_t = h(x_{1:t}) + e_t.$$  

(97a, 97b)

which is a non-degenerate, non-Markovian model. This reformulation illustrates the intimate relationship between degenerate models and non-Markovian models. In fact, this is nothing but another application of marginalization as discussed in Section 4.4, where the $z$-state is conditionally deterministic and thus trivially marginalizable.

The model (97) will in general not allow for a simple evaluation of the expression (93) in constant time. Hence, the computational complexity for applying backward simulation...
in this model is $O(T^2)$. Whether or not this is prohibitive clearly depends upon the application. One possibility to reduce the complexity is to make use of a truncation as discussed in Section 4.5.

### 4.6.2 Collapsing of state-space models

Another subclass of SSMs, for which backward simulation in problematic, is that in which the transition density function $f$ is not available on closed form. It is quite common that the transition density $f$ can be simulated from, but not evaluated point-wise, see e.g. Fearnhead et al. (2008); Golightly and Wilkinson (2008); Murray et al. (2012); Hall et al. (2012). If this is the case, it is not possible to evaluate the backward sampling weights (38), since these depend explicitly on $f$.

To find a way around this, we note that if $f(x_t \mid x_{t-1})$ can be simulated from, then this is typically done by generating a random variable $v_t \sim p(v_t)$ and computing $x_t = a(x_{t-1}, v_t)$ for some function $a$. By a similar argument as in the previous section, it follows that $x_t$ is $\sigma(v_{1:t})$-measurable. Hence, the model can be written as,

$$v_t \sim p(v_t),$$

$$y_t \sim g(y_t \mid v_{1:t-1}),$$

which is a non-Markovian model with latent variables $v_t \in V$. This formulation has been exploited by Murray et al. (2012); Hall et al. (2012) to construct auxiliary particle filters for SSMs in which the transition density function is not available on closed form. By using Algorithm 12 we can employ backward simulation for the model (98), thus simulating the innovation process $v_{1:T}$. Again, the computational complexity for evaluating the backward sampling weights will be $O(T^2)$ in the general case, but truncation can be used to reduce the complexity at the cost of an approximation error.

### 4.6.3 Non-centered parameterizations

A different reason for reformulating an SSM into the form (98), can arise in the context of parameter inference using data augmentation, e.g. Gibbs sampling. Assume that we have a parametric model of an SSM as in (3). For simplicity, we assume that only the transition density function (3a) is parameterized by $\theta$. In the Bayesian setting, i.e. with $\theta$ modeled as a random variable, a graphical model for this SSM is given in Figure 14 (left).

Papaspiliopoulos et al. (2003) refer to this as a centered parameterization. This parameterization suggest a Gibbs sampler according to;

(i) Draw $\theta' \sim p(\theta \mid x_{1:T}, y_{1:T})$;

(ii) Draw $x_{1:T}' \sim p(x_{1:T} \mid \theta', y_{1:T})$.

The conditioning on $x_{1:T}$ in Step (i) typically allows for a fairly straightforward updating of $\theta$. Step (ii) can be addressed by using a backward simulator. In particular, the PMCMC framework (see Section 5 and, in particular, Sections 5.4–5.5) provides the means of using SMC-based backward simulators within Gibbs sampling.

However, as can be seen in Figure 14, $x_{1:T}$ and $\theta$ exhibit a priori dependence. In many cases, this dependence can be very strong, resulting in slow convergence of the Gibbs sampler outlined above. By instead making use of the formulation in (98), we can choose
the variables $v_{1:T}$ to be a priori independent of $\theta$. The graphical model is given in Figure 14 (right). This is referred to as a non-centered parameterization, and it results in the Gibbs sampler:

(i)' Draw $\theta' \sim p(\theta | v_{1:T}, y_{1:T})$;

(ii)' Draw $v'_{1:T} \sim p(v_{1:T} | \theta', y_{1:T})$.

Often, the non-centered parameterization can result in a Gibbs sampler with better convergence properties than the centered parameterization, though the opposite is also possible (see Papaspiliopoulos et al. (2003) for a discussion). However, Steps (i)'--(ii)' are typically more computationally involved than Steps (i)--(ii). In particular, Step (ii)' requires the simulation of a posterior sample from the innovation process $v_{1:T}$. As pointed out above, backward simulation in the model (98) can lead to a high computational complexity. Consequently, the potential benefit of using a non-centered parameterization has to be evaluated on a case by case basis.

### 5 Backward simulation in particle MCMC

Particle Markov chain Monte Carlo (PMCMC) is a systematic way of combining the two main tools used in Monte Carlo statistical inference; SMC and MCMC. This is a class of inferential methods, introduced by Andrieu et al. (2010), in which SMC is used to construct proposal mechanisms for MCMC samplers. Within this framework, backward simulation has proved to be an important component. This section is devoted to PMCMC, and in particular to how we can benefit from using backward simulation within this framework.

#### 5.1 Introduction to PMCMC

In the previous sections, we have mostly been focusing on state-inference under the assumption that the parameters of the model are known (or at least fixed). However, what we are often interested in is to make posterior inference about some static parameter $\theta \in \Theta$ of the model; see Section 1.5.

As an example, consider the parameterized SSM (3), with prior density $\pi(\theta)$ for the parameter. Given a batch of observations $y_{1:T}$, we seek the posterior density $p(\theta | y_{1:T}) \propto p(y_{1:T} | \theta) \pi(\theta)$.
This posterior is typically not available in closed form. Furthermore, for a general nonlinear/non-Gaussian SSM, it is problematic to design an MCMC sampler targeting \( p(\theta \mid y_{1:T}) \) directly, since this would require us to analytically marginalize the states \( x_{1:T} \). As noted in Section 1.5, a possible remedy is to include the latent states \( x_{1:T} \) as auxiliary variables and instead target the joint posterior \( p(\theta, x_{1:T} \mid y_{1:T}) \) with an MCMC sampler. For example, this opens up for Gibbs sampling using the following scheme:

(i) Draw \( \theta' \sim p(\theta \mid x_{1:T}, y_{1:T}) \);

(ii) Draw \( x'_{1:T} \sim p(x_{1:T} \mid \theta', y_{1:T}) \).

Sampling \( \theta \) in Step (i), i.e. conditionally on both the observed data and the latent states, is in general much easier than sampling \( \theta \) conditionally only on the observed data. In fact, if conjugate priors are used, this step can be carried out exactly. For non-conjugate models, one option is to replace Step (i) with a Metropolis-Hastings step, which is possible since the unnormalized density \( p(\theta, x_{1:T}, y_{1:T}) \) can be evaluated point-wise.

Step (ii) of the above Gibbs sampler, however, is less straightforward since it requires us to generate a sample from the JSD for a fixed value of the system parameter. In some special cases this can be done exactly. For instance, if the model is linear Gaussian, we can make use of the backward simulator for LGSS models derived in Section 1.7 to sample \( x'_{1:T} \).

For a general nonlinear/non-Gaussian model, on the other hand, the JSD is not available and the second step of the Gibbs sampler cannot be carried out. However, with the material of Sections 3 and 4 in mind, a natural idea is to employ SMC to approximately sample from the JSD, using e.g. the backward simulator of Algorithm 4 on page 72.

However, to simply replace Step (ii) of the Gibbs sampler with a backward simulator, will result in a method suffering from some serious drawbacks. Since it is based on SMC, we only obtain an approximate sample from the JSD by running Algorithm 4. This will introduce additional bias and it is not clear how the error will propagate through the Gibbs sampler. As a consequence, the success of this approach relies heavily on using a large number of particles \( N \) in the underlying SMC sampler, to obtain accurate backward kernel approximations. This might seem like a natural requirement, but it also means that we waste a lot of computational resources. The reason is that, at each iteration of the Gibbs sampler, we run an SMC sampler with a large number of particles, extract a single state trajectory and then discard all the remaining particles. As we will see in Section 5.4, it is possible to do much better if we, instead of viewing the Gibbs sampler and the SMC sampler as two separate components, analyze them together.

We will start our exploration of the PMCMC framework by reviewing the particle marginal Metropolis-Hastings (PMMH) sampler by Andrieu et al. (2010) in Section 5.2. PMMH is an example of a pseudo-marginal method, meaning that it uses unbiased likelihood estimates in place of exact likelihoods in an MCMC sampler (Beaumont, 2003; Andrieu and Roberts, 2009). Somewhat surprisingly, this will not alter the stationary distribution of the chain. PMMH is a good starting point for introducing PMCMC, and we will fall back on this material in the later sections. In Section 5.3 we will see how backward simulation...
can be used to improve the quality of the PMMH estimates, using a method proposed by Olsson and Rydén (2011).

We then turn to a different member of the PMCMC family, particle Gibbs (PG), for which backward simulation has turned out to play a quite different role. In Sections 5.4 and 5.5, we will discuss two approaches to backward simulation in the context of PG, which enables us to employ this method even for large data records and using very few particles. These two sections cover material previously presented by Whiteley et al. (2010); Lindsten and Schön (2012); Lindsten et al. (2012).

5.2 Particle Marginal Metropolis-Hastings

We will work within the general framework introduced in Section 4, but now with the addition of an unknown static parameter \( \theta \). Hence, the target density on \( \Theta \times X^T \) is given by

\[
\gamma_T(\theta, x_{1:T}) = \frac{\bar{\gamma}_T(\theta, x_{1:T})}{Z_T} = \frac{\bar{\gamma}_T^\theta(x_{1:T})\pi(\theta)}{Z_T},
\]

(99)

where \( \pi(\theta) \) is the prior density for the parameter and \( Z_T \) is a normalization constant. When using \( \theta \) as a superscript, it should be thought of as conditionally on \( \theta \), but for notational simplicity we prefer to write \( \bar{\gamma}_T^\theta(x_{1:T}) \) rather than \( \bar{\gamma}_T(x_{1:T} \mid \theta) \), etc.

For fixed \( \theta \), a sequence of intermediate densities

\[
\gamma_t^\theta(x_{1:t}) = \frac{\bar{\gamma}_t^\theta(x_{1:t})}{Z_t(\theta)}
\]

(100)

for \( t = 1, \ldots, T \) is available, analogously to (64). Note that the normalization factor \( Z_t(\theta) = \int \bar{\gamma}_t^\theta(x_{1:t}) \, dx_{1:t} \) depends on \( \theta \). These densities can sequentially be approximated by the application of an SMC sampler, analogously to Section 4.2 (simply add \( \theta \) to the notation for all quantities introduced in Section 4.2).

Example 14: The special case of SSMs

In the special case of an SSM, we typically have

\[
\gamma_T(\theta, x_{1:T}) = p(\theta, x_{1:T} \mid y_{1:T}) \quad \text{and} \quad \bar{\gamma}_T(\theta, x_{1:T}) = p(\theta, x_{1:T}, y_{1:T}).
\]

Hence, (99) corresponds to

\[
p(\theta, x_{1:T} \mid y_{1:T}) = \frac{p(\theta, x_{1:T}, y_{1:T})}{p(y_{1:T})} = \frac{p(x_{1:T}, y_{1:T} \mid \theta)\pi(\theta)}{p(y_{1:T})},
\]

with the normalization constant \( Z_T = p(y_{1:T}) \). Similarly, for fixed \( \theta \), the intermediate quantities of the SMC sampler are the JSDs, so (100) corresponds to

\[
p(x_{1:t} \mid \theta, y_{1:t}) = \frac{p(x_{1:t}, y_{1:t} \mid \theta)}{p(y_{1:t} \mid \theta)},
\]

with the conditional normalization constant \( Z_t(\theta) = p(y_{1:t} \mid \theta) \).

For large \( T \), the high dimension of the space \( \Theta \times X^T \) is prohibitive when designing an MCMC sampler for the target density (99). As previously pointed out, we will try to
alleviate this by constructing a proposal based on SMC, and use this within a Metropolis-Hastings sampler. To start with, consider a proposal kernel on $\Theta \times X^T$ given by,

$$q(x'_{1:T}, \theta' | \theta, x_{1:T}) = q(\theta' | \theta) \gamma^\theta_T(x'_{1:T}). \quad (101)$$

This proposal makes use of a marginal Markov kernel $q(\theta' | \theta)$ on $\Theta$ and is perfectly adapted to the target on $X^T$. This proposal cannot be used in practice, since the second factor on the right-hand side, $\gamma^\theta_T(x_{1:T})$, is assumed to be unavailable. However, this factor can be approximated using SMC, resulting in an empirical point-mass distribution, similarly to (68),

$$\tilde{\gamma}^\theta_{T,N}(dx_{1:T}) \triangleq \sum_{i=1}^N w_i \delta_{x_i^{1:T}}(dx_{1:T}). \quad (102)$$

In order to mimic the proposal kernel (101), we can thus make use of the following sampling strategy;

(i) Draw $\theta' \sim q(\theta' | \theta)$;

(ii) Parameterize the model with $\theta'$ and run an SMC sampler (Algorithm 11, page 96) targeting $\gamma^\theta_T(x_{1:T})$;

(iii) Draw $k$ with $\Pr(k = i) = w_i$ and set $x'_{1:T} = x^k_{1:T}$.

Note that the particle trajectory generated in Step (iii) is the ancestral path of particle $x^k_{1:T}$, as discussed in Example 10. That is, we can write

$$x^k_{1:T} = x_{1:T}^{b_1} \triangleq (x^{b_1}_1, \ldots, x^{b_T}_T), \quad (103)$$

where the indices $b_{1:T}$ are given recursively by the ancestor indices; $b_T = k$ and $b_t = a_{t+1}^{b_{t+1}}$.

The above sampling procedure can straightforwardly be implemented and it defines a proposal kernel on $\Theta \times X^T$, akin to the “ideal” kernel (101). The problem, however, in using this proposal mechanism in a Metropolis-Hastings sampler lies in the computation of the acceptance probability. To compute the acceptance probability, we need to be able to evaluate the proposal kernel density point-wise. For the sampling scheme defined by Steps (i)–(iii) above, the proposal kernel is given by

$$q(d\theta' | \theta) \mathbb{E} \left[ \sum_{i=1}^N w_i \delta_{x_i^{1:T}}(dx_{1:T}) \right], \quad (104)$$

where the expectation is taken w.r.t. the randomness of the SMC sampler (the weights $\{w_i\}_{i=1}^N$ and the particles $\{x_i^{1:T}\}_{i=1}^N$ are random variables). In other words, to be able to evaluate the proposal kernel, we have to marginalize over all the random variables generated by the SMC sampler, which is an utterly hopeless task.

The solution to this problem, as often, lies in including the random variables generated by the SMC sampler as auxiliary variables in the sampling scheme. By doing so, we avoid the need to marginalizing over them. For this cause, we introduce the boldface notation

$$\mathbf{x}_t = \{x^1_t, \ldots, x^N_t\}, \quad \mathbf{a}_t = \{a^1_t, \ldots, a^N_t\}, \quad (105)$$
to refer to all the particles and ancestor indices, respectively, generated by the SMC sampler at time $t$. It follows that, for a fixed $\theta$, the SMC sampler in Algorithm 11 (page 96) generates a single sample on the extended space $X^{NT} \times \{1, \ldots, N\}^{N(T-1)}$. The probability density of this sample is given by,

$$
\psi^\theta(x_{1:T}, a_{2:T}) \triangleq \prod_{i=1}^N r_1^\theta(x_1^i) \prod_{t=2}^T \prod_{i=1}^N M_t^\theta(a_t^i, x_t^i).
$$

That is, at time $t = 1$, we sample $\{x_1^i\}_{i=1}^N$ independently from the proposal density $r_1^\theta(x_1)$. Then, for each time point $t = 2, \ldots, T$, we sample $\{a_t^i, x_t^i\}_{i=1}^N$ independently from the proposal kernel $M_t^\theta(a_t, x_t)$. Recall that $M_t^\theta(a_t, x_t)$ depends on all the random variables generated up to time $t - 1$, i.e. $\{x_{1:t-1}, a_{2:t-1}\}$, but we will not make this dependence explicit for notational convenience.

Using the notion of sampling on an extended space, the procedure given by Steps (i)–(iii) above can be seen as generating a sample $\{\theta', x_{1:T}, a_{2:T}, k\}$ in the space $\Omega \triangleq \Theta \times X^{NT} \times \{1, \ldots, N\}^{N(T-1)+1}$. Furthermore, the density of these random variables on $\Omega$ is given by,

$$
q(\theta' \mid \theta) \psi^\theta(x_{1:T}, a_{2:T}) w_k^k,
$$

where the three factors, from left to right, correspond to the three steps of the sampling procedure.

We are now in an unusual situation – we wish to use the proposal kernel (107), which we can call

$$
\gamma^T(\theta, x_{1:T}) \triangleq \gamma^T(\theta, x_{1:T}) \triangleq \gamma^T(x_{1:T} \mid \theta, x_{1:T})
$$

as a surrogate for the original target $\gamma^T(\theta, x_{1:T})$. If $\phi$ admits $\gamma^T$ as a marginal and if we are able to design an MCMC sampler with stationary distribution $\phi$, then we implicitly target also $\gamma^T$. More precisely, the subchain constructed by only considering the variables corresponding to this marginal will have $\gamma^T$ as stationary distribution.

The second requirement is more vague, but it is crucial in order to obtain an efficient (and simple) algorithm. Guided by these requirements, we define the extended target density on $\Omega$ according to,

$$
\phi(\theta, x_{1:T}, a_{2:T}, k) = \phi(\theta, x_{1:T}^{b_1:T}, b_{1:T}) \phi(x_{1:T}^{-b_1:T}, a_{2:T}^{-b_2:T} \mid \theta, x_{1:T}^{b_1:T}, b_{1:T})
$$

\[ \triangleq \frac{\gamma^T(\theta, x_{1:T})}{N^T} \prod_{i=1}^N r_1^\theta(x_1^i) \prod_{t=2}^T \prod_{i=1}^N M_t^\theta(a_t^i, x_t^i). \]
In the above expression, we have introduced the notation
\[ x_t^{-1} = \{x_t^1, \ldots, x_t^{i-1}, x_t^{i+1}, \ldots, x_t^N\}, \quad x_{1:T}^{b_1:T} = \{x_1^{b_1}, \ldots, x_T^{-b_T}\} \]
and similarly for the ancestor indices. The factorization into a marginal and a conditional density, which can be done since \( \phi \) is a probability density function, is intended to reveal some of the structure of the extended target. First, the marginal density of the variables \( \{\theta, x_{1:T}^{b_1:T}, b_{1:T}\} \) is defined to be equal to the original target density \( \gamma_T(\theta, x_{1:T}^{b_1:T}) \), up to a factor \( N^{-T} \) (related to the index variables \( b_{1:T} \)). By defining the marginal in this way, we fulfill the first of the two requirements. More precisely, if \( \{\theta, x_{1:T}, a_{2:T}, k\} \) are distributed according to \( \phi \), then, by construction, the variables \( \{\theta, x_{1:T}, a_{2:T}, k\} \) are distributed according to \( \gamma_T \). Note that the particle trajectory \( x_{1:T}^{b_1:T} \) is the ancestral path of \( x_T^k \), as defined in (103).

Second, to define the conditional density of the remaining variables, we look at the second requirement. That is, we strive to make this conditional density as close as possible to the proposal density (107), or in effect as close as possible to \( \psi^\theta \) defined in (106). This is done by defining the conditional, i.e. the second factor in (108), similarly to (106). The difference is that, since we now condition on the path \( x_{1:T}^{b_1:T} \) and the indices \( b_{1:T} \), we remove the corresponding factors from the products.

With the target density (108) and the proposal (107) in place, we are close to having a complete Metropolis-Hastings sampler. What remains is to compute the acceptance probability. For a proposed move from \( \{\theta, x_{1:T}, a_{2:T}, k\} \) to \( \{\theta', x_{1:T}', a_{2:T}', k'\} \), this is given by a standard Metropolis-Hastings ratio,
\[ \frac{1 \wedge \phi(\theta', x_{1:T}', a_{2:T}', k') q(\theta \mid \theta') \psi^\theta(\theta, x_{1:T}, a_{2:T}) w_T^k}{\phi(\theta, x_{1:T}, a_{2:T}, k) q(\theta' \mid \theta) \psi^{\theta'}(\theta', x_{1:T}', a_{2:T}') w_T^{k'}}. \tag{109} \]
To simplify this expression we will rewrite (108) on an alternative form. Note first that we can write,
\[ \gamma_t^\theta(x_{1:t}) = \gamma_t^\theta(x_1) \prod_{s=2}^t \gamma_s^\theta(x_{1:s-1}). \tag{110} \]
By using the definition of the weight function (67), this expression can be expanded according to
\[ \gamma_t^\theta(x_{1:t}) = W_1^\theta(x_1) r_1^\theta(x_1) \prod_{s=2}^t W_s^\theta(x_{1:s}) r_s^\theta(x_{1:s-1}). \tag{111} \]
By plugging the trajectory \( x_{1:T}^{b_1:T} \) into the above expression, we get
\[ \gamma_t^\theta(x_{1:t}^{b_1:t}) = \bar{w}_1^{b_1} r_1^\theta(x_1^{b_1}) \prod_{s=2}^t \bar{w}_s^{b_s} r_s^\theta(x_s^{b_s}) | x_{1:s-1}^{b_{1:s-1}} \]
\[ = w_t^{b_t} \left( \prod_{s=1}^N \sum_{l=1}^N \bar{w}_s^l \right) r_1^\theta(x_1^{b_1}) \prod_{s=2}^t M_t^\theta(a_s^{b_s}, x_s^{b_s}), \tag{112} \]
Algorithm 14 Particle marginal Metropolis-Hastings (Andrieu et al., 2010)

1: Set $\theta[0]$ arbitrarily.
2: Run an SMC sampler, targeting $\gamma^0_T(x_{1:T})$, and compute an estimate of the normalization constant, $\hat{Z}^N_T(\theta[0])$.
3: Sample $k$ with $\mathbb{P}(k = i) = w^i_T$ and set $x_{1:T}[0] = x^k_{1:T}$.
4: for $r \geq 1$ do
5: Draw $\theta' \sim q(\theta' \mid \theta[r - 1])$.
6: Run an SMC sampler, targeting $\gamma^\theta_T(x_{1:T})$, and compute an estimate of the normalization constant, $\hat{Z}^N_T(\theta')$.
7: Sample $k$ with $\mathbb{P}(k = i) = w^i_T$ and set $x'_{1:T} = x^k_{1:T}$.
8: With probability
   \[
   1 \wedge \frac{\hat{Z}^N_T(\theta')}{{\hat{Z}^N_T(\theta'[r - 1])}} \frac{\pi(\theta')}{\pi(\theta'[r - 1])} \frac{q(\theta[r - 1] \mid \theta')}{q(\theta' \mid \theta[r - 1])},
   \]
   set $\{\theta[r], x_{1:T}[r], \hat{Z}^N_T(\theta'[r])\} = \{\theta', x'_{1:T}, \hat{Z}^N_T(\theta')\}$, otherwise set $\{\theta[r], x_{1:T}[r], \hat{Z}^N_T(\theta[r])\} = \{\theta[r - 1], x_{1:T}[r - 1], \hat{Z}^N_T(\theta[r - 1])\}$.
9: end for

where $\{\bar{w}^i_t\}_{i=1}^N$ are the unnormalized importance weights at time $t$. Let
\[
\hat{Z}^N_t(\theta) \triangleq \prod_{s=1}^t \left( \frac{1}{N} \sum_{i=1}^N \bar{w}^i_s \right).
\tag{113}
\]

Note that this quantity depends on all the random variables generated by the SMC sampler, $\{x_{1:t}, a_{2:t}\}$, though this dependence is not explicit in the notation. Using (113) we can now rewrite the extended target density (108) as,
\[
\phi(\theta, x_{1:T}, a_{2:T}, k) \overset{(99)}{=} \frac{\tilde{\gamma}^\theta_T(x_{1:T}) \pi(\theta)}{\hat{Z}^N_{T} \psi^\theta_T(x_{1:T}, a_{2:T})} \frac{\psi^\theta_T(x_{1:T}, a_{2:T})}{\psi^\theta_T(x_{1:T}, a_{2:T})} \frac{w^k_{T} \psi^\theta_T(x_{1:T}, a_{2:T})}{\hat{Z}^N_{T}},
\tag{112}
\]

where we have used the identity $b_T = k$. The random variable $\hat{Z}^N_T(\theta)$ has a natural interpretation as an estimator of the normalization constant $Z_T(\theta)$ in (100). It is well known that this estimator is unbiased (Pitt et al., 2012; Del Moral, 2004), which in fact follows directly from (114) since $\phi$ is a probability density function and thus integrates to one. By plugging (114) into (109), we obtain the, surprisingly simple, expression for the acceptance probability,
\[
1 \wedge \frac{\hat{Z}^N_T(\theta')}{{\hat{Z}^N_T(\theta)}} \frac{\pi(\theta')}{\pi(\theta)} \frac{q(\theta \mid \theta')}{{q(\theta' \mid \theta)}},
\tag{115}
\]

The resulting Metropolis-Hastings sampler, using the SMC-based proposal kernel (107) for the extended target density (108), is referred to as particle marginal Metropolis-Hastings (PMMH). It should be noted that, despite the rather cumbersome derivation, the method is very simple to implement. We summarize the PMMH sampler in Algorithm 14.
From the derivation above, it follows that the PMMH sampler generates a sequence \( \{ \theta[r], x_{1:T}[r] \}_{r \geq 0} \) with stationary distribution \( \gamma_T(\theta, x_{1:T}) \). Under additional weak assumptions, it also follows that the underlying Markov chain is ergodic, and PMMH is thus a valid MCMC sampler.

(A3) Let \( S = \{ \theta \in \Theta : \pi(\theta) > 0 \} \). Then, for any \( \theta \in S \) and any \( t \in \{1, \ldots, T\} \), \( S^\theta_t \subseteq Q^\theta_t \) where,

\[
S^\theta_t = \{ x_{1:t} \in X^t : \gamma^\theta_t(x_{1:t}) > 0 \}, \\
Q^\theta_t = \{ x_{1:t} \in X^t : r^\theta_t(x_{t} \mid x_{1:t-1}) \gamma^\theta_{t-1}(x_{1:t-1}) > 0 \}.
\]

(A4) The ideal Metropolis-Hastings sampler with target density \( \gamma_T(\theta, x_{1:T}) \) and proposal density \( (101) \) is irreducible and aperiodic.

Theorem 4. Assume (A3) and (A4). Then, for any \( N \geq 1 \), the PMMH sampler generates a sequence \( \{ \theta[r], x_{1:T}[r] \}_{r \geq 0} \) whose marginal distributions \( L^N(\{ \theta[r], x_{1:T}[r] \} \in \cdot) \) satisfy,

\[
\| L^N(\{ \theta[r], x_{1:T}[r] \} \in \cdot) - \gamma_T(\cdot) \|_{TV} \to 0 \quad \text{as } r \to \infty,
\]

where \( \| \cdot \|_{TV} \) is the total variation norm.

Proof: See (Andrieu et al., 2010, Theorem 4).

See Andrieu and Vihola (2012); Andrieu and Roberts (2009) for additional and more precise convergence results related to the PMMH method.

An interesting, and aesthetically appealing, property of PMMH is that it is reminiscent of an ideal Metropolis-Hastings sampler, targeting the marginal density

\[
\gamma_T(\theta) \triangleq \int \gamma_T(\theta, x_{1:T}) dx_{1:T},
\]

using the proposal kernel \( q(\theta' \mid \theta) \) (or, equivalently, using the ideal proposal \( (101) \) to target the joint density \( \gamma_T(\theta, x_{1:T}) \)). For this marginal Metropolis-Hastings sampler, the acceptance probability is given by,

\[
1 \wedge \frac{Z_T(\theta') \pi(\theta') q(\theta' \mid \theta')}{Z_T(\theta) \pi(\theta) q(\theta' \mid \theta)}.
\]

The difference between this expression and the PMMH acceptance probability \( (115) \) is that, in the latter, the unknown normalization constants are replaced by their SMC-based estimators \( (113) \).

PMMH can thus be thought of as an SMC approximation of an ideal marginal Metropolis-Hastings sampler. Since the estimator \( (113) \) is consistent, the PMMH acceptance probability converges to \( (117) \) as the number of particles \( N \) increase. That is, the convergence speed of PMMH converges to that of the ideal sampler. We emphasize that PMMH is exact for any number of particles, in the sense that the stationary distribution of the sampler is \( \gamma_T(\theta, x_{1:T}) \) for any \( N \geq 1 \) (see Theorem 4). However, for small \( N \), the variance of the estimator \( (113) \) will be large and PMMH tends to get stuck, leading to slow convergence.
For a fixed computational time, there is a trade-off between taking $N$ large to get a high acceptance probability, and to run many iterations of the MCMC sampler. This trade-off has been analyzed by Doucet et al. (2012); Pitt et al. (2012) who, under certain assumption, conclude that it is optimal to choose $N$ so that the variance of the logarithm of $\hat{Z}_T^N$ is around 1. As a rule of thumb, $N$ should thus scale at least linearly with $T$ to keep the variance of the normalization constant estimate in check (Andrieu et al., 2010).

Another thing that is interesting to note is that the variable $k$, which is sampled at line 7 of Algorithm 14, does not affect the accept/reject decision on line 8. That is, the acceptance probability depends only on the estimate of the normalization constant (113), and not on the specific particle trajectory $x_1^T$ that is extracted at line 7. Hence, if the object of interest is the marginal density (116), there is no need to sample $k$ at all.

### 5.3 PMMH with backward simulation

The PMMH targets the joint density $\gamma_T(\theta, x_{1:T})$. However, as pointed out in the previous section, if the focus is only on the marginal density (116), then the state trajectory $x_1^T$ does not have to be sampled.

On the contrary, if the focus is on the latent variables $x_{1:T}$, then it seems like a waste to generate $N$ particle trajectories at each iteration of the PMMH sampler, but keep only a single one, as is done in Algorithm 14. In fact, since the acceptance probability does not depend on the specific trajectory that is extracted, it is possible to average over all the trajectories instead of randomly picking one of them.

Assume that we wish to compute

$$E_{\gamma_T} [\varphi(\theta, x_{1:T})] = \int \varphi(\theta, x_{1:T}) \gamma_T(\theta, x_{1:T}) d\theta dx_{1:T},$$

(118)

for some test function $\varphi : \Theta \times X^T \rightarrow \mathbb{R}$. We run $R$ iterations of Algorithm 14 (possibly with some burn-in), resulting in a realisation of the process, $\{\theta[r], x_{1:T}[r]\}_{r=0}^R$. The most straightforward estimator of (118) is then

$$\hat{\varphi}_{PMMH} = \frac{1}{R} \sum_{r=0}^{R} \varphi(\theta[r], x_{1:T}[r]),$$

(119)

which, by the ergodic theorem, converges almost surely to (118). However, it was recognized by Andrieu et al. (2010) that an alternative is to use the Rao-Blackwellized estimator

$$\hat{\varphi}_{PMMH-RB} = \frac{1}{R} \sum_{r=0}^{R} \mathbb{E} \left[ \varphi(\theta[r], x_{1:T}^k[r]) \mid \theta[r], x_{1:T}[r], a_{2:T}[r] \right]$$

$$= \frac{1}{R} \sum_{r=0}^{R} \sum_{i=1}^{N} w_T^i[r] \varphi(\theta[r], x_{1:T}^i[r]).$$

(120)

The possibility to make use of all the generated particles to reduce the variance of the estimator seems promising. However, a problem with the above estimator is that the particle systems $\{x_{1:T}^i[r], w_T^i[r]\}_{i=1}^N$ are generated by SMC samplers, and will thus suffer from path degeneracy. Hence, the possible benefit of Rao-Blackwellization is limited, due
Algorithm 15 PMMH with backward simulation (Olsson and Rydén, 2011)

1: Set $\theta[0]$ arbitrarily.
2: Run an SMC sampler, targeting $\gamma_{T}^{\theta[0]}(x_{1:T})$, and compute an estimate of the normalization constant, $\hat{Z}_{T}^{N}(\theta[0])$.
3: Generate $\{\tilde{x}_{j1:T[0]}^{r}\}_{j=1}^{M}$ by backward simulation (Algorithm 12).
4: for $r \geq 1$ do
   5: Draw $\theta' \sim q(\theta' | \theta[r-1])$.
   6: Run an SMC sampler, targeting $\gamma_{T}^{\theta'}(x_{1:T})$, and compute an estimate of the normalization constant, $\hat{Z}_{T}^{N}(\theta')$.
   7: With probability \[ 1 \land \frac{\hat{Z}_{T}^{N}(\theta')}{\hat{Z}_{T}^{N}(\theta[r-1])} \frac{\pi(\theta')}{{\pi(\theta[r-1])}} \frac{q(\theta[r-1] | \theta')}{{q(\theta' | \theta[r-1])}}, \]
      set $I_{\text{accept}} = 1$, otherwise set $I_{\text{accept}} = 0$.
   8: if $I_{\text{accept}} = 1$ then
      9: Set $\{\theta[r], \hat{Z}_{T}^{N}(\theta[r])\} = \{\theta', \hat{Z}_{T}^{N}(\theta')\}$.
   10: Generate $\{\tilde{x}_{j1:T}[r]\}_{j=1}^{M}$ by backward simulation (Algorithm 12).
   11: else
   12: Set $\{\theta[r], \hat{Z}_{T}^{N}(\theta[r])\} = \{\theta[r-1], \hat{Z}_{T}^{N}(\theta[r-1])\}$.
   13: Set $\{\tilde{x}_{j1:T}[r]\}_{j=1}^{M} = \{\tilde{x}_{j1:T}[r-1]\}_{j=1}^{M}$.
   14: end if
   15: end for

To achieve a better variance reduction effect, Olsson and Rydén (2011) have proposed to complement PMMH with a run of a backward simulator. That is, line 7 of Algorithm 14 on page 115 is replaced by an execution of Algorithm 12 (page 98). In practice, it is preferable to run the backward simulator only if the proposed sample is accepted. This is possible since, as pointed out above, the acceptance probability is independent of the extracted trajectories. The PMMH sampler with backward simulation is given in Algorithm 15.

Let the $M$ backward trajectories generated at iteration $r$ of the PMMH sampler be denoted $\{\tilde{x}_{j1:T}[r]\}_{j=1}^{M}$. An estimator of (118) is then given by

$$\hat{\varphi}_{\text{PMMH-BS}} = \frac{1}{RM} \sum_{r=0}^{R} \sum_{j=1}^{M} \varphi(\theta[r], \tilde{x}_{j1:T}[r]).$$

(121)

It is shown by Olsson and Rydén (2011) that the backward simulator leaves the target distribution invariant. Again, by Rao-Blackwellization type of arguments, it then follows that (121) converges almost surely as $R \to \infty$, for any $M \geq 1$. 

To the low particle diversity for time points $t$ far away from the final time $T$. 


Olsson and Rydén (2011) provide an expression for the variance of the estimator (121),
\[
\text{Var} (\hat{\varphi}_{\text{PMMH-BS}}) = \mathbb{E} \left[ \text{Var} \left( \hat{\varphi}_{\text{PMMH-BS}} \mid \{\theta[r], x_{1:T}[r], a_{2:T}[r]\}_{r=0}^R \right) \right] + \text{Var} \left( \mathbb{E} \left[ \hat{\varphi}_{\text{PMMH-BS}} \mid \{\theta[r], x_{1:T}[r], a_{2:T}[r]\}_{r=0}^R \right] \right)
\]
\[
= \frac{1}{R} \left( \frac{\sigma^2}{M} + \sigma^2_R \right) \approx \frac{1}{R} \left( \frac{\sigma^2}{M} + \sigma^2_\infty \right),
\]
(122)
where
\[
\sigma^2 = \mathbb{E} \left[ \text{Var} \left( \varphi (\theta, \tilde{x}_{1:T}) \mid \{\theta, x_{1:T}, a_{2:T}\} \right) \right],
\]
\[
\sigma^2_R = \frac{1}{R} \text{Var} \left( \sum_{r=0}^R \mathbb{E} \left[ \varphi (\theta[r], \tilde{x}_{1:T}[r]) \mid \{\theta[r], x_{1:T}[r], a_{2:T}[r]\} \right] \right),
\]
and \(\sigma^2_\infty = \lim_{R \to \infty} \sigma^2_R\) is the time-average variance constant. This expression can be used to find an optimal trade off between \(R\) and \(M\), depending on the run times of the algorithm for different settings. In practice, the parameters \(\sigma^2\) and \(\sigma^2_\infty\) are not known, and to make use of (122) to tune the sampler it is thus necessary to estimate these parameters from data.

We illustrate the effects of Rao-Blackwellization and backward simulation in the example below.

--- Example 15: PIMH with backward simulation ---

We present a simulation study similar to one of the examples considered by Olsson and Rydén (2011). Consider again the nonlinear time-series model,
\[
x_{t+1} = 0.5x_t + 25 \frac{x_t}{1 + x_t^2} + 8 \cos(1.2t) + v_t,
\]
\[
y_t = 0.05x_t^2 + e_t,
\]
with \(v_t \sim \mathcal{N}(0, 10), e_t \sim \mathcal{N}(0, 1)\) and \(x_1 \sim \mathcal{N}(0, 5)\). We seek the joint smoothing distribution \(p(x_{1:T} \mid y_{1:T})\). For simplicity, we assume that there are no unknown parameters in the model. Since the PMCMC samplers in Algorithms 14 and 15 address the joint parameter and state inference problem, smoothing is covered as a special case (see Section 5.7). In this case, i.e. in the absence of an unknown static parameter, the PMMH sampler is referred to as particle independent Metropolis-Hastings (PIMH).

We wish to compare the estimator variances for the three alternative estimators \(\hat{\varphi}_{\text{PIMH}}, \hat{\varphi}_{\text{PIMH-RB}}\) and \(\hat{\varphi}_{\text{PIMH-BS}}\). We generate one batch of \(T = 100\) observations from the time series. We then run 100 independent copies of Algorithms 14 and 15 (all using the same data), each for \(R = 5000\) iterations. We use \(N = 500\) particles for the SMC samplers and \(M = 25\) backward trajectories in Algorithm 15. For each run, we compute three estimates of the posterior mean of \(x_{1:T}\) according to (119), (120) and (121), respectively.

Figure 15 shows the estimator variances for each time point \(t\). Close to the final time point \(T\), the Rao-Blackwellization in (120) has a clear impact on the estimator variance, which is reduced by about two orders of magnitude compared to the crude PIMH estimator. However, due to path degeneracy, the variance reduction gets less pronounced for times \(t\) far from \(T\). For \(t < 50\), there is more or less no reduction at all. For the backward-
simulation-based estimator (121), however, we obtain a clear variance reduction for all time points.

5.4 Particle Gibbs with backward simulation

PMMH is a Metropolis-Hastings sampler, targeting (108) with the specific proposal given in (107). However, now that we have the extended target (108) in place, we can think about other MCMC samplers targeting this distribution, leading to other members of the PMCMC family. One possibility is to design a multi-stage Gibbs sampler for \( \phi \), using the following sweep (note that \( b_1:T = \{a_{1:T}^{b_1:T}, b_{1:T}\} \)),

(i) Draw \( \theta' \sim \phi(\theta \mid x_{1:T}^{a_{1:T}^{b_1:T}}, b_{1:T}) \);

(ii) Draw \( \{x_{1:T}^{a_{1:T}^{b_1:T}}, a_{1:T}^{a_{2:T}^{b_2:T}}\} \sim \phi(x_{1:T}^{-a_{1:T}^{b_1:T}}, a_{2:T}^{-a_{2:T}^{b_2:T}} \mid \theta', x_{1:T}^{a_{1:T}^{b_1:T}}, b_{1:T}) \);

(iii) Draw \( k' \sim \phi(k \mid \theta', x_{1:T}^{-a_{1:T}^{b_1:T}}, a_{2:T}^{-a_{2:T}^{b_2:T}}, x_{1:T}^{a_{2:T}^{b_2:T}}, a_{2:T}^{b_{2:T}}) \);

This is the particle Gibbs (PG) sampler, proposed by Andrieu et al. (2010). The first step of the procedure is a partially collapsed Gibbs step (see Section 2.2.3 or Dyk and Park (2008); Liu (2001)), which leaves the target distribution invariant. Alternatively, we may view Steps (i)–(ii) as a grouped Gibbs step for the variables \( \{\theta, x_{1:T}^{-a_{1:T}^{b_1:T}}, a_{2:T}^{-a_{2:T}^{b_2:T}}\} \).

Later in this section, we will consider an alternative Gibbs sweep which makes use of backward simulation to improve the mixing of the sampler. However, to be able to see why backward simulation can be useful in this context, we first need to understand the properties of the PG sampler. To get a better picture of what this method does, let us therefore go through the three steps of the procedure and discuss how they can be implemented.

For Step (i) we have, by the construction of the extended target distribution \( \phi \) in (108),

\[
\phi(\theta \mid x_{1:T}^{a_{1:T}^{b_1:T}}, b_{1:T}) = \gamma_T(\theta \mid x_{1:T}) \triangleq \frac{\gamma_T(\theta, x_{1:T})}{\int \gamma_T(\theta, x_{1:T}) d\theta}.
\] (123)
As argued before, it is in general much easier to sample the parameter \( \theta \) conditionally on the latent variables \( x_{1:T} \), than from the marginal density (116). We shall thus assume that this step can be carried out exactly. This is possible when a conjugate prior is used for the latent variables \( x \). Hence, we can sample from (124) by using a procedure reminiscent of the SMC sampler, leaving (123) invariant.

The conditional density used in Step (ii) is also given by construction,

\[
\phi(x_{1:T}^{-b_{1:T}}, a_{2:T}^{-b_{2:T}} \mid \theta, x_{1:T}^{b_{1:T}}, b_{1:T}) = \prod_{i=1}^{N} r_{i}^{(1)}(x_{i}^{1}) \prod_{t=2}^{T} \prod_{i=1}^{N} M_{t}^{(\theta)}(a_{i}^{t}, x_{i}^{t}).
\] (124)

It is interesting to note that this expression does not depend explicitly on the target density \( \gamma_{T}(\theta, x_{1:T}) \), but only on the proposal kernels used in the SMC sampler (cf. (106)). Hence, we can sample from (124) by using a procedure reminiscent of the SMC sampler in Algorithm 11 on page 96. The difference is that in (124), the factors corresponding to the indices \( b_{1:T} \) have been excluded. Hence, to sample from (124) we first generate \( x_{1}^{1} \sim r_{1}^{(1)}(x_{1}) \) for \( i \in \{1, \ldots, N\} \setminus b_{1} \), then \( \{a_{2}^{1}, x_{2}^{1}\} \sim M_{2}^{(\theta)}(a_{2}, x_{2}) \) for \( i \in \{1, \ldots, N\} \setminus b_{2} \), etc. Throughout this sampling process, the trajectory \( x_{1:T}^{b_{1:T}} \) is kept fixed.

Before stating an algorithm corresponding to this sampling procedure, however, we note that the indices \( b_{1:T} \) are nuisance variables. That is, in practice we are not interested in the values of these variables and they are only introduced to aid in the derivation of the algorithm. Furthermore, the SMC sampler is invariant to permutations of the particle indices, i.e. it does not matter in which order we enumerate the particles. Consequently, the actual values of the indices \( b_{1:T} \) are of no significance when sampling from (124), meaning that we do not have to keep track of these variables. Instead we can fix \( b_{1:T} \) to some arbitrary sequence which we find more convenient, e.g. \( b_{1:T} = (N, \ldots, N) \). With this convention, a method for sampling from (124), referred to as a conditional SMC (CSMC) sampler, is given in Algorithm 16.

**Algorithm 16** Conditional SMC (conditioned on \( x_{1:T}^{b_{1:T}} \))

1: Draw \( x_{1}^{1} \sim r_{1}^{(1)}(x_{1}) \) for \( i = 1, \ldots, N - 1 \) and set \( x_{1}^{N} = x_{1}^{1} \).
2: Compute \( \bar{w}_{1}^{i} = W_{1}(x_{1}^{i}) \) for \( i = 1, \ldots, N \).
3: Normalize the weights \( w_{1}^{i} = \bar{w}_{1}^{i} / \sum_{i=1}^{N} \bar{w}_{1}^{i} \) for \( i = 1, \ldots, N \).
4: for \( t = 2 \) to \( T \) do
5: Draw \( \{a_{t}^{i}, x_{t}^{i}\} \sim M_{t}^{(\theta)}(a_{t}, x_{t}) \) for \( i = 1, \ldots, N - 1 \).
6: Set \( a_{N}^{i} = N \) and \( x_{N}^{i} = x_{t}^{i} \).
7: Set \( x_{1:t}^{i} = \{x_{1:t-1}^{a_{t}^{i}}, x_{t}^{i}\} \) for \( i = 1, \ldots, N \).
8: Compute \( \bar{w}_{t}^{i} = W_{t}(x_{1:t}) \) for \( i = 1, \ldots, N \).
9: Normalize the weights \( w_{t}^{i} = \bar{w}_{t}^{i} / \sum_{i=1}^{N} \bar{w}_{t}^{i} \) for \( i = 1, \ldots, N \).
10: end for
**Algorithm 17** Particle Gibbs (Andrieu et al., 2010)

1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily.
2: for $r \geq 1$ do
3: Draw $\theta[r] \sim \gamma_T(\theta \mid x_{1:T}[r-1])$.
4: Run a CSMC sampler (Algorithm 16) targeting $\gamma^\theta_T(x_{1:T})$, conditioned on $x_{1:T}[r-1]$.
5: Sample $k$ with $P(k = i) = w_i^T$ and trace the ancestral path of particle $x_{1:T}^k$, i.e. set $x_{1:T}[r] = x_{1:T}^k$.
6: end for

Finally, for Step (iii), we note that

$$\phi(k \mid \theta, x_{1:T}, a_{2:T}) \propto \phi(\theta, x_{1:T}, a_{2:T}, k) \propto w_k^T.$$  

Hence, as in PMMH, we sample the index $k$ with $P(k = i) = w_i^T$. We can now reinterpret the three steps of the PG sampler as in Algorithm 17.

As can be seen in Theorem 5 below, ergodicity of PG holds under similar assumptions as for PMMH. See also Chopin and Singh (2013) for a uniform ergodicity result for the PG sampler.

**(A5)** The ideal Gibbs sampler, defined by alternating between sampling from $\gamma_T(\theta \mid x_{1:T})$ and $\gamma^\theta_T(x_{1:T})$, is irreducible and aperiodic.

**Theorem 5.** Assume (A3) and (A5). Then, for any $N \geq 2$, the PG sampler generates a sequence $\{\theta[r], x_{1:T}[r]\}_{r \geq 0}$ whose marginal distributions $L^N(\{\theta[r], x_{1:T}[r]\} \in \cdot)$ satisfy

$$\|L^N(\{\theta[r], x_{1:T}[r]\} \in \cdot) - \gamma_T(\cdot)\|_{TV} \to 0 \quad \text{as } r \to \infty,$$

where $\|\cdot\|_{TV}$ is the total variation norm.

**Proof:** See (Andrieu et al., 2010, Theorem 5).

As pointed out above, a key property of PMCMC methods is that they do not rely on asymptotics in $N$ to be valid MCMC samplers. However, for the PMMH sampler we found that the acceptance probability (115) depends on the SMC estimate of the normalization constant (113). Hence, to obtain a reasonable probability of acceptance, we have to take $N$ large enough to get sufficiently small variance of this estimator.

For the PG sampler, on the other hand, the dependence on $N$ is not as obvious. Since it is a Gibbs sampler, all generated samples will in fact be accepted. To investigate how the method is affected by different values of $N$, we apply it to a simple toy problem in the example below.
**Example 16: PG for stochastic volatility model**

Consider a simple stochastic volatility model on state-space form,

\[ x_{t+1} = ax_t + v_t, \quad v_t \sim \mathcal{N}(0, \theta), \]

\[ y_t = e_t \exp\left(\frac{1}{2}x_t\right), \quad e_t \sim \mathcal{N}(0, 1). \]

For brevity, we keep \( a = 0.9 \) fixed, but assume that the variance \( \theta \) of the latent process is unknown. We put a conjugate inverse gamma prior on \( \theta \), with hyperparameters \( a = b = 0.01 \). We then generate a batch of \( T = 100 \) measurements with the true value \( \theta = 0.5^2 \), and employ the PG sampler to find the posterior \( p(\theta \mid y_{1:T}) \) of the parameter.

To see how the sampler is affected by the number of particles, we consider four independent runs with \( N = 5, 20, 100 \) and \( 1000 \), respectively. We run the samplers for 100 000 iterations, discarding the first 10 000 iterations as burn-in. The empirical ACFs for the residuals \( \theta[r] - \mathbb{E}[\theta \mid y_{1:T}] \) are reported in the left panel of Figure 16. For \( N = 1000 \), we get a fairly sharp drop in autocorrelation, indicating a sampler that mixes well. However, as we decrease the number of particles, there is a dramatic change in the ACF. For \( N = 20 \), the ACF drops off very slowly and for \( N = 5 \) it is almost constant, meaning that the sampler is more or less stuck at a single point in the parameter space for all 100 000 iterations.

Furthermore, to see how the sampler is affected by the size of the data set, we repeat the experiment with \( T = 1000 \). The ACFs are given in the right panel of Figure 16. The same effect is clearly visible and even more severe in this scenario. Even for \( N = 1000 \) the method struggles, and as we reduce the number of particles further, the sampler gets stuck.

To get a better insight into the poor mixing of the PG kernel for small \( N \) and/or large \( T \), let us analyze two consecutive iterations of the sampler in more detail. For clarity, we use a small number of particles and time steps, \( N = 20 \) and \( T = 50 \), respectively.
Figure 17 (top) shows the particle system generated by the CSMC sampler at iteration \( r \). Due to path degeneracy, there is only one distinct particle trajectory for \( t \leq 27 \). The extracted trajectory \( x_{1:T}[r] \), corresponding to the ancestral path of particle \( x^*_k \), is illustrated by a thick black line. At the next iteration of the PG sampler we run CSMC conditioned on \( x_{1:T}[r] \). This results in the system shown in Figure 17 (bottom). Due to path degeneracy, we once again obtain only a single distinct particle trajectory for time points far from \( T \), here for \( t \leq 35 \). However, the particle system generated by the CSMC sampler must contain the conditioned path. Hence, all particle trajectories available at iteration \( r + 1 \) are identical to \( x_{1:T}[r] \) up to time \( t = 27 \). Consequently, when we sample \( x_{1:T}[r + 1] \) at iteration \( r + 1 \), this trajectory will to a large extent be identical to \( x_{1:T}[r] \). This results in a poor exploration of the state-space, which in turn means that the Gibbs kernel will mix slowly.

Based on this argument we note that the mixing will be particularly poor when the path degeneracy is severe. This will be the case if the length of the data record \( T \) is large and/or if the number of particles \( N \) is small. This is in agreement with the results reported in Figure 16.

From the discussion in the example above we conclude that the PG sampler suffers from poor mixing as a result of path degeneracy. Hence, for this method to work, we need to take \( N \) large enough to tackle the degeneracy, which for many problems is unrealistic from a computational point of view.

An alternative interpretation of this issue is that it is caused by the fact that the Gibbs sweep defining the PG sampler, Steps (i)–(iii) above, is incomplete. More precisely, if we collect the variables from the three steps of the procedure, we note that we never sample new values for \( \{x_{1:T}^{b_1:T}, b_{1:T-1}\} \) in this sweep. Despite this incompleteness of the Gibbs sweep, it holds under the assumptions of Theorem 5 that PG is ergodic. Intuitively, this can be explained by the fact that the collection of variables that is left out is chosen randomly at each iteration. In the long run, we will thus include all the variables with probability one. However, due to degeneracy of the SMC sampler, the collections of variables that are left out at any two consecutive iterations will be strongly dependent, resulting in poor mixing.

A way to address this problem was proposed by Whiteley (2010) and further explored by Whiteley et al. (2010); Lindsten and Schön (2012). The idea is to complement the PG sampler with a backward simulator to mitigate the path degeneracy problem, and thus obtain a faster mixing sampler. Alternatively, based on the interpretation above, this modification can be seen as a way to include the index variables \( b_{1:T-1} \) in the Gibbs sampler\(^4\). As noted above, we are not interested in these index variables per se, but including them in the Gibbs sweep can lead to a general improvement in mixing, affecting all the variables of the model. The idea is to sample the variables \( b_t \) backward in time, using a sequence of Gibbs steps. More precisely, we replace Step (iii) of the PG sampler with the following,

---

\(^4\) Ideally, we would like to include the variables \( x_{1:T}^{b_{1:T}} \) as well, but this is in general not possible since it would be similar to sampling from the original target density, which we assume is infeasible.
Figure 17: Particle system generated by CSMC at iterations $r$ (top) and $r + 1$ (bottom) of the PG sampler. The dots show the particle positions, the thin black lines show the ancestral dependence of the particles and the thick black lines show the sampled trajectories $x_{1:T}[r]$ and $x_{1:T}[r + 1]$, respectively. In the bottom panel, the thick gray line illustrates the conditioned path at iteration $r + 1$, which by construction equals $x_{1:T}[r]$. Note that, due to path degeneracy, the particles shown as grey dots are not reachable by tracing any of the ancestral lineages from time $T$ and back.
(iii)' Draw,

\[ b'_T \sim \phi(b_T \mid \theta', x_{1:T}^{t,-b_1:T}, a_{2:T}^{t,-b_2:T}, x_{b_1:T}^{1:T}, b_{b_2:T}) ; \]

\[ \vdots \]

\[ b'_t \sim \phi(b_t \mid \theta', x_{1:t}^{t,-b_1:t}, a_{2:t}^{t,-b_2:t}, x_{b_1:t}^{1:t}, b_{b_2:t}^{t+1:T}, b_{t+1:T}) ; \]

\[ \vdots \]

\[ b'_1 \sim \phi(b_1 \mid \theta', x_{1}^{1,-b_1}, x_{b_1}^{1}, b_{b_2:T}, b_{2:T}). \]

Note that the densities involved in this sampling scheme are conditionals, not under the full joint density \( \phi(\theta, x_{1:T}, a_{2:T}, k) \), but under marginals thereof. That is, Step (iii)' corresponds to a sequence of partially collapsed Gibbs steps. As we have noted before, collapsing, or marginalization, is often used within Gibbs sampling and it does not violate the invariance of the sampler.

As hinted at above, this backward sampling of the ancestor indices corresponds to a run of a backward simulator. We call the resulting method, defined by the Gibbs Steps (i), (ii) and (iii)', particle Gibbs with backward simulation (PGBS). Note that the first row of Step (iii)' is identical to Step (iii) of the original PG sampler, where we sample the variable \( k = b_T \) from its full conditional. However, in PGBS we do not stop there, but continue to draw new values for the indices \( b_t \) down to \( t = 1 \). This will break the strong dependencies caused by the SMC path degeneracy, and allow for a much faster mixing Gibbs kernel.

To see that Step (iii)' indeed corresponds to a run of a backward simulator, note first that by marginalizing (124) over \( \{x_{t+1:T}^{b_1:T}, a_{t+1:T}^{b_2:T}\} \) we get

\[ \phi(x_{1:t}^{b_1:t}, a_{2:t}^{b_2:t} \mid \theta, x_{1:T}^{1:b_1:T}, b_1:T) = \prod_{i=1}^{N} r_{2}^{\theta}(x_{1}^{i}) \prod_{s=2}^{N} \prod_{i=1}^{N} M_{s}^{\theta}(a_{s}^{i}, x_{s}^{i}). \]  

(126)

We can thus write

\[ \phi(b_t \mid \theta, x_{1:t}, a_{2:t}, x_{t+1:T}^{b_1:t}, b_{t+1:T}) \propto \phi(\theta, x_{1:t}, a_{2:t}, x_{t+1:T}^{b_1:t}, b_{t+1:T}) \]

\[ = \phi(\theta, x_{1:t}^{b_1:t}, b_{1:T}) \phi(x_{1:t}^{b_1:t}, a_{2:t}^{b_2:t} \mid \theta, x_{1:t}^{b_1:t}, b_{1:T}) \]

\[ = \frac{\tilde{\gamma}_{T}(x_{1:t}^{b_1:t})}{\tilde{\gamma}_{t}(x_{1:t}^{b_1:t})} \frac{\tilde{\gamma}_{t}(x_{1:t}^{b_1:t})}{\pi(\theta)} \prod_{i=1}^{N} \prod_{s=1}^{N} \prod_{i=1}^{N} M_{s}^{\theta}(a_{s}^{i}, x_{s}^{i}), \]  

(127)

where we have used (99) and (126) for the last equality. By expanding \( \tilde{\gamma}_{T}(x_{1:t}^{b_1:t}) \) in the numerator according to (112), we obtain,

\[ \phi(b_t \mid \theta, x_{1:t}, a_{2:t}, x_{t+1:T}^{b_1:t}, b_{t+1:T}) \propto w_{b_t}^{t} \frac{\tilde{\gamma}_{T}(x_{1:t}^{b_1:t})}{\tilde{\gamma}_{t}(x_{1:t}^{b_1:t})}, \]  

(128)

which is exactly the expression for the backward simulation weights (72). We summarize the PGBS sampler in Algorithm 18. On line 5 of the algorithm, the indices \( b_{1:T} \) are
Algorithm 18 Particle Gibbs with backward simulation (Whiteley et al., 2010; Lindsten and Schön, 2012)

1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily.
2: for $r \geq 1$ do
3: Draw $\theta[r] \sim \gamma_T(\theta \mid x_{1:T}[r - 1])$.
4: Run a CSMC sampler (Algorithm 16) targeting $\gamma_T^{\theta[r]}(x_{1:T})$, conditioned on $x_{1:T}[r - 1]$.
5: Run a backward simulator (Algorithm 12, page 98, with $M = 1$) to sample the trajectory $x_{1:T}[r]$.
6: end for

generated according to Step (iii)' of the sampler. However, as noted above, we are not interested in these indices themselves, but only in the trajectory $x_{1:T}[r]$. Consequently, for practical convenience, the algorithm makes no explicit reference to $b_{1:T}$. Note that we only generate a single backward trajectory at each iteration of the sampler. Hence, the computational complexity is still linear in the number of particles. The effect of adding backward simulation to the sampler is quite substantial, which is illustrated on our toy problem in the example below.

---

Example 17: PGBS for stochastic volatility model

We return to Example 16 and apply the PGBS sampler to the same batches of data, for $T = 100$ and $T = 1000$, respectively. As before, we consider four independent runs with $N = 5, 20, 100$ and $1000$, respectively. The empirical ACFs for the residuals $\theta[r] - \mathbb{E}[\theta \mid y_{1:T}]$, based on 100 000 iterations of the samplers, are given in Figure 18.

When compared with the corresponding plots for the PG sampler in Figure 16, we see a large improvement. The addition of a backward simulation sweep appears to make the method much more robust to a small number of particles, as well as larger data sets. In fact, there is no noticeable improvement when increasing the number of particles above $N = 20$. Hence, for this specific example, we can conclude that PGBS with $N \gtrsim 20$ performs very close to an ideal Gibbs sampler, i.e. a Gibbs sampler in which $x_{1:T}$ is sampled from the exact JSD.

To provide further insight into the effect of the backward simulation step in PGBS, we make a similar inspection as in Example 16. That is, we look at two consecutive iterations of the sampler in more detail, using $N = 20$ and $T = 50$ for clarity.

Figure 19 (top) shows the particles generated by the CSMC sampler at iteration $r$, as well as the specific trajectory $x_{1:T}[r]$ which is sampled by the backward simulator. At the next iteration of the PGBS sampler we run CSMC conditioned on $x_{1:T}[r]$. This results in the particles shown in Figure 19 (bottom). The conditioned path is illustrated by a thick gray line, and the path $x_{1:T}[r + 1]$ is shown as a black line.

As discussed in Example 16, the CSMC sampler will degenerate toward the conditioned path. However, since $x_{1:T}[r + 1]$ is sampled using a backward simulator, we are not constrained to any of the ancestral lineages. Instead, we explore all the particles generated in the forward sweep, and not only those that survive to time $T$. This results in a path
Figure 18: Empirical ACF for $\theta$ for the PGBS samplers using different numbers of particles $N$.

$x_{1:T}^{\tau + 1}$ which, with high probability, is to a large extent different from $x_{1:T}^{\tau}$. The effect is that PGBS explores the state-space around the conditioned path much better than PG, resulting in a faster mixing Gibbs kernel.

In the example above, PGBS appeared to be quite robust to both large $T$ and small $N$, performing close to an ideal Gibbs sampler even with very few particles. This has been experienced also in more challenging scenarios, see e.g. Whiteley et al. (2010) for inference in jump Markov linear systems and Whiteley et al. (2011) for multiple change-point problems.

Intuitively, there are two reasons for why PGBS can perform well, even for small $N$. First, the conditioning in the CSMC sampler is crucial. It can be thought of as guiding the particles toward the conditioned path. In that way we retain information from one MCMC iteration to the next. In this aspect, PMMH, for instance, is more blunt, since it is based on independent SMC samplers at each iteration. Second, the use of backward simulation to update the indices $b_{1:T}$ allows for a perturbation around the conditioned path. Due to this, the sampler can avoid getting stuck as a result of path degeneracy, and therefore explores the state-space efficiently around the conditioned path.

PGBS is reminiscent of the Gibbs sampler proposed by Neal et al. (2004) for inference in SSMs. In this method, a pool of candidate states is generated, followed by a backward trajectory simulation. From this perspective, PGBS can be seen as a clever way of generating the candidate states by running an SMC sampler.

5.5 Particle Gibbs with ancestor sampling

The CSMC sampler that is used in PG and PGBS is a sequential method, progressing forward in time, generating the variables $\{x_t^{b_t}, a_t^{b_t}\}$ for $t = 1, \ldots, T$. Similarly to the backward simulation pass, this forward sweep can be interpreted as a sequence of
Figure 19: Particles generated by CSMC at iterations $r$ (top) and $r + 1$ (bottom) of the PGBS sampler. The dots show the particle positions and the black lines show the trajectories $x_{1:T}[r]$ and $x_{1:T}[r + 1]$, respectively, sampled by backward simulation. In the bottom panel, the thick gray line illustrates the conditioned path at iteration $r + 1$, which by construction equals $x_{1:T}[r]$. Note that all the particles, at all time points, are reachable by the backward simulators.
As for PG and PGBS, it can be verified that the above procedure is a partially collapsed Gibbs steps. To see this, we note that (126) implies that

$$\phi(x_1^{-b_1} \mid \theta, x_{1:1:T}^{b_1:T}, b_{1:T}) = \prod_{i=1}^{N} r_i^\theta(x_i^i),$$

(129a)

and, for \( t = 2, \ldots, T, \)

$$\phi(x_t^{-b_t}, a_t^{-b_t} \mid \theta, x_{1:t-1}^{b_{1:t-1}}, a_{2:t-1}^{-b_{2:t-1}}, x_{1:T}^{b_1:T}, b_{1:T})$$

$$= \frac{\phi(x_{1:t-1}^{b_{1:t-1}}, a_{2:t-1}^{-b_{2:t-1}} \mid \theta, x_{1:T}^{b_1:T}, b_{1:T})}{\phi(x_{1:t-1}^{-b_{1:t-1}}, a_{2:t-1}^{b_{2:t-1}} \mid \theta, x_{1:T}^{b_1:T}, b_{1:T})} = \prod_{i=1}^{N} M_t^\theta(a_i^i, x_i^i).$$

(129b)

That is, we can interpret the CSMC sampler in Algorithm 16 as first sampling \( x_1^{-b_1} \) from (129a) and then, for \( t = 2, \ldots, T, \) sampling \( \{x_t^{-b_t}, a_t^{-b_t}\} \) from (129b). This corresponds to a sequence of partially collapsed Gibbs steps.

In PGBS, this forward sweep is complemented with a backward sweep, in which the variables \( b_t \) are generated from the conditionals (128) for \( t = T, T - 1, \ldots, 1. \) However, it is possible to instead alternate between these two sequences and generate all the variables in a single forward sweep. That is, after an initial draw from (129a), we alternate between sampling \( \{x_t^{-b_t}, a_t^{-b_t}\} \) from (129b) and \( b_{t-1} \) from (128). Finally, we sample \( b_T \) from its full conditional (125), analogously to the PG sampler. This leads to a related method proposed by Lindsten et al. (2012), referred to as particle Gibbs with ancestor sampling (PGAS).

In summary, PGAS is a Gibbs sampler for the extended target distribution \( \phi \) defined in (108), using the following sweep;

1. Draw \( \theta' \sim \phi(\theta \mid x_1^{b_1:T}, b_{1:T}); \)
2. Draw \( x_1^{t-1} \sim \phi(x_1^{-b_1} \mid \theta, x_{1:1:T}^{b_1:T}, b_{1:T}) \) and, for \( t = 2, \ldots, T, \)
3. Draw \( b_t \sim \phi(b_t \mid \theta', x_1^{t-1}^{-b_1:t-1}, a_{2:t-1}^{-b_{2:t-1}}, x_{1:T}^{b_1:T}, b_{t-1:T}); \)
4. Draw \( (a_t^{-b_t}) = b_{t-1} \sim \phi(b_{t-1} \mid \theta', x_1^{t-1}^{-b_1:t-1}, a_{2:t-1}^{-b_{2:t-1}}, x_{1:T}^{b_1:T}, b_{t-1:T}); \)

Hence, at each iteration of the CSMC sampler, we draw a new value for the ancestor index \( a_t^{b_t}, \) instead of setting this according to the conditioning. The resulting method, denoted CSMC with ancestor sampling, is given in Algorithm 19. Again, for convenience we put the conditioned particles in the \( N \)th positions, i.e. we set \( b_{1:T} = (N, \ldots, N). \) The PGAS sampler is summarized in Algorithm 20.

As for PG and PGBS, it can be verified that the above procedure is a partially collapsed Gibbs sampler and it will thus leave \( \phi \) invariant. Furthermore, the ergodicity result from Theorem 5 applies without modification to PGAS. In practice, the ancestor sampling in the CSMC procedure gives rise to a considerable improvement over PG, comparable to

\(^5\)As pointed out above, Algorithm 16 fixates \( b_{1:T} = (N, \ldots, N) \) for the conditioned path.
We return again to the toy problem studied in Examples 16 and 17. We now apply the Algorithm 20 Particle Gibbs with ancestor sampling (Lindsten et al., 2012)

CSMC with ancestor sampling (conditioned on Algorithm 19) will degenerate. The difference from the basic CSMC procedure gives the same effect. In fact, the paths generated by the CSMC procedure with ancestor sampling used in PGAS can be used to mitigate this problem, which explains why PGBS can outperform the basic PG sampler. However, it might be harder to see why the ancestor sampling used in PGAS gives the same effect. In fact, the paths generated by the CSMC procedure with ancestor sampling (Algorithm 19) will degenerate. The difference from the basic CSMC procedure

5 Backward simulation in particle MCMC

Algorithm 19 CSMC with ancestor sampling (conditioned on $x_{1:T}^r$)

1: Draw $x_1^i \sim r_1^\theta(x_1)$ for $i = 1, \ldots, N - 1$ and set $x_1^N = x_1^i$.
2: Compute $\bar{w}_1^i = W_1(x_1^i)$ for $i = 1, \ldots, N$.
3: Normalize the weights $w_1^i = \bar{w}_1^i / \sum_{i=1}^N \bar{w}_1^i$ for $i = 1, \ldots, N$.
4: for $t = 2$ to $T$ do
5: Draw $\{a_t^i, x_t^i\} \sim M_t^\theta(a_t, x_t)$ for $i = 1, \ldots, N - 1$ and set $x_T^N = x_T^i$.
6: Draw $a_t^N$ with

$$P(a_t^N = i) \propto \frac{\gamma_t^\theta(\{x_t^i, x_{i:T}^r\})}{\bar{\gamma}_{t-1}^\theta(x_{i:t-1}^i)}.$$

7: Set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$ for $i = 1, \ldots, N$.
8: Compute $\bar{w}_t^i = W_t^\theta(x_{1:t}^i)$ for $i = 1, \ldots, N$.
9: Normalize the weights $w_t^i = \bar{w}_t^i / \sum_{i=1}^N \bar{w}_t^i$ for $i = 1, \ldots, N$.
10: end for

Algorithm 20 Particle Gibbs with ancestor sampling (Lindsten et al., 2012)

1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily.
2: for $r \geq 1$ do
3: Draw $\theta[r] \sim \gamma_T(\theta \mid x_{1:T}[r-1])$.
4: Run CSMC with ancestor sampling (Algorithm 19) targeting $\gamma_{T[r]}^\theta(x_{1:T})$, conditioned on $x_{1:T}[r-1]$.
5: Sample $k$ with $P(k = i) = w_T^i$, and trace the ancestral path of particle $x_T^k$, i.e. set $x_{1:T}[r] = x_T^k$.
6: end for

that of backward simulation. The method has been successfully applied to challenging inference problems, such as Wiener system identification (Lindsten et al., 2013b) and learning of nonparametric, nonlinear SSMs (Frigola et al., 2013). We illustrate the PGAS method in the following example.

Example 18: PGAS for stochastic volatility model

We return again to the toy problem studied in Examples 16 and 17. We now apply the PGAS sampler to the same batches of data, for $T = 100$ and $T = 1000$, respectively. As before, we consider four independent runs with $N = 5, 20, 100$ and $1000$, respectively. The empirical ACFs for the residuals $\hat{\theta}[r] - E[\theta \mid y_{1:T}]$, based on 100 000 iterations of the samplers, are given in Figure 20.

The results are similar to those obtained by PGBS; see Figure 18. As for PGBS, in this specific example, the PGAS sampler performs very close to an ideal Gibbs sampler for $N \gtrsim 20$. As was discussed in the previous section, the source of the poor mixing of the basic PG sampler is path degeneracy. We have previously seen how backward simulation can be used to mitigate this problem, which explains why PGBS can outperform the basic PG sampler. However, it might be harder to see why the ancestor sampling used in PGAS gives the same effect. In fact, the paths generated by the CSMC procedure with ancestor sampling (Algorithm 19) will degenerate. The difference from the basic CSMC procedure
Figure 20: Empirical ACF for $\theta$ for the PGAS samplers using different numbers of particles $N$.

(Algorithm 16, page 121), however, is that they do not degenerate to the conditioned path. To understand the meaning of this, let us again look at two consecutive iterations of the sampler in more detail, using $N = 20$ and $T = 50$ as before. Figure 21 (top) shows the particles generated by the CSMC procedure with ancestor sampling, at iteration $r$. As pointed out above, due to path degeneracy, all the trajectories coincide for times $t$ far from the final time $T$. The extracted trajectory $x_{1:T}[r]$, corresponding to the ancestral path of particle $x^k_T$, is illustrated by a thick black line.

At the next iteration of the PGAS sampler we run CSMC with ancestor sampling, conditioned on $x_{1:T}[r]$. This results in the particle system shown in Figure 21 (bottom). As in Figures 17 and 19, the conditioned path is illustrated by a thick gray line. However, since, at each time point $t$, we sample a new ancestor for the conditioned particle $x_t[r]$, the conditioned path is broken up into pieces. This has the effect that the particle system generated at iteration $r + 1$ of PGAS degenerates to a path which, to a large extent, is different from the conditioned path $x_{1:T}[r]$. Similarly to using backward simulation, the effect is that PGAS explores the state-space around the conditioned path much better than PG, resulting in a faster mixing Gibbs kernel.

As could be seen in the example above, the use of ancestor sampling in PGAS gives a similar improvement over the basic PG sampler, as the backward simulation pass used in PGBS. This is not surprising, since both methods in fact make use of the same collections of Gibbs steps to sample from the extended target $\phi$. The difference between the methods lies in the order in which these Gibbs steps are carried out. Here, PGBS requires separate forward and backward passes, whereas PGAS only proceeds in the forward direction, without the need of an explicit backward pass.

Lindsten et al. (2012) point out some more practical differences between the methods. Primarily, they derive the PGAS sampler with the class of non-Markovian latent variable models in mind. They then make use of a weight truncation as discussed in Section 4.5,
Figure 21: Particle system generated by CSMC with ancestor sampling at iterations $r$ (top) and $r + 1$ (bottom) of the PGAS sampler. The dots show the particle positions, the thin black lines show the ancestral dependence of the particles and the thick black lines show the sampled trajectories $x_{1:T}[r]$ and $x_{1:T}[r + 1]$, respectively. In the bottom panel, the thick gray line fragments illustrates the conditioned path at iteration $r + 1$, with newly sampled ancestor indices. As an effect of ancestor sampling, the particle system at iteration $r + 1$ does not degenerate to the conditioned path.
and illustrate empirically that PGAS is more robust to the truncation error than alternative backward-simulation-based methods, such as PGBS. Additionally, they point out that PGAS can be implemented more straightforwardly than PGBS, especially in the case of non-Markovian models. The reason is that, since it only proceeds in the forward direction, there is no need to store or regenerate intermediate quantities from the SMC sampler, thus requiring less bookkeeping. For the same reason, PGAS can be more memory efficient than alternative backward-simulation-based methods, in particular when the state is high-dimensional or equipped with some high-dimensional statistic.

5.6 PMCMC for maximum likelihood inference

The PMCMC framework provides efficient methods for Bayesian parameter inference, which rely on the notion of exact approximation. However, as noted by Donnet and Samson (2011); Andrieu and Vihola (2011); Lindsten (2013), these attractive methods are not exclusive to the Bayesian. Indeed, it is possible to make use of PMCMC kernels when addressing, for instance, maximum likelihood problems as well. In this section, we review the method by Lindsten (2013), which is a procedure for maximum likelihood parameter inference in general SSMs, based on CSMC with ancestor sampling (Algorithm 19, page 131).

For ease of comparison with the PSEM methods discussed in Section 3.5 and to highlight the fact that we are indeed dealing with a maximum likelihood problem, we restrict our attention to SSMs. However, the same approach can straightforwardly be used for parameter inference in the class of general latent variable models considered above. Hence, consider an SSM, parameterized by \( \theta \in \Theta \),

\[
\begin{align*}
x_{t+1} &\sim f_\theta(x_{t+1} \mid x_t), \\
y_t &\sim g_\theta(y_t \mid x_t),
\end{align*}
\]

and \( x_1 \sim \mu_\theta(x_1) \). We observe a batch of measurements \( y_{1:T} \) and seek the maximum likelihood estimator,

\[
\hat{\theta}_{ML} = \arg \max_{\theta \in \Theta} \log p_\theta(y_{1:T}).
\]

In Section 3.5, we derived a PSEM algorithm to address this problem. However, we also noted that PSEM relies on asymptotics in the number of particles and backward trajectories to obtain a convergent sequence of parameter estimates. Hence, a computationally expensive particle smoother has to be run at each iteration of the algorithm, leading to a learning procedure with a very high computational cost.

This issue has been recognized as a problem with the basic Monte Carlo EM algorithm, of which PSEM can be seen as a generalization. To be able to make more efficient use of the simulated variables in Monte Carlo EM, a related method, referred to as stochastic approximation EM (SAEM) was proposed by Delyon et al. (1999). This method uses a stochastic approximation update of the auxiliary quantity of the EM algorithm. In the
SSM setting, this quantity is defined in (5), and we thus get the following approximation,\[
\hat{Q}_r(\theta) = (1 - \alpha_r)\hat{Q}_{r-1}(\theta) + \alpha_r \left( \frac{1}{M_r} \sum_{j=1}^{M_r} \log p_\theta(\tilde{x}_{1:T}, y_{1:T}) \right). \tag{132}
\]

In the vanilla form of SAEM, the samples \(\{\tilde{x}_{j1:T}\}_{j=1}^{M_r}\) are independent draws from the JSD \(p_{\theta[r-1]}(x_{1:T} \mid y_{1:T})\), parameterized by the current estimate \(\theta[r-1]\) (assuming for the time being that these samples can be generated).

As for PSEM, the M-step of the EM algorithm remains unchanged, but now we maximize the stochastic approximation \(\hat{Q}_r(\theta)\) instead of \(Q(\theta, \theta[r-1])\). In (132), \(\{\alpha_r\}_{r \geq 1}\) is a decreasing sequence of positive step sizes, satisfying the usual stochastic approximation conditions, \(\sum_r \alpha_r = \infty\) and \(\sum_r \alpha_r^2 < \infty\); see e.g. Kushner and Yin (1997). In SAEM, all simulated values contribute to \(\hat{Q}_{r-1}(\theta)\), but they are down-weighted using a forgetting factor given by the step size. Under appropriate assumptions, SAEM can be shown to converge for fixed \(M_r\) (e.g. \(M_r \equiv 1\)), as \(r \to \infty\) (Delyon et al., 1999; Cappé et al., 2005). When the simulation step is computationally involved, there is a considerable computational advantage of SAEM over Monte Carlo EM (Delyon et al., 1999).

To make use of this approach, we still need to generate samples from the JSD. One option is of course to apply a standard FFBSi to generate a collection of backward trajectories \(\{\tilde{x}_{i1:T}\}_{i=1}^{M_r}\) which are used in (132). Due to the fact that we now use a stochastic approximation update, a small number of backward trajectories (e.g. \(M_r \equiv 1\)), are sufficient. However, we still require these trajectories to be generated from a distribution with a small (and diminishing) discrepancy from the JSD. In other words, this approach still relies on asymptotics in the number of forward filter particles \(N_r\), in order to obtain accurate backward kernel approximations. This is not satisfactory, since we aim for a method which enjoys the exact approximation property obtained in the PMCMC setting.

To enable this, and thus be able to further reduce the computational complexity, we will use a Markovian version of stochastic approximation (Benveniste et al., 1990; Andrieu et al., 2005). It has been recognized that it is not necessary to sample exactly from the posterior distribution of the latent variables, to assess convergence of the SAEM algorithm. Indeed, it is sufficient to simulate from a family of Markov kernels \(\{K_\theta : \theta \in \Theta\}\), leaving the family of posteriors invariant (Kuhn and Lavielle, 2004). This is where PMCMC comes into play. Using any one of the PMCMC samplers presented above, we can construct a family of Markov kernels on \(X^T\), such that, for each \(\theta \in \Theta\), \(K_\theta(x_{1:T}' \mid x_{1:T})\) leaves the JSD \(p_\theta(x_{1:T} \mid y_{1:T})\) invariant. In combination with Markovian SAEM, we refer to this approach as particle SAEM (PSAEM).

To be more specific, we follow Lindsten (2013) and employ the CSMC with ancestor sampling presented in Algorithm 19 on page 131. That is, at iteration \(r\), we first run a CSMC sampler with ancestor sampling, targeting \(p_{\theta[r-1]}(x_{1:T} \mid y_{1:T})\). We then sample one of the particle trajectories, with probabilities given by the importance weights at the final time point. With a similar argument as was used to prove the validity of the PGAS sampler (see Section 5.5), this procedure will leave the JSD invariant.

Lindsten (2013) suggests to reuse all the particle trajectories in the approximation of the
Algorithm 21 PSAEM using CSMC with ancestor sampling

1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily. Set $\hat{Q}_0(\theta) \equiv 0$.
2: for $r \geq 1$ do
3: Run CSMC with ancestor sampling (Algorithm 19) targeting $p_{\theta[r-1]}(x_{1:T} | y_{1:T})$, conditioned on $x_{1:T}[r-1]$.
4: Compute $\hat{Q}_r(\theta)$ according to (133).
5: Compute $\theta[r] = \arg \max_{\theta \in \Theta} \hat{Q}_r(\theta)$.
6: if convergence criterion is met then
7: break
8: end if
9: Sample $k$ with $\mathbb{P}(k = i) = w^i_T$ and trace the ancestral path of particle $x^k_{1:T}$, i.e. set $x_{1:T}[r] = x^k_{1:T}$.
10: end for
11: return $\hat{\theta}_{PSAEM} = \theta[r]$.

auxiliary quantity $Q$. This amounts to a Rao-Blackwellization, similarly to (120). Hence, let $\{x^i_{1:T}, w^i_T\}_{i=1}^N$ be the weighted particle system generated by Algorithm 19. We then compute a stochastic approximation according to,

$$\hat{Q}_r(\theta) = (1 - \alpha_r)\hat{Q}_{r-1}(\theta) + \alpha_r \sum_{i=1}^N w^i_T \log p_\theta(x^i_{1:T}, y_{1:T}).$$

(133)

This approximation is then maximized w.r.t. $\theta$ in the M-step of the EM algorithm. We emphasize that, due to the use of stochastic approximation updates and the invariance property of the CSMC with ancestor sampling, this approach does not rely on asymptotics in $N$.

We summarize the method in Algorithm 21, and illustrate its performance in Example 19 below.

---

Example 19: PSAEM

We return to the nonlinear time-series model studied in Example 9. We use the same settings and the same batch of data with $T = 1500$ observations. The unknown parameters are given by the process noise and measurement noise variances, i.e. $\theta = (\sigma^2_v, \sigma^2_e)$. We apply Algorithm 21 with $N = 15$ particles for 2 000 iterations, initialized at $\theta[0] = (2, 2)$. We let $\alpha_r \equiv 1$ for $r \leq 100$, and $\alpha_r \sim r^{-0.7}$ for $r > 100$. This allows for a rapid change in the parameter estimates during the initial iterations, followed by a convergent phase. The resulting parameter estimates $\theta[r]$ are shown in Figure 22. As can be seen, the estimates converge to values close to the true parameters, despite the fact that we use a fixed (and small) number of particles. Compared to PSEM (see Figure 8), there is a significant improvement in terms of variance of the estimates and, most notably, in computational complexity.

---
5.7 PMCMC for state smoothing

Before we leave this section on PMCMC, it is worth to emphasize that all the PMCMC methods covered in this section can be used for state inference, as well as for parameter inference. The PMCMC samplers target the density $\gamma_T(\theta, x_{1:T})$, or, in the SSM setting, $p(\theta, x_{1:T} | y_{1:T})$. Consequently, if there are no unknown parameters in the models, the JSD $p(x_{1:T} | y_{1:T})$ can be obtained as a special case.

In fact, the addition of backward simulation to the PMMH sampler, as discussed in Section 5.3, is a way to improve the inference about the state, but does not alter the inferential performance for the parameter. Hence, this method should primarily be seen as a state smoother, and can be thought of as a way to Metropolise the backward simulator in Algorithm 12 (page 98) or the FFBSi in Algorithm 4 (page 72). See also Example 15.

Similarly, the PGAS and PGBS samplers can be applied equally well in the absence of any unknown parameter $\theta$, by simply skipping the corresponding Gibbs step. These methods can thus also be used as state smoothers. In fact, these methods could prove to be valuable alternatives to the backward simulator in Algorithm 12. Since they require much fewer particles than a stand-alone backward simulator, they could potentially reach higher accuracy at the same computational cost, despite the fact that they need many iterations of the outer MCMC loop. To date, no exhaustive comparison of these different approaches to smoothing has been made. Furthermore, it is likely that the preference for one method over another has to be investigated on a case by case basis.

6 Discussion

The purpose of this monograph has been to present and discuss various backward simulation methods for Monte Carlo statistical inference. We have focused on SMC-based backward simulators, which are useful for inference in analytically intractable models, such as nonlinear and/or non-Gaussian SSMs, but also in more general latent variable models. This is an active area of research. Inference for linear Gaussian and finite state-
space models is by now well understood. For nonlinear models, on the other hand, there are still many challenges and open problems. This is apparent, for instance, by looking at the large number of related methods that have been presented in the literature during the past few years, many of which have been reviewed in this monograph.

We have seen that there exists a large number of different approaches to backward simulation based on SMC. Indeed, backward simulation underlies a wide range of related particle smoothers, many of which were discussed in Sections 3–4 and in Section 5.7. Hence, when facing a certain state inference problem, we have many possible methods to choose from. Which method that is most suitable for the given problem has to be evaluated on a case by case basis, based on specific model properties, variance-bias trade-offs, etc. Computational complexities and overheads, memory usage and possibilities for parallelization are also important factors that have to be taken into account. The backward simulators presented in Sections 3–4 are particularly well suited for parallelization, since the backward trajectories can be generated independently. To the best of the authors’ knowledge, no exhaustive comparison of different particle smoothing algorithms has been conducted to date.

As we have seen, backward simulation is useful for both state inference and parameter inference. For instance, PMCMC (discussed in Section 5) enables Bayesian parameter learning in complicated latent variable models. In this context, the particle Gibbs samplers presented in Sections 5.4 and 5.5, used backward simulation as a way to do exact approximate grouping of the state variables, in order to improve the mixing over a single-state Gibbs sampler. We also made use of backward-simulation-based methods for maximum likelihood parameter inference through PSEM in Section 3.5 and particle SAEM in Section 5.6.

Both the Gibbs samplers and the EM-based methods for parameter inference, rely on inherent state inference problems to be solved. In EM, the E-step amounts to solving a smoothing problem, and in the Gibbs sampler, the states are used as auxiliary variables to enable learning of the parameters. Hence, it is not surprising that state inference procedures can be used as components in these algorithms. However, there are alternative methods for parameter inference based on SMC, which do not make use of intermediate state inference steps in the same way. These include the PMMH sampler presented in Section 5.2 and direct maximum likelihood methods (e.g. based on zero-order stochastic optimization). For these methods, it is not clear that backward simulation can aid in solving the parameter inference problem. Indeed, in Section 5.3 we saw that the acceptance probability of the PMMH sampler was unaffected by the introduction of a backward simulator. Whether or not it is possible to exploit the backward simulation idea for these parameter inference methods as well, is a question which requires further investigation.

Many of the algorithms discussed throughout this monograph can be thought of as (non-trivial) combinations of more basic algorithms. For instance, the samplers presented in Section 3.4 make use of MCMC within SMC to generate backward trajectories. PMCMC goes the other way around and makes use of SMC within MCMC to construct specialized Markov kernels. Due to the large number of alternative methods that have been developed, it is possible to come up with many more composite algorithms in the same way. For instance, the MH-IPS discussed in Section 9 makes use of single-state Gibbs sam-
plers to rejuvenate a degenerate particle system produced by a forward filter. Promising results have been reported for this method, but we have also experienced that single-state samplers can have slow convergence. An alternative is to make use of PMCMC samplers (e.g. PGAS, see Section 5.5) to update the particle trajectories in MH-IPS instead. This will, among other things, open up for easy parallelization of PMCMC samplers. It is not unlikely that such composite methods can prove to be serious competitors to the more basic ones, since they are able to exploit the strengths of each of the components. An obvious drawback, however, is that by combining more and more advanced methods, the implementation and analysis of the resulting algorithms will become increasingly more complex.

Another direction of future work is to tailor backward simulators for specific model classes. We saw an example of this with the Rao-Blackwellized FFBSi for conditionally linear Gaussian models in Section 4.4. Other models of central interest are those discussed in Section 4.1. For instance, the tree sampler for undirected graphical models discussed in Section 4.1.1 exploited a partitioning of the graph into disjoint chains. Under this partitioning, a sequential structure of the latent variables can be identified which makes the problem well suited for SMC and backward simulation. However, this partitioning is only one possibility. Hamze et al. (2006); Hamze and de Freitas (2004) consider other alternatives, e.g. to partition the graph into branching trees. Application of backward simulation to these models, or more generally to graphical models containing loops, requires further generalizations of the algorithmic framework. Interesting developments in this direction have been made by Sudderth et al. (2010); Ihler and McAllester (2009); Briers et al. (2005), who use SMC for loopy belief propagation in general undirected graphical models.

The example considered in Section 4.1.2, i.e. to use inference strategies to solve optimal control problems, is also an interesting topic for future work. As previously mentioned, Hoffman et al. (2009b); Doucet et al. (2010) have addressed this problem using trans-dimensional MCMC. By combining this approach with PMCMC, it might be possible to design efficient samplers for the optimal control problem in a rather general setting. If fact, the combination of trans-dimensional sampling and PMCMC is interesting in its own right, with many potential applications.

The challenging classes of non-Markovian models discussed in Section 4.6 are also of key relevance. Although it is possible to apply the general backward simulator of Algorithm 12 (page 98) to these models, this is plagued by a high computational cost. To find efficient samplers for these models is a topic for future work. There are also open challenges in applying backward-simulation-based methods to Bayesian nonparametric models (see e.g. Hjort et al. (2010)), such as Dirichlet process mixture models (Antoniak, 1974; Ferguson, 1973). The Chinese restaurant process (Aldous, 1985; Blackwell and MacQueen, 1973) can be thought of as a non-Markovian, sequential latent variable model, suggesting that the inference methods discussed in this monograph can be applied to models containing this structure. However, it is not obvious how to correctly represent the backward kernel in such nonparametric settings.
Bibliography


Bibliography


T. B. Schön and F. Lindsten. *Computational Learning in Dynamical Systems*. 2013. (forthcoming, draft manuscript is available from the authors).


Ancestor Sampling for Particle Gibbs

Authors: Fredrik Lindsten, Michael I. Jordan and Thomas B. Schön

Edited version of the paper:

Ancestor Sampling for Particle Gibbs

Fredrik Lindsten*, Michael I. Jordan** and Thomas B. Schön*

*Dept. of Electrical Engineering, Linköping University, SE–581 83 Linköping, Sweden
†lindsten,schon}@isy.liu.se
**Departments of EECS and Statistics, University of California, Berkeley CA 94720–1776, USA
jordan@eecs.berkeley.edu

Abstract

We present a novel method in the family of particle MCMC methods that we refer to as particle Gibbs with ancestor sampling (PGAS). Similarly to the existing PG with backward simulation (PGBS) procedure, we use backward sampling to (considerably) improve the mixing of the PG kernel. Instead of using separate forward and backward sweeps as in PGBS, however, we achieve the same effect in a single forward sweep. We apply the PGAS framework to the challenging class of non-Markovian state-space models. We develop a truncation strategy of these models that is applicable in principle to any backward-simulation-based method, but which is particularly well suited to the PGAS framework. In particular, as we show in a simulation study, PGAS can yield an order-of-magnitude improved accuracy relative to PGBS due to its robustness to the truncation error. Several application examples are discussed, including Rao-Blackwellized particle smoothing and inference in degenerate state-space models.

1 Introduction

State-space models (SSMs) are widely used to model time series and dynamical systems. The strong assumptions of linearity and Gaussianity that were originally invoked in state-space inference have been weakened by two decades of research on sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC). These Monte Carlo methods have not, however, led to substantial weakening of a further strong assumption, that of Markovianity. It remains a major challenge to develop inference algorithms for non-Markovian SSMs:

\[
\begin{align*}
{x}_{t+1} & \sim f(x_{t+1} \mid \theta, x_{1:t}), \\
y_t & \sim g(y_t \mid \theta, x_{1:t}),
\end{align*}
\]

where \(\theta \in \Theta\) is a static parameter with prior density \(p(\theta)\), \(x_t\) is the latent state and \(y_t\) is the observation at time \(t\), respectively. Models of this form arise in many different application scenarios, either from direct modeling or via a transformation or marginalization of a larger model. We provide several examples in Section 5.

To tackle the challenging problem of inference for non-Markovian SSMs, we work within the framework of particle MCMC (PMCMC), a family of inferential methods introduced
by Andrieu et al. (2010). The basic idea in PMCMC is to use SMC to construct a proposal kernel for an MCMC sampler. Assume that we observe a sequence of measurements $y_{1:T}$. We are interested in finding the density $p(x_{1:T}, \theta | y_{1:T})$, i.e., the joint posterior density of the state sequence and the parameter. In an idealized Gibbs sampler we would target this density by sampling as follows: (i) Draw $\theta^* \mid x_{1:T} \sim p(\theta \mid x_{1:T}, y_{1:T})$; (ii) Draw $x_{1:T}^* \mid \theta^* \sim p(x_{1:T} \mid \theta^*, y_{1:T})$. The first step of this procedure can be carried out exactly if conjugate priors are used. For non-conjugate models, one option is to replace Step (i) with a Metropolis-Hastings step. However, Step (ii)—sampling from the joint smoothing density $p(x_{1:T} \mid \theta, y_{1:T})$—is in most cases very difficult. In PMCMC, this is addressed by instead sampling a particle trajectory $x_{1:T}^*$ based on an SMC approximation of the joint smoothing density. More precisely, we run an SMC sampler targeting $p(x_{1:T} \mid \theta^*, y_{1:T})$. We then sample one of the particles at the final time $T$, according to their importance weights, and trace the ancestral lineage of this particle to obtain the trajectory $x_{1:T}^*$. This overall procedure is referred to as particle Gibbs (PG).

The flexibility provided by the use of SMC as a proposal mechanism for MCMC seems promising for tackling inference in non-Markovian models. To exploit this flexibility we must address a drawback of PG in the high-dimensional setting, which is that the mixing of the PG kernel can be very poor when there is path degeneracy in the SMC sampler (Whiteley et al., 2010; Lindsten and Schön, 2012). This problem has been addressed in the generic setting of SSMs by adding a backward simulation step to the PG sampler, yielding a method denoted PG with backward simulation (PGBS). It has been found that this considerably improves mixing, making the method much more robust to a small number of particles as well as larger data records (Whiteley et al., 2010; Lindsten and Schön, 2012).

Unfortunately, however, the application of backward simulation is problematic for non-Markovian models. The reason is that we need to consider full state trajectories during the backward simulation pass, leading to $O(T^2)$ computational complexity (see Section 4 for details). To address this issue, we develop a novel PMCMC method which we refer to as particle Gibbs with ancestor sampling (PGAS) that achieves the effect of backward sampling without an explicit backward pass. As part of our development, we also develop a truncation method geared to non-Markovian models. This method is a generic method that is also applicable to PGBS, but, as we show in a simulation study in Section 6, the effect of the truncation error is much less severe for PGAS than for PGBS. Indeed, we obtain up to an order of magnitude increase in accuracy in using PGAS when compared to PGBS in this study.

Since we assume that it is straightforward to sample the parameter $\theta$ of the idealized Gibbs sampler, we will not explicitly include sampling of $\theta$ in the subsequent sections to simplify our presentation.

## 2 Sequential Monte Carlo

We first review the standard auxiliary SMC sampler, see e.g. Doucet and Johansen (2011); Pitt and Shephard (1999). Let $\gamma_t(x_{1:t})$ for $t = 1, \ldots, T$ be a sequence of unnormalized densities on $X^t$, which we assume can be evaluated pointwise in linear time. Let
\( \tilde{\gamma} (x_{1:t}) \) be the corresponding normalized probability densities. For an SSM we would typically have \( \tilde{\gamma} (x_{1:t}) = p(x_{1:t} \mid y_{1:t}) \) and \( \gamma (x_{1:t}) = p(x_{1:t}, y_{1:t}) \). Assume that \( \{x_{1:t-1}^m, w_{t-1}^m\}_{m=1}^N \) is a weighted particle system targeting \( \tilde{\gamma}_{t-1} (x_{1:t-1}) \). This particle system is propagated to time \( t \) by sampling independently from a proposal kernel,

\[
M_t (a_t, x_t) = \frac{w_t a_t - 1 \nu_{t-1}^m}{\sum_{l=1}^N w_{t-1}^l \nu_{t-1}^l} R_t (x_t \mid x_{1:t-1}) \tag{2}
\]

In this formulation, the resampling step is implicit and corresponds to sampling the ancestor indices \( a_t \). Note that \( a_t^m \) is the index of the ancestor particle of \( x_t^m \). When we write \( x_1^m \), we refer to the ancestral path of \( x_t^m \). The factors \( \nu_t^m = \nu_t (x_{1:t}^m) \), known as adjustment multiplier weights, are used in the auxiliary SMC sampler to increase the probability of sampling ancestors that better describe the current observation (Pitt and Shephard, 1999). The particles are then weighted according to

\[
w_t^m = W_t (x_{1:t}^m),
\]

for \( t \geq 2 \). The procedure is initiated by sampling from a proposal density \( x_1^m \sim R_1 (x_1) \)

and assigning importance weights \( w_1^m = W_1 (x_1^m) \) with \( W_1 (x_1) = \gamma_1 (x_1) / R_1 (x_1) \). In PMCMC it is instructive to view this sampling procedure as a way of generating a single sample from the density

\[
\psi (x_{1:T}, a_{2:T}) \triangleq \prod_{m=1}^N R_1 (x_1^m) \prod_{t=2}^T \prod_{m=1}^N M_t (a_t^m, x_t^m) \tag{4}
\]

on the space \( X^{NT} \times \{1, \ldots, N\}^{NT} \). Here we have introduced the boldface notation \( x_t = \{x_t^1, \ldots, x_t^N\} \) and similarly for the ancestor indices.

### 3 Particle Gibbs with ancestor sampling

PMCMC methods is a class of MCMC samplers in which SMC is used to construct proposal kernels (Andrieu et al., 2010). The validity of these methods can be assessed by viewing them as MCMC samplers on an extended state space in which all the random variables generated by the SMC sampler are seen as auxiliary variables. The target density on this extended space is given by

\[
\phi (x_{1:T}, a_{2:T}, k) \triangleq \frac{\tilde{\gamma}_T (x_{1:T})}{NT} \frac{\psi (x_{1:T}, a_{2:T})}{R_1 (x_1^b_1) \prod_{t=2}^T \prod_{m=1}^N M_t (a_t^m, x_t^m)}. \tag{5}
\]

By construction, this density admits \( \tilde{\gamma}_T (x_{1:T}) \) as a marginal, and can thus be used as a surrogate for the original target density \( \tilde{\gamma}_T \) (Andrieu et al., 2010). Here \( k \) is a variable indexing one of the particles at the final time point and \( b_{1:T} \) corresponds to the ancestral path of this particle: \( x_{1:T}^k = x_1^{b_1:T} = \{x_1^{b_1}, \ldots, x_T^{b_T}\} \). These indices are given recursively from the ancestor indices by \( b_T = k \) and \( b_t = a_{t+1}^{b_t} \). The PG sampler by Andrieu et al. (2010) is a Gibbs sampler targeting \( \phi \) using the following sweep (note that \( b_{2:T} = \{a_{2:T}^m, b_T\} \)).
1. Draw $x_{1:T}^{*, b_1:T}, a_{2:T}^{*, b_2:T} \sim \phi(x_{1:T}^{-b_1:T}, a_{2:T}^{-b_2:T} | x_{1:T}^{b_1:T}, b_{1:T})$.

2. Draw $k^* \sim \phi(k | x_{1:T}^{*, b_1:T}, a_{2:T}^{*, b_2:T}, x_{1:T}^{b_1:T}, a_{2:T}^{b_2:T})$.

Here we have introduced the notation $x_{t}^{-m} = \{x_{1}^{1}, \ldots, x_{t}^{m-1}, x_{t}^{m+1}, \ldots, x_{t}^{N}\}$, $x_{1:T}^{-b_1:T} = \{x_{1}^{1}, \ldots, x_{T}^{b_2:T}\}$ and similarly for the ancestor indices. In (Andrieu et al., 2010), a sequential procedure for sampling from the conditional density appearing in Step 1 is given. This method is known as conditional SMC (CSMC). It takes the form of an SMC sampler in which we condition on the event that a prespecified path $x_{1:T}^{b_1:T} = x_{1:T}',$ with indices $b_{1:T},$ is maintained throughout the sampler (see Algorithm 1 for a related procedure). Furthermore, the conditional distribution appearing in Step 2 of the PG sampler is shown to be proportional to $w_{T}^{b}$, and it can thus straightforwardly be sampled from.

Note that we never sample new values for the variables $\{x_{1:T}^{b_1:T}, b_{1:T-1}\}$ in this sweep. Hence, the PG sampler is an “incomplete” Gibbs sampler, since it does not loop over all the variables of the model. It still holds that the PG sampler is ergodic, which intuitively can be explained by the fact that the collection of variables that is left out is chosen randomly at each iteration. However, it has been observed that the PG sampler can have very poor mixing, especially when $N$ is small and/or $T$ is large (Whiteley et al., 2010; Lindsten and Schön, 2012). The reason for this poor mixing is that the SMC path degeneracy causes the collections of variables that are left out at any two consecutive iterations to be strongly dependent.

We now turn to our new procedure, PGAS, which aims to address this fundamental issue. Our idea is to sample new values for the ancestor indices $b_{1:T-1}$ as part of the CSMC procedure$^1$. By adding these variables to the Gibbs sweep, we can considerably improve the mixing of the PG kernel. The CSMC method is a sequential procedure to sample from $\phi(x_{1:T}^{-b_1:T}, a_{2:T}^{-b_2:T} | x_{1:T}^{b_1:T}, b_{1:T})$ by sampling according to $\{x_{t}^{*, b_t}, a_{t}^{*, b_t}\} \sim \phi(x_{t}^{-b_t}, a_{t}^{-b_t} | x_{1:t-1}^{*, b_{1:t-1}}, a_{2:t-1}^{*, b_{2:t-1}}, x_{1:T}^{b_1:T}, b_{1:T})$, for $t = 1, \ldots, T$. After having sampled these variables at time $t$, we add a step in which we generate a new value for $b_{t-1} (= a_{t}^{b_t})$, resulting in the following sweep:

1’. (CSMC with ancestor sampling) For $t = 1, \ldots, T$, draw

$$x_{t}^{*, b_t}, a_{t}^{*, b_t} \sim \phi(x_{t}^{-b_t}, a_{t}^{-b_t} | x_{1:t-1}^{*, b_{1:t-1}}, a_{2:t-1}^{*, b_{2:t-1}}, x_{1:T}^{b_1:T}, b_{1:T}),$$

$$(a_{t}^{*, b_t} =) b_{t-1}^{*} \sim \phi(b_{t-1} | x_{1:t-1}^{*, b_{1:t-1}}, a_{2:t-1}^{*, b_{2:t-1}}, x_{1:T}^{b_1:T}, b_{1:T}).$$

2’. Draw $(k^* =) b_{T}^{*} \sim \phi(b_{T} | x_{1:T}^{*, b_1:T}, a_{2:T}^{*, b_2:T}, x_{1:T}^{b_1:T}).$

It can be verified that this corresponds to a partially collapsed Gibbs sampler (Dyk and Park, 2008) and will thus leave $\phi$ invariant. To determine the conditional densities from which the ancestor indices are drawn, consider the following factorization, following di-

---

$^1$Ideally, we would like to include the variables $x_{1:T}^{b_1:T}$ as well, but this is in general not possible since it would be similar to sampling from the original target density (which we assume is infeasible).
rectly from (3),
\[ \gamma_t(x_{1:t}) = W_t(x_{1:t}) \nu_{t-1}(x_{1:t-1}) R_t(x_t | x_{1:t-1}) \gamma_{t-1}(x_{1:t-1}) \]
\[ \Rightarrow \gamma_t(x_{1:t}) = w_t^{-b_1} \sum_l \nu_{t-1}^{l, t} \frac{w_{t-1}^{b_t-1} \nu_{t-1}^{l-1}}{w_{t-1}^{b_t-1}} R_t(x_t^{b_t-1} | x_{1:t-1}) \gamma_{t-1}(x_{1:t-1}) \]
\[ = \cdots = w_t^{-b_1} \left( \prod_{s=1}^{t-1} \sum_l u_{s}^{l, t} \right) R_1(x_1^{b_1}) \prod_{s=2}^t M_t(a_s^{b_s}, x_s^{b_s}). \] (6)

Furthermore, we have
\[ \phi(b_t | x_{1:t}, a_{2:t}, x_{t+1:T}, b_{t+1:T}) \propto \phi(x_{1:t}, a_{2:t}, x_{t+1:T}, b_{t:T}) \]
\[ \propto \frac{\gamma_T(x_{1:T}^k) \psi(x_{1:t}, a_{2:t})}{R_1(x_1^{b_1}) \prod_{s=2}^t M_s(a_s^{b_s}, x_s^{b_s})} \propto \frac{\gamma_T(x_{1:T}^k)}{\gamma_T(x_{1:t}^{b_t})} \frac{\gamma_T(x_{1:t}^{b_t})}{\gamma_t(x_1^{b_1})} \prod_{s=2}^t M_s(a_s^{b_s}, x_s^{b_s}). \] (7)

By plugging (6) into the numerator we get,
\[ \phi(b_t | x_{1:t}, a_{2:t}, x_{t+1:T}, b_{t+1:T}) \propto w_t^{-b_1} \frac{\gamma_T(x_{1:T}^k)}{\gamma_t(x_1^{b_1})}. \] (8)

Hence, to sample a new ancestor index for the conditioned path at time \( t + 1 \), we proceed as follows. Given \( x_{t+1:T}^{'} = x_{t+1:T}^{b_t} \) we compute the backward sampling weights,
\[ w_{t|T}^m = w_t^m \frac{\gamma_T(x_{1:T}^m, x_{t+1:T}^{'})}{\gamma_t(x_1^{b_1})}, \] (9)

for \( m = 1, \ldots, N \). We then set \( b_t = m \) with probability proportional to \( w_{t|T}^m \).

It follows that the proposed CSMC with ancestor sampling (Step 1'), conditioned on \( \{x_{1:T}^{'}, b_{1:T}\} \), can be realized as in Algorithm 1. The difference between this algorithm and the CSMC sampler derived by Andrieu et al. (2010) lies in the ancestor sampling step 2(b) (where instead, they set \( a_t^{b_t} = b_{t-1} \)). By introducing the ancestor sampling, we break the strong dependence between the generated particle trajectories and the path on which we condition. We call the resulting method, defined by Steps 1' and 2' above, PG with ancestor sampling (PGAS).

**Algorithm 1** CSMC with ancestor sampling, conditioned on \( \{x_{1:T}^{'}, b_{1:T}\} \)

1. Draw \( x_{1:T}^m \sim R_1(x_1) \) for \( m \neq b_1 \) and set \( x_1^{b_1} = x_1^{'}. \)
2. Set \( w_1^m = W_1(x_1^m) \) for \( m = 1, \ldots, N. \)
3. for \( t = 2 \) to \( T \) do
   4. Draw \( \{a_t^m, x_t^m\} \sim M_t(a_t, x_t) \) for \( m \neq b_t \) and set \( x_t^{b_t} = x_t^{'} \).
   5. Draw \( a_t^{b_t} \) with \( \mathbb{P}(a_t^{b_t} = m) \propto w_{t-1|T}^m \).
   6. Set \( x_{1:T}^m = \{x_{1:t-1}^m, x_{t}^m\} \) and \( w_t^m = W_t(x_{1:T}^m) \) for \( m = 1, \ldots, N. \)
4. end for

The idea of including the variables \( b_{1:T-1} \) in the PG sampler has previously been suggested by Whiteley (2010) and further explored by Whiteley et al. (2010); Lindsten and Schön (2012). This previous work, however, accomplishes this with a explicit backward
simulation pass, which, as we discuss in the following section, is problematic for our applications to non-Markovian SSMs. In the PGAS sampler, instead of requiring distinct forward and backward sequences of Gibbs steps as in PG with backward simulation (PGBS), we obtain a similar effect via a single forward sweep.

4 Truncation for non-Markovian state-space models

We return to the problem of inference in non-Markovian SSMs of the form shown in (1). To employ backward sampling, we need to evaluate the ratio

$$\frac{\gamma_T(x_{1:T})}{\gamma_t(x_{1:t})} = \frac{p(x_{1:T}, y_{1:T})}{p(x_{1:t}, y_{1:t})} = \prod_{s=t+1}^{T} g(y_s | x_{1:s}) f(x_{s} | x_{1:s-1}).$$

(10)

In general, the computational cost of computing the backward sampling weights will thus be $O(T)$. This implies that the cost of generating a full backward trajectory is $O(T^2)$. It is therefore computationally prohibitive to employ backward simulation type of particle smoothers, as well as the PG samplers discussed above, for general non-Markovian models.

To make progress, we consider non-Markovian models in which there is a decay in the influence of the past on the present, akin to that in Markovian models but without the strong Markovian assumption. Hence, it is possible to obtain a useful approximation when the product in (10) is truncated to a smaller number of factors, say $p$. We then replace (9) with the approximation,

$$\tilde{w}_{t|m}^{p,m} = w_{t|m}^{p} \frac{\gamma_t(x_{1:t})}{\gamma^{t+p}(\{x_{1:t}, x'_{t+1:t+p}\})}.$$  

(11)

The following proposition formalizes our assumption.

**Proposition 1.** Let $P$ and $\tilde{P}_p$ be the probability distributions on $\{1, \ldots, N\}$, defined by the backward sampling weight (9) and the truncated backward sampling weights (11), respectively. Let $h_s(k) = g(y_{t+s} | x'_{1:t}, x'_{t+1:t+s}) f(x'_{t+s} | x_{1:t}, x'_{t+1:t-s})$ and assume that $\max_{k,t} (h_s(k)/h_s(l) - 1) \leq A \exp(-cs)$, for some constants $A$ and $c > 0$. Then, $D_{\text{KLD}}(P||\tilde{P}_p) \leq C \exp(-cp)$ for some constant $C$, where $D_{\text{KLD}}$ is the Kullback-Leibler divergence (KLD).

**Proof:** See Appendix A.

From (11), we see that we can compute the backward weights in constant time under the truncation within the PGAS framework. The resulting approximation can be quite useful; indeed, in our experiments we have seen that even $p = 1$ can lead to very accurate inferential results. In general, however, it will not be known a priori how to set the truncation level $p$ for any given problem. To address this problem, we propose to use an adaptation of the truncation level. Since the approximative weights (11) can be evaluated sequentially, the idea is to start with $p = 1$ and then increase $p$ until the weights have, in some sense,
5 Application areas

In this section we present examples of problem classes involving non-Markovian SSMs for which the proposed PGAS sampler can be applied. Numerical illustrations are provided in Section 6.

5.1 Rao-Blackwellized particle smoothing

One popular approach to increase the efficiency of SMC samplers for SSMs is to marginalize over one component of the state, and apply an SMC sampler in the lower-dimensional

Figure 1: Probability under $\tilde{P}_p$ as a function of the truncation level $p$ for two different systems; one 5 dimensional (left) and one 20 dimensional (right). The $N = 5$ dotted lines correspond to $\tilde{P}_p(m)$ for $m \in \{1, \ldots, N\}$, respectively (N.B. two of the lines overlap in the left figure). The dashed vertical lines show the value of the truncation level $p_{\text{adpt}}$, resulting from the adaption scheme with $\gamma = 0.1$ and $\tau = 10^{-2}$. See Section 6.2 for details on the experiments.
marginal space. This leads to what is known as the Rao-Blackwellized particle filter (RBPF) (Chen and Liu, 2000; Doucet et al., 2000; Schön et al., 2005). The same approach has also been applied to state smoothing (Särkkä et al., 2012; Fong et al., 2002), but it turns out that Rao-Blackwellization is less straightforward in this case, since the marginal state-process will be non-Markovian. As an example, a mixed linear/nonlinear Gaussian SSM (see, e.g., Schön et al. (2005)) with “nonlinear state” \( x_t \) and “linear state” \( z_t \), can be reduced to \( x_t \sim p(x_t | x_{1:t-1}, y_{1:t-1}) \) and \( y_t \sim p(y_t | x_{1:t}, y_{1:t-1}) \). These conditional densities are Gaussian and can be evaluated for any fixed marginal state trajectory \( x_{1:t-1} \) by running a conditional Kalman filter to marginalize the \( z_t \)-process.

In order to apply a backward-simulation-based method (e.g., a particle smoother) for this model, we need to evaluate the backward sampling weights (9). In a straightforward implementation\(^2\), we thus need to run \( N \) Kalman filters for \( T - t \) time steps, for each \( t = 1, \ldots, T - 1 \). The computational complexity of this calculation can be reduced by employing the truncation proposed in Section 4.

### 5.2 Particle smoothing for degenerate state-space models

Many dynamical systems are most naturally modelled as degenerate in the sense that the transition kernel of the state process does not admit any dominating measure. For instance, consider a nonlinear system with additive noise of the form,

\[
\xi_t = f(\xi_{t-1}) + G\omega_{t-1}, \quad y_t = g(\xi_t) + e_t, \tag{12}
\]

where \( G \) is a tall matrix, and consequently \( \text{rank}(G) < \text{dim}(\xi_t) \). That is, the process noise covariance matrix is singular. SMC samplers can straightforwardly be applied to this type of models, but it is more problematic to address the smoothing problem using particle methods. The reason is that the backward kernel also will be degenerate and it cannot be approximated in a natural way by the forward filter particles, as is normally done in backward-simulation-based particle smoothers.

A possible remedy for this issue is to recast the degenerate SSM as a non-Markovian model in a lower-dimensional space. Let \( G = U \Sigma V^T \) with unitary \( U \) and \( V \) be a singular value decomposition of \( G \) and let,

\[
\begin{bmatrix} x_t \\ z_t \end{bmatrix} \triangleq U^T \xi_t = U^T f(UU^T \xi_{t-1}) + \left[ \Sigma V^T \omega_{t-1} \right]. \tag{13}
\]

For simplicity we assume that \( z_1 \) is known. If this is not the case, it can be included in the system state or seen as a static parameter of the model. Hence, the sequence \( z_{1:t} \) is \( \sigma(x_{1:t-1}) \)-measurable and we can write \( z_t = z_t(x_{1:t-1}) \). With \( v_t \triangleq \Sigma V^T \omega_t \) and by appropriate definitions of the functions \( f_x \) and \( h \), the model (12) can thus be rewritten as, \( x_t = f_x(x_{1:t-1}) + v_{t-1} \) and \( y_t = h(x_{1:t}) + e_t \), which is a non-degenerate, non-Markovian SSM. By exploiting the truncation proposed in Section 4 we can thus apply PGAS to do inference in this model.

\(^2\)For the specific problem of Rao-Blackwellized smoothing in conditionally Gaussian models, a backward simulator which can be implemented in \( O(T) \) computational complexity has recently been proposed by Särkkä et al. (2012). This is based on the idea of propagating information backward in time as the backward samples are generated.
5.3 Additional problem classes

There are many more problem classes in which non-Markovian models arise and in which backward-simulation-based methods can be of interest. For instance, the Dirichlet process mixture model (DPMM, see, e.g., Hjort et al. (2010)) is a popular nonparametric Bayesian model for mixtures with an unknown number of components. Using a Polya urn representation, the mixture labels are given by a non-Markovian stochastic process, and the DPMM can thus be seen as a non-Markovian SSM. SMC has previously been used for inference in DPMMs (MacEachern et al., 1999; Fearnhead, 2004). An interesting venue for future work is to use the PGAS sampler for these models. A second example in Bayesian nonparametrics is Gaussian process (GP) regression and classification (see, e.g., Rasmussen and Williams (2006)). The sample path of the GP can be seen as the state-process in a non-Markovian SSM. We can thus employ PMCMC, and in particular PGAS, to address these inference problems.

An application in genetics, for which SMC has been been successfully applied, is reconstruction of phylogenetic trees (Bouchard-Côté et al., 2012). A phylogenetic tree is a binary tree with observation at the leaf nodes. SMC is used to construct the tree in a bottom up fashion. A similar approach has also been used for Bayesian agglomerative clustering, in which SMC is used to construct a binary clustering tree based on Kingman’s coalescent (Teh et al., 2008). The generative models for the trees used by Bouchard-Côté et al. (2012); Teh et al. (2008) are in fact Markovian, but the observations give rise to a conditional dependence which destroys the Markov property. To employ backward simulation to these models, we are thus faced with problems of a similar nature as those discussed in Section 4.

6 Numerical evaluation

This section contains a numerical evaluation of the proposed method. We consider linear Gaussian systems, which is instructive since the exact smoothing density then is available, e.g., by running a modified Bryson-Frazier (MBF) smoother (Bierman, 1973). For more details on the experiments, and for additional (nonlinear) examples, see Lindsten et al. (2012b).

6.1 RBPS: Linear Gaussian state-space model

As a first example, we consider Rao-Blackwellized particle smoothing (RBPS) in a single-output 4th-order linear Gaussian SSM. We generate $T = 100$ samples from the system and run PGAS and PGBS, marginalizing three out of the four states using an RBPF, i.e., $\dim(x_t) = 1$. Both methods are run for $R = 10000$ iterations using $N = 5$ particles. The truncation level is set to $p = 1$, leading to a coarse approximation. We discard the first 1000 iterations and then compute running means of the state trajectory $x_{1:T}$. From these, we then compute the running root mean squared errors (RMSEs) $\epsilon_r$ relative to the true posterior means (computed with an MBF smoother). Hence, if no approximation would have been made, we would expect $\epsilon_r \to 0$, so any static error can be seen as the effect of the truncation. The results for five independent runs from both PG samplers are shown in Figure 2. First, we note that both methods give accurate results. Still, the error
for PGAS is close to an order of magnitude less than for PGBS. Furthermore, it appears as if the error for PGAS would decrease further, given more iterations, suggesting that the bias caused by the truncation is dominated by the Monte Carlo variance, even after $R = 10000$ iterations.

For further comparison, we also run an untruncated forward filter/backward simulator (FFBSi) particle smoother (Godsill et al., 2004), using $N = 5000$ forward filter particles and $M = 500$ backward trajectories (with a computational complexity of $O(NMT^2)$). The resulting RMSE value is shown as a solid line in Figure 2. These results suggest that PMCMC samplers, such as the PGAS, indeed can be serious competitors to more “standard” particle smoothers. Even with $p = 1$, PGAS outperforms FFBSi in terms of accuracy and, due to the fact that the ancestor sampling allows us to use as few as $N = 5$ particles at each iteration, at a lower computational cost.

### 6.2 Random linear Gaussian systems with rank deficient process noise covariances

To see how the PG samplers are affected by the choice of truncation level $p$ and by the mixing properties of the system, we evaluate them on random linear Gaussian SSMs of different orders. We generate 150 random systems, using the MATLAB function `drss` from the Control Systems Toolbox, with model orders 2, 5 and 20 (50 systems for each model order). The number of outputs are taken as 1, 2 and 4 for the different model orders, respectively. The systems are then simulated for $T = 200$ time steps, driven by Gaussian process noise entering only on the first state component. Hence, the rank of the process noise covariance is 1 for all systems.

We run the PGAS and PGBS samplers for 10000 iterations using $N = 5$ particles. We consider different fixed truncation levels, as well as an adaptive level with $\gamma = 0.1$ and $\tau = 10^{-2}$. Again, we compute running posterior means (discarding 1000 samples) and RMSE values relative the true posterior mean. Box plots are shown in Figure 3. Since the process noise only enters on one of the state components, the mixing tends to deteriorate as we increase the model order. Figure 1 shows how the probability distributions on $\{1, \ldots, N\}$ change as we increase the truncation level, in two representative cases for a 5th and a 20th order system, respectively. By using an adapted level, we can obtain
Figure 3: Box plots of the RMSE errors for PGAS (black) and PGBS (gray), for 150 random systems of different dimensions $d$ (left, $d = 2$; middle, $d = 5$; right, $d = 20$). Different values for the truncation level $p$ are considered. The rightmost boxes correspond to an adaptive threshold and the values in parentheses are the average over all systems and MCMC iterations (the same for both methods). The dots within the boxes show the median errors.

accurate results for systems of different dimensions, without having to change any settings between the runs.

7 Discussion

PGAS is a novel approach to PMCMC that makes use of backward simulation ideas without needing an explicit backward pass. Compared to PGBS, a conceptually similar method that does require an explicit backward pass, PGAS has advantages, most notably for inference in the non-Markovian SSMs that have been our focus here. When using the proposed truncation of the backward weights, we have found PGAS to be more robust to the approximation error than PGBS. Furthermore, for non-Markovian models, PGAS is easier to implement than PGBS, since it requires less bookkeeping. It can also be more memory efficient, since it does not require us to store intermediate quantities that are needed for a separate backward simulation pass, as is done in PGBS. Finally, we note that PGAS can be used as an alternative to PGBS for other inference problems to which PMCMC can be applied, and we believe that it will prove attractive in problems beyond the non-Markovian SSMs that we have discussed here.

Appendix

A Proof of Proposition 1

With $M = T - t$ and $w(k) = w^k_t$, the distributions of interest are given by

$$P(k) = \frac{w(k) \prod_{s=1}^{M} h_s(k)}{\sum_{l=1}^{N} w(l) \prod_{s=1}^{M} h_s(l)}$$

and

$$\tilde{P}_p(k) = \frac{w(k) \prod_{s=1}^{p} h_s(k)}{\sum_{l=1}^{N} w(l) \prod_{s=1}^{p} h_s(l)},$$
respectively. Let $\varepsilon_s \triangleq \max_{k,l} \left( h_s(k)/h_s(l) - 1 \right) \leq A \exp(-cs)$ and consider

$$\left( \sum_l w(l) \prod_{s=1}^p h_s(l) \right) \prod_{s=p+1}^M h_s(k) \leq \sum_l \left( w(l) \prod_{s=1}^p h_s(l) \prod_{s=p+1}^M h_s(l)(1 + \varepsilon_s) \right) = \left( \sum_l w(l) \prod_{s=1}^M h_s(l) \right) \prod_{s=p+1}^M (1 + \varepsilon_s).$$

It follows that the KLD is bounded according to,

$$D_{\text{KLD}}(P\|\tilde{P}_p) = \sum_k P(k) \log \frac{P(k)}{\tilde{P}_p(k)} = \sum_k P(k) \log \left( \frac{\prod_{s=p+1}^M h_s(k) \left( \sum_l w(l) \prod_{s=1}^p h_s(l) \right)}{\sum_l w(l) \prod_{s=1}^M h_s(l)} \right) \leq \sum_k P(k) \sum_{s=p+1}^M \log(1 + \varepsilon_s) \leq \sum_{s=p+1}^M \varepsilon_s \leq A \sum_{s=p+1}^M \exp(-cs) = A \frac{e^{-c(p+1)} - e^{-c(M+1)}}{1 - e^{-c}}.$$

$\square$
Bibliography


Authors: Fredrik Lindsten, Thomas B. Schön and Michael I. Jordan

Edited version of the paper:

Parts of the theory presented in this paper have also been presented in:
Abstract

We present a novel method for Wiener system identification. The method relies on a semiparametric, i.e. a mixed parametric/nonparametric, model of a Wiener system. We use a state-space model for the linear dynamical system and a nonparametric Gaussian process model for the static nonlinearity. We avoid making strong assumptions, such as monotonicity, on the nonlinear mapping. Stochastic disturbances, entering both as measurement noise and as process noise, are handled in a systematic manner. The nonparametric nature of the Gaussian process allows us to handle a wide range of nonlinearities without making problem-specific parameterizations. We also consider sparsity-promoting priors, based on generalized hyperbolic distributions, to automatically infer the order of the underlying dynamical system. We derive an inference algorithm based on an efficient particle Markov chain Monte Carlo method, referred to as particle Gibbs with ancestor sampling. The method is profiled on two challenging identification problems with good results. Blind Wiener system identification is handled as a special case.

1 Introduction

Block-oriented systems are a useful and general class of nonlinear dynamical systems. These systems consist of interconnected linear dynamics and static nonlinearities. The most well-known members of this family are the Hammerstein (static nonlinearity followed by a linear dynamical system) and the Wiener (linear dynamical system followed by a static nonlinearity) systems, introduced by Hammerstein (1930) and Wiener (1966), respectively. In this work, we are concerned with identification of the latter class. Based on observed inputs \( u_{1:T} \triangleq \{u_t\}_{t=1}^T \) and outputs \( y_{1:T} \), we wish to infer the linear dynamical system \( G \) and the static nonlinearity \( h \) of the Wiener system depicted in Figure 1.

Wiener systems have attracted significant attention in the system identification community, as is evident from the vast literature on the topic. See e.g. Wills et al. (2013); Giri and Bai (2010); Pillonetto and Chiuso (2009); Greblicki and Pawlak (2008); Raich et al. (2005); Greblicki (1997); Kalafatis et al. (1997); Westwick and Verhaegen (1996) and...
the references therein. However, the approach presented here differs from the existing literature on several accounts.

We consider a semiparametric (i.e., a mixed parametric/nonparametric) model of a Wiener system, in which a parametric state-space model is used for the linear block $G$ and a nonparametric model is used for the nonlinear block $h(\cdot)$. Let $\theta = \{G, h(\cdot)\}$ denote the unknowns of the system, i.e. $\theta$ contains both the parameters of $G$ and the nonparametric representation of $h(\cdot)$ (the precise definition of $\theta$ will be made clear in Section 3). We take a Bayesian approach, modeling the parameters as random variables and the nonparametric function $h(\cdot)$ as a stochastic process. In particular, we use a Gaussian process (GP) model for $h(\cdot)$. We then provide a method for computing $p(\theta \mid y_{1:T})$, the posterior probability density function (PDF) of $\theta$ given the measurements $y_{1:T}$ (and, implicitly, the inputs $u_{1:T}$). To the best of our knowledge, this is the first time the posterior PDF $p(\theta \mid y_{1:T})$ is computed for the Wiener identification problem.

In this probabilistic framework, we can handle stochastic disturbances in a systematic manner. Most notably, we are able to deal with process noise entering internally to the linear dynamical system, which can be critical in obtaining an accurate model (Wills et al., 2013). The inclusion of such process noise in the model significantly complicates the estimation problem and is therefore often neglected in the existing literature. Furthermore, due to the nonparametric nature of the GP, the proposed method is flexible. It can be used for a wide range of nonlinear mappings, without making any problem-specific parameterizations. We do not impose strong assumptions such as invertibility or monotonicity of the nonlinearity.

The posterior PDF $p(\theta \mid y_{1:T})$ does not allow for a closed-form solution. To cope with this, we make use of a Markov Chain Monte Carlo (MCMC) method (see e.g. Robert and Casella (2004) for a general introduction) to compute an approximation of $p(\theta \mid y_{1:T})$. More specifically, we employ the recently proposed particle MCMC (PMCMC) framework (Andrieu et al., 2010). The basic idea underlying PMCMC is to use a particle filter (PF) as a component of an MCMC sampler. This is done in a manner such that, for any fixed (and finite) number of particles, no systematic error is introduced. Here, we use a state-of-the-art PMCMC method denoted particle Gibbs with ancestor sampling (PGAS), which has been found to be efficient even when using few particles in the underlying PF (Lindsten et al., 2012a).

Finally, we note that the proposed method can also be applied in the absence of any (measured) exogenous input $u_t$. This problem, referred to as blind Wiener system identification, has received considerable attention on its own (Wills et al., 2011; Vanbeylen...
et al., 2009; Bai, 2002; Abed-Meraim et al., 1997), and it can be treated as a special case of the proposed method. We have published a preliminary version of the current work (specifically targeting the blind identification problem) in (Lindsten et al., 2012b).

2 A Bayesian semiparametric model

We consider a semiparametric model of a Wiener system. The linear dynamical system is modeled using a (parametric) state-space representation, and a nonparametric GP model is used for the static nonlinearity. The model can be described in state-space form as

\[
\begin{align*}
    x_{t+1} &= Ax_t + Bu_t + w_t, & w_t &\sim \mathcal{N}(0, Q), \\
    z_t &= Cx_t, & (1a) \\
    y_t &= h(z_t) + e_t, & e_t &\sim \mathcal{N}(0, r). & (1c)
\end{align*}
\]

Here, \( x_t \in \mathbb{R}^{n_x} \) is the state of the dynamical system, \( w_t \in \mathbb{R}^{n_w} \) is the process noise, \( u_t \in \mathbb{R}^{n_u} \) is the input signal, \( z_t \in \mathbb{R} \) is the output from the linear block and \( y_t \in \mathbb{R} \) is the output from the static nonlinearity \( h(z_t) \) with measurement noise \( e_t \) added. For clarity, we will write \( X \triangleq \mathbb{R}^{n_x} \) and \( Y \triangleq \mathbb{R} \) for the state-space and measurement space, respectively. For simplicity, we have restricted our attention to multiple-input single-output systems, since we then only have to consider one-dimensional GPs. However, the proposed method can be extended to multiple outputs via the use of a multidimensional GP.

The linear system is assumed to be observable. Hence, we can, without loss of generality, fix the matrix \( C \) according to \( C = (1 \ 0 \ \cdots \ 0) \). Let \( \Gamma = [A \ B] \). Then, the unknown quantities of the model are the system parameters \( \Gamma, Q \) and \( r \) as well as the nonlinear mapping \( h(\cdot) \). We take a Bayesian approach and model the parameters as random variables. In the two subsequent sections we describe two different models that will be employed for the linear dynamics and in Section 2.3 the GP model for the nonlinearity is introduced.

2.1 Alt. I – Conjugate priors

If the order of the dynamical system \( n_x \) is assumed to be known, we can place conjugate priors on the matrices describing the linear dynamics. Conjugate priors are commonly used in Bayesian statistics, since they result in closed-form expressions for the posterior distributions. A conjugate prior for the linear Gaussian model \((1a)\) is the matrix normal, inverse Wishart (MNIW) distribution. Hence, we place an MNIW prior on the pair \( \{\Gamma, Q\} \),

\[
p(\Gamma, Q) = p(\Gamma \mid Q)p(Q),
\]

where,

\[
\begin{align*}
    p(\Gamma \mid Q) &= \mathcal{MN}(\Gamma; M, Q, L), & (2a) \\
    p(Q) &= \mathcal{IW}(Q; n_0, S_0). & (2b)
\end{align*}
\]

Here \( \mathcal{MN}(\Gamma; M, V, L) \) is a matrix normal density\(^1\) with mean matrix \( M \) and left and right covariances \( L^{-1} \) and \( V \), respectively; \( \mathcal{IW}(\Sigma; n, S) \) is an inverse Wishart (IW) density with \( n \) degrees of freedom and scale matrix \( S \). As pointed out above, the MNIW prior is a standard choice for a linear Gaussian model as in \((1a)\) (see e.g. West and Harrison (1997)). Furthermore, for suitably chosen hyperparameters (i.e. \( M, L, n_0 \) and \( S_0 \)), the

\(^1\)If \( \Gamma \sim \mathcal{MN}(M, V, L) \), then \( \text{vec}(\Gamma) \sim \mathcal{N}(\text{vec}(M), L^{-1} \otimes V) \).
effects of this prior on the posterior density will be minor. For a discussion on how to
choose the hyperparameters, see Appendix A. Similarly, we put a conjugate IW prior on
\( r \) (the univariate IW distribution is also known as inverse Gamma), according to,

\[
p(r) = \text{IW}(r; m_0, R_0).
\]

2.2 Alt. II – Sparsity-promoting prior

It is also possible to do automatic order selection via the use of an over-parameterized
model, which is then regulated by some sparsity-promoting mechanism. For optimization-
based methods, it is common to use \( \ell_1 \)-regularization to control sparsity. This gives rise to
well known methodologies such as the lasso (Tibshirani, 1996) and compressed sensing
(Donoho, 2006). In the Bayesian setting, the analogue of sparsity regularization is to use
sparsity-promoting priors. As an example, lasso can be interpreted as the Bayes posterior
mode under the Laplace prior (Park and Casella, 2008).

A general class of sparsity-promoting priors are the generalized hyperbolic (GH) distri-
butions (Barndorff-Nielsen and Shephard, 2001). This class contains many distributions
that have been successfully used to control sparsity in different settings; examples include
the Laplace (Park and Casella, 2008), normal inverse-Gaussian (Caron and Doucet, 2008)
and Student’s \( t \) (Tipping, 2001). Automatic relevance determination (ARD) by MacKay
(1994); Neal (1996), which is a popular Bayesian approach to automatic order selection,
is also based on a special type of GH prior. The class of GH distributions has recently
been extended to a dynamic setting by Caron et al. (2012).

We use a hierarchical representation of the GH distribution. If \( x \sim \mathcal{N}(0, \tau) \) with \( \tau \sim
\mathcal{GI}_G(\nu, a, b) \), then it holds that \( x \) is distributed according to the (zero-mean) GH distribu-
tion with parameters \( \nu, a \) and \( b \). Here, \( \mathcal{GI}_G \) is the generalized inverse-Gaussian (GIG)
distribution with density,

\[
\frac{(a/b)^{\nu/2}}{2K_{\nu}(\sqrt{ab})} \tau^{\nu-1} \exp \left( -\frac{1}{2} \left( a\tau + b/\tau^{-1} \right) \right), \tag{4}
\]

where \( K_{\nu} \) is a modified Bessel function of the second kind. The distribution is defined
for \( a \geq 0, b \geq 0 \) and \( \nu \in \mathbb{R} \). For \( a = 0 \) or \( b = 0 \), the normalization constant must be
interpreted in a limiting sense.

To make use of sparsity-promoting GH priors for automatic order selection in state-space
models, we use a multivariate generalization of the GH distribution. The prior is defined
by placing independent, zero-mean Gaussian priors on the columns \( \{\gamma_j\}_{j=1}^{n_x+n_u} \) of the
matrix \( \Gamma \),

\[
p(\Gamma | \bar{\tau}) = \prod_{j=1}^{n_x+n_u} \mathcal{N}(\gamma_j; 0, \tau_j I_d), \tag{5}
\]

where \( I_d \) is a \( d \times d \) identity matrix and \( \bar{\tau} = \{\tau_j\}_{j=1}^{n_x+n_u} \) are hyperparameters governing
the variances of each of the columns. These are assigned independent GIG priors,

\[
p(\tau_j) = \mathcal{GI}_G(\tau_j; \nu, a, b), \tag{6}
\]
for \( j = 1, \ldots, n_x + n_u \). A similar construction has previously been used by Fox et al. (2011), for the special case of ARD, to automatically identify the order of a state-space model.

The resulting marginal distributions of the columns of \( \Gamma \) will have distinct peaks at the origin. Hence, if there is not enough evidence for the \( j \)th state/input component to be non-zero, the corresponding variance parameter \( \tau_j \) will decrease toward zero. This will in turn drive the \( j \)th column of \( \Gamma \) to zero. Contrary to the MNIW prior, which in general will lead to a full \( \Gamma \)-matrix, the GH prior will thus result in a \( \Gamma \)-matrix with a sparse column-pattern. As a result, the corresponding state components will be unobservable and they can be discarded from the model. We emphasize that these unobservable modes are inherent to the model and they should not be thought of as representing unobservable modes of the true system (which is assumed to be observable).

In summary, if a suitable model order is not known beforehand, the GH prior can thus be used for automatic order determination (as well as input selection). This is done by overparameterizing the model and letting the GH prior switch irrelevant model components off.

The process noise and measurement noise variances are given the same IW priors as in the MNIW case, i.e. \( p(Q) \) is defined according to (2b) and \( p(r) \) according to (3).

2.3 Gaussian process prior

For the nonlinear mapping we develop a nonparametric model by placing a GP prior on \( h \),
\[
    h(\cdot) \sim GP(m(z), k_\eta(z, z')).
\] (7)

See Rasmussen and Williams (2006) for a thorough introduction to GPs. The GP is governed by a mean function \( m(z) \) and a covariance function (also referred to as a kernel) \( k_\eta(z, z') \). We use a linear mean function \( m(z) = z \), i.e. the prior is that no nonlinearity is present. However, any alternative mean function can be used if desired.

The covariance function can be taken as any positive definite kernel. Standard choices in the GP regression literature are the squared exponential kernel, the Matérn class of kernels and the rational quadratic kernel. See (Rasmussen and Williams, 2006, Chapter 4) for further details and additional examples. The covariance function is (typically) parameterized by some hyperparameter \( \eta \), determining for instance its amplitude and length-scale. The hyperparameter is inferred from data alongside the system parameters. To complete the model we place a prior \( p(\eta) \) on the hyperparameter, depending on the choice of kernel.

Note that, due to the nonparametric nature of the GP, the proposed model is flexible and can describe a wide range of nonlinear mappings. We do not assume any specific form of \( h \). However, since we are dealing with data affected by stochastic disturbances, we will in general favor smooth regression functions to avoid over-fitting. Still, as we shall see in Section 6, the proposed method can perform well even when the true nonlinearity is non-differentiable.
3 Inference via particle Gibbs sampling

Assume that we have observed a batch of input/output data. Let \( \Pi \triangleq \{ \Gamma, Q, r \} \) (for the MNIW prior), or \( \Pi \triangleq \{ \Gamma, Q, r, \bar{\tau} \} \) (for the GH prior) denote the system parameters. The task at hand is to identify the unknown quantities of the model, i.e. the parameters \( \Pi \), the hyperparameter \( \eta \) and the nonlinear mapping \( h(\cdot) \). Let us introduce the augmented parameter \( \theta \triangleq \{ \Pi, \eta, h(\cdot) \} \in \Theta \triangleq S \times F \), where \( S \) is a finite-dimensional space (containing \( \Pi \) and \( \eta \)) and \( F \) is an appropriate function space. Note that we use the term “parameter” to refer to \( \theta \), which also includes the nonparametric part of the model, \( h \). We then seek the posterior density of \( \theta \) given the observations \( y_{1:T} \). More generally, we compute the joint posterior density of the parameter and the system states \( x_{1:T} \), i.e.

\[
p(\theta, x_{1:T} | y_{1:T}) = p(x_{1:T} | \theta, y_{1:T}) p(\theta | y_{1:T}). \tag{8}
\]

The density \( p(\theta | y_{1:T}) \) is obtained by straightforward marginalization of (8). Here, and throughout this paper, conditioning on the inputs \( u_{1:T} \) is implicit.

The posterior density (8) is analytically intractable and we shall make use of an MCMC sampler to address the inference problem. In Section 3.1 below we outline the solution offered by a standard Gibbs sampler and point out a fundamental problem with this approach. This problem is then solved by introducing the particle Gibbs sampler in Section 3.2.

3.1 Ideal Gibbs sampling

A Gibbs sampler is an MCMC method which targets some joint density by alternately sampling from its conditionals (Robert and Casella, 2004). For the problem under study, we suggest to use a multi-stage Gibbs sampler, targeting (8) by iterating the following steps:

\[
\begin{align*}
\text{Draw } \Pi^* | h, x_{1:T}, y_{1:T}; \tag{9a} \\
\text{Draw } \eta^* | \Pi^*, x_{1:T}, y_{1:T}; \tag{9b} \\
\text{Draw } h^* | \eta^*, \Pi^*, x_{1:T}, y_{1:T}; \tag{9c} \\
\text{Draw } x^*_{1:T} | \theta^* = \{ \Pi^*, \eta^*, h^* \}, y_{1:T}. \tag{9d}
\end{align*}
\]

These four steps represent the basic splitting of the variables used in the Gibbs sampler. For the conjugate MNIW prior, the posterior distribution for the system parameters in (9a) is available in closed form. For the GH prior, we need to divide step (9a) into further substeps. We return to this in Section 4.2. Note that the system parameters \( \Pi \) are conditionally independent of the hyperparameter \( \eta \). Step (9b) is partially collapsed (we do not condition on \( h \) when sampling \( \eta \)). When possible, collapsing is beneficial since it allows larger updates of the involved variables and it will thus improve the mixing of the chain. For this reason, it is a standard procedure, frequently used in Gibbs sampling; see e.g. Dyk and Park (2008) and (Liu, 2001, Sec. 6.7).

Unfortunately, step (9d) of this Gibbs sweep is still problematic. Sampling from the exact posterior is not possible since the joint smoothing density \( p(x_{1:T} | \theta, y_{1:T}) \) is not available in closed form. In other words, the state inference problem is intractable, even if we fix the parameters of the model, due to the presence of the nonlinearity. Neither is it easy
to construct a good proposal kernel for a Metropolis-Hastings (MH) sampler, due to the high dimension of $x_{1:T}$ (for large $T$). However, it is possible to address this problem by exploiting a powerful statistical inference tool, recently introduced by Andrieu et al. (2010), known as particle MCMC (PMCMC).

### 3.2 Particle Gibbs sampling

A thorough treatment of PMCMC is well beyond the scope of this paper and we refer the interested reader to Andrieu et al. (2010); Andrieu and Roberts (2009); Pitt et al. (2012); Lindsten et al. (2012a); Whiteley et al. (2010); Lindsten and Schön (2012). However, in this section we briefly introduce the particular PMCMC method that we have employed in this work. It is a version of the particle Gibbs (PG) sampler that we refer to as $PG$ with ancestor sampling (PGAS) (Lindsten et al., 2012a). It is worth emphasizing that from a practitioner’s point of view, it is not necessary to understand all the technical details of PMCMC to be able to use it as a component in a composite identification procedure. Whenever we are faced with the problem of sampling from an intractable joint smoothing density, such as $p(x_{1:T} \mid \theta, y_{1:T})$, PMCMC can be used as a substitute for an exact sample, without introducing any systematic error.

The basic idea underlying PMCMC is to use a particle filter (PF) to construct a Markov kernel leaving the exact joint smoothing distribution invariant. This Markov kernel can then be used as a component of an MCMC sampler, e.g. the multi-stage Gibbs sampler given by (9). We thus seek a family of Markov kernels on $X_T$,

$$\{M_\theta : \theta \in \Theta\}, \quad (10)$$

such that, for each $\theta$, $M_\theta(x_{1:T} \mid x'_{1:T})$ leaves the joint smoothing density $p(x_{1:T} \mid \theta, y_{1:T})$ invariant. In PGAS, these kernels are constructed using a procedure referred to as a conditional particle filter with ancestor sampling (CPF-AS). Other options are available, e.g. to use the original CPF by Andrieu et al. (2010) or the CPF with backward simulation by Whiteley (2010); Lindsten and Schön (2012). However, we focus on CPF-AS since ancestor sampling has been found to considerably improve the mixing over the basic CPF, it can be implemented in a forward only recursion and its computational cost is linear in the number of particles.

CPF-AS is a sequential Monte Carlo sampler, similar to a standard PF, but with the important difference that one particle at each time step is specified $a priori$. Let these particles be denoted as $x'_{1:T} = \{x'_1, \ldots, x'_T\}$. The method is most easily described as an auxiliary PF; see Doucet and Johansen (2011); Gustafsson (2010); Pitt and Shephard (1999) for an introduction. As in a standard auxiliary PF, the sequence of joint smoothing densities $p(x_{1:t} \mid \theta, y_{1:t})$, for $t = 1, \ldots, T$, is approximated sequentially by collections of weighted particles. Let $\{x_{1:t-1}^i, w_{t-1}^i\}_{i=1}^N$ be a collection of weighted particles approximating $p(x_{1:t-1} \mid \theta, y_{1:t-1})$ by the empirical distribution,

$$\hat{p}^N(dx_{1:t-1} \mid \theta, y_{1:t-1}) \triangleq \sum_{i=1}^N w_{t-1}^i \delta_{x_{1:t-1}^i}(dx_{1:t-1}). \quad (11)$$

Here, $\delta_z(dx)$ is a point mass located at $z$. To propagate this sample to time $t$, we introduce the auxiliary variables $\{a_t^i\}_{i=1}^N$, referred to as ancestor indices. The variable $a_t^i$ is the
index of the ancestor particle at time $t - 1$, of particle $x_i^t$. Hence, $x_i^t$ is generated by first sampling the ancestor index with $\mathbb{P}(a_i^t = j) = w_{t-1}^j$. Then, $x_i^t$ is drawn from some proposal kernel,

$$x_i^t \sim q(x_t | \theta, x_{i-1}^{a_t^i}, y_t).$$

(12)

The particle trajectories are then augmented according to $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$. In the auxiliary PF formulation, the resampling step is implicit and corresponds to sampling the ancestor indices.

In a standard auxiliary PF, this procedure is repeated for each $i = 1, \ldots, N$, to generate $N$ particles at time $t$. In CPF-AS, however, we condition on the event that $x_i^t$ is contained in the collection $\{x_i^t\}_{i=1}^N$. To accomplish this, we sample according to (12) only for $i = 1, \ldots, N - 1$. The $N$th particle is then set deterministically: $x_N^t = x_t^t$.

To be able to construct the $N$th particle trajectory, the conditioned particle has to be associated with an ancestor at time $t - 1$. This is done by sampling a value for the corresponding ancestor index $a_i^N$ conditionally on $x_i^t$. From Bayes' theorem we have $p(x_{t-1} | \theta, x_t^i, y_{1:t}) \propto p(x_t^i | \theta, x_{t-1})p(x_{t-1} | \theta, y_{1:t-1})$. By plugging (11) into this expression, we arrive at the approximation,

$$\hat{p}^N(dx_{t-1} | \theta, x_t^i, y_{1:t}) = \sum_{i=1}^N w_{t-1|t}^i \delta_{x_{t-1}^i}(dx_{t-1})$$

(13)

with $w_{t-1|t}^i \propto w_{t-1}^j p(x_t^i | \theta, x_{t-1}^{a_t^i})$. To sample an ancestor particle for $x_t^i$, we draw from this empirical distribution. That is, we sample $a_i^N$ with $\mathbb{P}(a_i^N = j) = w_{t-1|t}^j$.

Finally, all the particles, for $i = 1, \ldots, N$, are assigned importance weights, analogously to a standard auxiliary PF. The CPF-AS is summarized in Algorithm 1. The transition and observation densities used to compute the importance weights are, for the model (1), given by,

$$p(x_{t+1} | \theta, x_t) = \mathcal{N}(x_{t+1}; Ax_t + Bu_t, Q),$$

(14a)

$$p(y_t | \theta, x_t) = \mathcal{N}(y_t; h(Cx_t), r).$$

(14b)

The conditioning on a prespecified collection of particles implies an invariance property of the CPF-AS, which is key to its applicability in an MCMC sampler. To state this more formally, we first make a standard assumption on the support of the proposal kernels used in the PF.

(A1) For any $\theta \in \Theta$ and $t = 1, \ldots, T$, $\mathcal{P}_t^\theta \subset \mathcal{Q}_t^\theta$ where,

$$\mathcal{P}_t^\theta = \{x_{1:t} : p(x_{1:t} | \theta, y_{1:t}) > 0\},$$

$$\mathcal{Q}_t^\theta = \{x_{1:t} : q(x_t | \theta, x_{t-1}, y_t)p(x_{1:t-1} | \theta, y_{1:t-1}) > 0\}.$$

The key property of CPF-AS can now be stated as follows.

**Proposition 1.** Assume (A1). Then, for any $\theta \in \Theta$ and any $N \geq 2$, the procedure

(i) run Algorithm 1 conditionally on $x_{1:T}^i$;

...
Algorithm 1 CPF-AS, conditioned on $x_{1:T}^i$

1. Draw $x_{1}^i \sim q(x_{1} \mid \theta, y_{1})$ for $i = 1, \ldots, N - 1$.
2. Set $x_{1}^{N} = x_{1}^i$.
3. For $i = 1, \ldots, N$, set
   \[
   w_{i}^i \propto \frac{p(y_{1} \mid \theta, x_{1}^i)p(x_{1}^i \mid \theta, y_{1})}{q(x_{1}^i \mid \theta, y_{1})},
   \]
   where the weights are normalized to sum to 1.
4. for $t = 2$ to $T$ do
5. Draw $a_{t}^i$ with $\mathbb{P}(a_{t}^i = j) = w_{t-1}^j$ for $i = 1, \ldots, N - 1$.
6. Draw $x_{t}^i \sim q(x_{t} \mid \theta, x_{t-1}^i, y_{t})$ for $i = 1, \ldots, N - 1$.
7. Draw $a_{t}^{N}$ with $\mathbb{P}(a_{t}^{N} = j) \propto w_{t-1}^j p(x_{t}^i \mid \theta, x_{t-1}^j)$.
8. Set $x_{t}^{N} = x_{t}^i$.
9. Set $x_{1:t} = \{x_{1:t-1}^{a_{i}^t}, x_{t}^i\}$ for $i = 1, \ldots, N$.
10. For $i = 1, \ldots, N$, set
    \[
    w_{t}^i \propto \frac{p(y_{t} \mid \theta, x_{t}^i)p(x_{t}^i \mid \theta, x_{t-1}^{a_{t}^i})}{q(x_{t}^i \mid \theta, x_{t-1}^{a_{t}^i}, y_{t})},
    \]
    where the weights are normalized to sum to 1.
11. end for

(ii) sample $x_{1:T}^*$ with $\mathbb{P}(x_{1:T}^* = x_{1:T}^i) = w_{T}^i$;

defines an irreducible and aperiodic Markov kernel $M_{\theta}^N$ on $X_T$, with invariant distribution $p(x_{1:T} \mid \theta, y_{1:T})$.

Proof: The invariance property follows by the construction of the CPF-AS by Lindsten et al. (2012a), and the fact that the law of $x_{1:T}^*$ is independent of permutations of the particle indices. This allows us to always place the conditioned particles at the $N$th position. Irreducibility and aperiodicity follows from (Andrieu et al., 2010, Theorem 5).

Consequently, if $x_{1:T}^i \sim p(x_{1:T} \mid \theta, y_{1:T})$ and we sample $x_{1:T}^*$ according to the procedure given in Proposition 1, then, for any number of particles $N$, it holds that $x_{1:T}^* \sim p(x_{1:T} \mid \theta, y_{1:T})$. For $N = 1$ we get, by construction, $x_{1:T}^* = x_{1:T}^i$, i.e. the trajectories are perfectly correlated (this is why we need $N \geq 2$ to get an irreducible kernel). As $N \to \infty$, on the other hand, the conditioning will have a negligible effect on the CPF-AS and $x_{1:T}^i$ will be effectively independent of $x_{1:T}^i$. Hence, the number of particles $N$ will affect the mixing of the Markov kernel $M_{\theta}^N$. The invariance property of the kernel holds for any $N$, but the larger we take $N$, the smaller the correlation will be between $x_{1:T}^*$ and $x_{1:T}^i$. However, it has been experienced in practice that the correlation drops off very quickly as $N$ increases (Lindsten et al., 2012a; Lindsten and Schön, 2012), and for many models a moderate $N$ (e.g. in the range 5–20) is enough to obtain a rapidly mixing kernel.
4 Posterior parameter distributions

We now turn our attention to steps (9a)–(9c) of the Gibbs sampler. That is, we assume that a fixed state trajectory $x_{1:T}$ is given and consider the problem of sampling from the posterior parameter distributions. Conditioned on $x_{1:T}$, the variables $\{\Gamma, Q, \bar{\tau}\}$ are independent of $\{h(\cdot), \eta, r\}$. Furthermore, $\{\Gamma, Q, \bar{\tau}\}$ are conditionally independent of $y_{1:T}$. Hence, the densities of the conditional variables appearing in (9a)–(9c) can be written as

$$p(\Pi \mid h, x_{1:T}, y_{1:T}) = p(\Gamma, Q, \bar{\tau} \mid x_{1:T})p(r \mid h, x_{1:T}, y_{1:T}),$$  \hspace{1cm} (15a)

$$p(\eta \mid \Pi, x_{1:T}, y_{1:T}) = p(\eta \mid r, x_{1:T}, y_{1:T}),$$  \hspace{1cm} (15b)

$$p(h \mid \eta, \Pi, x_{1:T}, y_{1:T}) = p(h \mid \eta, r, x_{1:T}, y_{1:T}).$$  \hspace{1cm} (15c)

For the MNIW prior, the variable $\bar{\tau}$ is not present. The factorization of the posterior in (15a) suggests that sampling from this distribution can be done in two decoupled steps. In the subsequent sections, we derive expressions for the PDFs appearing on the right hand sides of (15).

4.1 MNIW prior – Posterior of $\Gamma$ and $Q$

For the MNIW prior, the posterior density of $\{\Gamma, Q\}$ is available in closed form and is given as follows. Let,

$$X = [x_2 \ldots x_T], \quad W = [w_1 \ldots w_{T-1}], \quad \bar{X} = [x_1 \ldots x_{T-1}].$$

It follows from (1a) that $p(x_{1:T} \mid \Gamma, Q)$ can be described in terms of the relation

$$X = \Gamma \bar{X} + W.$$  \hspace{1cm} (16)

The prior (2) is conjugate to this likelihood model and it follows (West and Harrison, 1997) that the posterior parameter distribution is MNIW and given by

$$p(\Gamma, Q \mid x_{1:T}) = \mathcal{MN}(\Gamma; S_{XX} S_{\bar{X}X}^{-1}, Q, S_{X\bar{X}}) \mathcal{IW}(Q; T - 1 + n_0, S_{X|X} + S_0),$$  \hspace{1cm} (17a)

with

$$S_{XX} = \bar{X} \bar{X}^T + L,$$  \hspace{1cm} (17b)

$$S_{X\bar{X}} = X \bar{X}^T + ML,$$  \hspace{1cm} (17c)

$$S_{XX} = XX^T + MLM^T,$$  \hspace{1cm} (17d)

$$S_{X|\bar{X}} = S_{XX} - S_{X\bar{X}} S_{XX}^{-1} S_{X\bar{X}}^T.$$  \hspace{1cm} (17e)

4.2 GH prior – Posterior of $\Gamma$, $Q$ and $\bar{\tau}$

If we instead use the GH prior for the system matrix $\Gamma$, there is no closed-form expression for the posterior density of $\{\Gamma, Q, \bar{\tau}\}$. To get around this, we split the sampling of these
variables (in step (9a)) into sub-steps according to,
\[ \Gamma^* \sim p(\Gamma \mid Q, \bar{\tau}, x_{1:T}), \]
\[ Q^* \sim p(Q \mid \Gamma^*, x_{1:T}), \]
\[ \bar{\tau}^* \sim p(\bar{\tau} \mid \Gamma^*). \]  

To find the posterior of $\Gamma$, we note that (1a) can be written
\[ x_{t+1} = \left[ \bar{x}_{t,1}I_{n_x} \ldots \bar{x}_{t,n_x+n_u}I_{n_x} \right] \text{vec}(\Gamma) + w_t, \] 
where $\bar{x}_t = [x_t^T u_t^T]^T$ and vec(·) is the vectorization operator, which stacks the columns of a matrix into a vector. Hence, we may write (16) as,
\[ \text{vec}(X) = (\bar{X}^T \otimes I_{n_x}) \text{vec}(\Gamma) + \text{vec}(W), \] 
where $\otimes$ is the Kronecker product. Together with the prior (5) this yields the posterior of $\Gamma$ as,
\[ p(\Gamma \mid Q, \bar{\tau}, x_{1:T}) = \mathcal{N}(\text{vec}(\Gamma); \mu_\Gamma, \Sigma_\Gamma), \] 
with
\[ \mu_\Gamma = \Sigma_\Gamma (\bar{X} \otimes Q^{-1}) \text{vec}(X), \]
\[ \Sigma_\Gamma = (\text{diag}(\bar{\tau})^{-1} \otimes I_{n_x} + (\bar{X} \otimes Q^{-1})\Psi)^{-1}, \]
where diag(ν) is a diagonal matrix with the elements of the vector ν on the diagonal.

For the posterior of $Q$, the IW prior (2b) is conjugate to the likelihood defined by (16) (now with $\Gamma$ considered fixed). Hence, the posterior is given by an IW distribution according to
\[ p(Q \mid \Gamma, x_{1:T}) = \mathcal{IW}(Q; T - 1 + n_0, S_{GH} + S_0), \] 
with $S_{GH} = (X - \Gamma \bar{X})(X - \Gamma \bar{X})^T$ (cf. with (17a)).

Finally, for the variance parameters of the GH prior, the (independent) Gaussian likelihoods given by (5) are conjugate to the GIG priors (6). We have,
\[ p(\tau_j \mid \gamma_j) \propto p(\gamma_j \mid \tau_j)p(\tau_j) \]
\[ \propto \tau_j^{-\frac{a+\nu}{2}} \exp\left(-\frac{1}{2\tau_j \gamma_j^T \gamma_j}\right)\tau_j^{\nu-1} \exp\left(-\frac{1}{2} (a\tau_j + b\tau_j^{-1})\right). \] 

It follows that,
\[ p(\bar{\tau} \mid \Gamma) = \prod_{j=1}^{n_x+n_u} \mathcal{GIG} \left( \tau_j; \nu - \frac{n_x}{2}, a, b + \gamma_j^T \gamma_j \right), \] 
where we recall that $\{\gamma_j\}_{j=1}^{n_x+n_u}$ are the columns of the matrix $\Gamma$. 

4.3 Posterior of $r$

For fixed $x_{1:T}$ and $h(\cdot)$, let $\mathbf{h} = (h(Cx_1) \cdots h(Cx_T))^T$ and $\mathbf{y} = (y_1 \cdots y_T)^T$ be the vectors of function outputs and observations, respectively. Furthermore, let $\mathbf{e} = (e_1 \cdots e_T)^T$. It then follows from (1c) that the likelihood $p(y_{1:T} \mid r, h, x_{1:T})$ can be described in terms of the relation $\mathbf{y} = \mathbf{h} + \mathbf{e}$. The prior $p(r \mid h, x_{1:T}) = p(r)$ given in (3) is conjugate to this likelihood model and it follows that the posterior parameter distribution is IW and given by,

$$p(r \mid h, x_{1:T}, y_{1:T}) = IW(r; T + m_0, S_r + R_0),$$

with $S_r = (\mathbf{y} - \mathbf{h})^T(\mathbf{y} - \mathbf{h})$.

4.4 Posterior of $h(\cdot)$

The GP prior (7) is conjugate to the likelihood model given by (1c). Hence, the posterior distribution of $h(\cdot)$ given $r, x_{1:T}$ and $y_{1:T}$ is a GP. Sampling from this posterior distribution thus involves drawing a sample path from the posterior stochastic process. When it comes to implementing a Gibbs sampler containing such a GP posterior, a problem that we need to address is how to represent this sample path.

Here, we present two alternative approaches. The first, and most proper, solution is to sample from the GP whenever an evaluation of the function $h$ is needed in the algorithm. This will be done for $N$ query points for each time $t = 1, \ldots, T$, where $N$ is the number of particles used in the PGAS sampler (see Section 3.2). The second alternative is a simpler approach, namely to evaluate the GP on a fixed grid of points. This is done once for each iteration of the MCMC sampler. When evaluating the function $h$ in the PF, we do a linear interpolation between the grid points. This approximate solution is the approach that we have employed in the numerical examples presented in Section 6.

In either approach, let $\mathbf{z}_* = (z^{(1)} \cdots z^{(M)})^T$ be the points for which we wish to evaluate the GP (these can either be random points generated in the PF or fixed grid points). Furthermore, let $\mathbf{h}_* = (h(z^{(1)}) \cdots h(z^{(M)}))^T$. It then follows (see (Rasmussen and Williams, 2006, Section 2.2)) that the posterior distribution of $\mathbf{h}_*$ is given by

$$p(h_* \mid \eta, r, x_{1:T}, y_{1:T}) = N(h_*; \mu_*, \Sigma_*),$$

where

$$\mu_* = \mathbf{m}_* + P_*^T(P + rI_T)^{-1}(\mathbf{y} - \mathbf{m}),$$

$$\Sigma_* = P_* - P_*^T(P + rI_T)^{-1}P_*.$$

Here, we have introduced the notation

$$\mathbf{m}_* = (m(z^{(1)}) \cdots m(z^{(M)}))^T,$$

$$\mathbf{m} = (m(z_1) \cdots m(z_T))^T.$$
5 Convergence analysis

and the matrices \( P, P_\star \) and \( P_{\star\star} \) are given by,
\[
[P]_{ij} = k_\eta(z_i, z_j), \quad i, j = 1, \ldots, T, \tag{27c}
\]
\[
[P_\star]_{ij} = k_\eta(z_i, z(j)), \quad i = 1, \ldots, T, \quad j = 1, \ldots, M, \tag{27d}
\]
\[
[P_{\star\star}]_{ij} = k_\eta(z(i), z(j)), \quad i, j = 1, \ldots, M. \tag{27e}
\]

Using the expressions above, we can generate a sample of \( \mathbf{h}_\star \) from the posterior distribution (26).

It should be noted that the computational complexity of evaluating and sampling from a posterior GP is cubic in the number of query points as well as in the number of data points, i.e. of order \( O(M^3 + T^3) \). Hence, the cost of sampling from the GP can be prohibitive when \( T \) is large. However, there exist several methods in the literature, dedicated to enabling GP regression for large datasets, e.g. based on low-rank approximations; see (Rasmussen and Williams, 2006, Chapter 8) and the references therein. In this work we have not resorted to such techniques.

4.5 Posterior of \( \eta \)

Finally, we need to sample from the posterior of the hyperparameters of the GP kernel in (9b). Due to the often intricate dependence of the covariance kernel on \( \eta \), it is in general not possible to find a closed-form expression for this posterior. Instead, we apply an MH accept/reject step to sample \( \eta \). That is, we sample a value from some proposal kernel \( \eta' \sim \nu(\eta' \mid \eta) \) (in this work we use a Gaussian random walk). The proposed sample is then accepted with probability
\[
1 \wedge \frac{p(y_{1:T} \mid \eta', r, x_{1:T}) p(\eta') \nu(\eta \mid \eta')}{p(y_{1:T} \mid \eta, r, x_{1:T}) p(\eta) \nu(\eta' \mid \eta)}, \tag{28}
\]
otherwise the previous value is kept. Let \( P_\eta \) be given as in (27c), but where we now emphasize the dependence on \( \eta \) in the notation. We then have
\[
p(y_{1:T} \mid \eta, r, x_{1:T}) = \mathcal{N}(y; \mathbf{m}, P_\eta + rI_T). \tag{29}
\]

To compute the acceptance probability in (28) in a numerically robust way, we make a Cholesky factorization of the covariance \( P_\eta + rI_T = R_\eta^T R_\eta \). By straightforward manipulations it then follows that
\[
\frac{p(y_{1:T} \mid \eta', r, x_{1:T})}{p(y_{1:T} \mid \eta, r, x_{1:T})} = \frac{\det(R_\eta)}{\det(R_{\eta'})} e^{(\frac{1}{2} s_\eta^T s_\eta - \frac{1}{2} s_{\eta'}^T s_{\eta'})}, \tag{30}
\]
where \( s_\eta = R_\eta^T (y - \mathbf{m}) \), which can be computed efficiently due to the triangularity of \( R_\eta \).

5 Convergence analysis

The proposed identification procedure is summarized in Algorithms 2–4. In this section we study the convergence properties of the method. Let \( \pi_T \) be the target distribution for the MCMC sampler, i.e. the distribution with density \( p(\theta, x_{1:T} \mid y_{1:T}) \). First, we provide a result stating that, for a fixed data record of length \( T \), the empirical distribution of the
Algorithm 2 Wiener system identification using PGAS

1: Set $A[0] = M$, $Q[0] = S_0$, $r[0] = R_0$ and $h_*[0] = z_*$. 
2: Set $r_1 : T[0]$ and (for GH prior) $\bar{\tau}[0]$ arbitrarily. 
3: for $k \geq 1$ do 
4: Run Algorithm 3 (for MNIW prior) or Algorithm 4 (for GH prior) to sample $\Pi[k]$. 
5: Sample $\eta[k]$ given $r[k]$, $x_1 : T[k-1]$ and $y_1 : T$ using an MH step as described in Section 4.5. 
6: Sample $h_*[k] \sim p(h_* \mid \eta[k], r[k], x_1 : T[k-1], y_1 : T)$ according to (26). 
7: Set $\theta[k] = \{\Pi[k], \eta[k], h_*[k]\}$. 
8: Run Algorithm 1, targeting $p(x_1 : T \mid \theta[k], y_1 : T)$, conditioned on $x_1 : T[k-1]$. 
9: Sample $J$ with $P(J = i) = w_i^T$. Set $x_1 : T[k] = x_1'[j]$. 
10: end for

Algorithm 3 Sampling system parameters for MNIW prior

1: Using (17) and (25), respectively, 
   (a) Sample $\{\Gamma[k], Q[k]\} \sim p(\Gamma, Q \mid x_1 : T[k-1])$. 
   (b) Sample $r[k] \sim p(r \mid h_*[k-1], x_1 : T[k-1], y_1 : T)$. 
2: Return $\Pi[k] = \{\Gamma[k], Q[k], r[k]\}$.

generated Markov chain approaches $\pi_T$ as the number of iterations $k \to \infty$. Second, we study the consistency of the Bayes estimator as the number of data points tend to infinity. We assess that the Bayes estimator is almost surely (a.s.) consistent w.r.t. the prior, for any identifiable functional.

5.1 Convergence of the Markov chain

Due to the invariance property of Proposition 1, the PGAS sampler can be treated as a regular MCMC sampler, and standard convergence analysis applies (see e.g. Tierney (1994); Robert and Casella (2004); Meyn and Tweedie (2009)). We start by analyzing the ideal Gibbs sampler, defined by running Algorithm 2 but replacing Steps 2(e)–(f) by a draw from the exact joint smoothing density, i.e. $x_1 : T[k] \sim p(x_1 : T \mid \theta[k], y_1 : T)$. This procedure cannot be implemented in practice, but it is useful to consider it as an intermediate step in the analysis of the PGAS sampler.

Lemma 1. The ideal Gibbs sampler has invariant distribution $\pi_T$.

Proof: The ideal Gibbs sampler is a cyclic MCMC sampler according to (9). Steps (9a), (9c) and (9d) are implemented as standard Gibbs steps. Step (9b) is implemented as an MH step. This hybrid scheme does not invalidate the MCMC kernel (Tierney, 1994). Collapsing is done over $h$ in step (9b). Since this is done prior to sampling $h$ in step (9c), the Gibbs sampler is properly collapsed (Dyk and Park, 2008). Consequently, each step of the sampler leaves $\pi_T$ invariant.

We thus know that $\pi_T$ is a possible equilibrium distribution of the ideal Gibbs sampler. To assess convergence, i.e. to show that a Markov chain generated by the sampler indeed
Algorithm 4 Sampling system parameters for GH prior

1: Using (21), (22), (24) and (25), respectively,
(a) Sample \( \Gamma[k] \sim p(\Gamma \mid Q[k-1], \bar{\tau}[k-1], x_{1:T}[k-1]) \).
(b) Sample \( Q[k] \sim p(Q \mid \Gamma[k], x_{1:T}[k-1]) \).
(c) Sample \( \bar{\tau}[k] \sim p(\bar{\tau} \mid \Gamma[k]) \).
(d) Sample \( r[k] \sim p(r \mid h_x[k-1], x_{1:T}[k-1], y_{1:T}) \).
2: Return \( \Pi[k] = \{ \Gamma[k], Q[k], r[k], \bar{\tau}[k] \} \).

will approach \( \pi_T \), we need to show that it is irreducible and aperiodic. For simplicity, we make the following assumption on the proposal kernel that is used in the inherent MH step for the hyperparameter \( \eta \).

**A2** \( \upsilon(\eta \mid \eta') > 0 \) for any \( (\eta, \eta') \) with \( p(\eta) > 0 \).

The assumption is trivially satisfied if the proposal kernel is taken as, for instance, a Gaussian random walk.

**Lemma 2.** Assume (A2). Then, the ideal Gibbs sampler is irreducible and aperiodic.

**Proof:** For a probability density \( p(x) \), let \( \text{supp}[p(x)] = \{ x : p(x) > 0 \} \) be its support. The target density can be expressed as

\[
p(\theta, x_{1:T} \mid y_{1:T}) \propto p(y_{1:T} \mid \theta, x_{1:T})p(x_{1:T} \mid \theta)p(\theta).
\]

The first two factors are Gaussian and thus everywhere positive, i.e.

\[
\text{supp}[p(\theta, x_{1:T} \mid y_{1:T})] = \text{supp}[p(\theta)] \times X^T.
\]

From the definition of the model in Section 2,

\[
\text{supp}[p(\theta)] = \text{supp}[p(\Pi)] \times \text{supp}[p(\eta)] \times \text{supp}[p(h)].
\]

That is, by construction, the support of the target density is the Cartesian product of the supports of the marginals. Hence, the target density satisfies a positivity condition (Robert and Casella, 2004, Definition 9.4). Let \( K \) be the Markovian transition kernel of the ideal Gibbs sampler. From the positivity condition and assumption (A2) it follows that \( K(A \mid \{ \theta', x'_{1:T} \}) > 0 \) for any \( A \) with \( \pi_T(A) > 0 \). That is, any set with positive posterior probability is accessible in one step. Irreducibility and aperiodicity of the ideal Gibbs sampler follows.

Due to Proposition 1, the properties of the ideal Gibbs sampler carries over directly to the PGAS sampler. We can thus provide the following convergence result for the proposed identification algorithm.

**Theorem 1.** Assume (A1) and (A2). For any \( N \geq 2 \), the PGAS sampler defined by Algorithm 2 generates a sequence \( \{ \theta[k], x_{1:T}[k] \} \) whose distribution \( \mathcal{L}^N(\{ \theta[k], x_{1:T}[k] \} \in \cdot) \) satisfies

\[
\| \mathcal{L}^N(\{ \theta[k], x_{1:T}[k] \} \in \cdot) - \pi_T \|_{TV} \to 0
\]
as \( k \to \infty \) for almost all \((\pi_T)\) starting points, where \( \| \cdot \|_{TV} \) is the total variation norm.

**Proof:** From Proposition 1 and Lemma 1, \( \pi_T \) is an invariant distribution of the PGAS sampler. Due to the fact that the family of Markov kernels defined in Proposition 1 are irreducible and aperiodic, the results of Lemma 2 carry over to the PGAS sampler. Convergence in total variation follows from (Tierney, 1994, Theorem 1).

### 5.2 Consistency of the Bayes estimator

We now turn to consistency of estimators constructed from the posterior distribution \( p(\theta \mid y_{1:T}) \). We adapt the general result by Doob (1949) to our setting (see also Lijoi et al. (2004); Schwartz (1965)). This classical result gives almost sure consistency w.r.t. the prior. More recent developments have improved upon the classical results, avoiding the exceptional null set on which consistency may fail; see e.g. Choi and Schervish (2007); Barron et al. (1999). Other forms of posterior convergence have also been studied extensively; see e.g. Castillo (2012) for a Bernstein-von Mises theorem in the semiparametric setting and van der Vaart and van Zanten (2008) for contraction rates under Gaussian process priors.

Let \( \mathcal{Y} = \mathcal{Y} \times \mathcal{Y} \times \ldots \) be the infinite product space, \( \mathcal{U} \) be the smallest \( \sigma \)-algebra generated by all open subsets of \( \mathcal{Y} \), and let \( y = (y_1, y_2, \ldots) \in \mathcal{Y} \) be the infinite sequence of observations. Let \( \lambda \) be the prior distribution of \( \theta \) (i.e. the distribution with density \( p(\theta) \)) and let \( \{P_\theta : \theta \in \Theta\} \) be a family of distributions governing the law of \( y \) for each \( \theta \). We use the following identifiability criterion.

**Condition 1.** Let \((Z, \mathcal{Z})\) be a measurable space. The mapping \( \zeta : \Theta \mapsto Z \) is said to satisfy Condition 1 if it is measurable, \( L^1 \)-integrable and if there exists a \( \mathcal{U}/\mathcal{Z} \)-measurable function \( f : \mathcal{Y} \mapsto Z \) with \( f(y) = \zeta(\theta) \) a.s. \((P_\theta)\) for any \( \theta \in \Theta \).

**Theorem 2.** Let \( \zeta \) satisfy Condition 1, then the Bayes estimator of \( \zeta(\theta) \) is strongly consistent a.s. \((\lambda)\), i.e.

\[
P_\theta \left( \lim_{t \to \infty} \beta_t = \zeta(\theta) \right) = 1, \quad \text{a.s.} \ (\lambda),
\]

with \( \beta_t = \int \zeta(\theta)p(\theta \mid y_{1:t})d\theta \).

**Proof:** The proof follows (Doob, 1949; Schwartz, 1965). Take \( \Omega = \Theta \times \mathcal{Y} \) and let \( \mu = P_\theta \times \lambda \) be the kernel product measure. Let \( \gamma(\theta, y) = \zeta(\theta) \) and \( \beta_t(\theta, y) = \beta_t(y) = \mathbb{E}[\gamma \mid y_{1:t}] \). By the tower property of conditional expectation,

\[
\mathbb{E}[\beta_t \mid y_{1:t-1}] = \beta_{t-1}.
\]

Hence, \( \{\beta_t\} \) is a martingale sequence and the martingale convergence theorem (see e.g. (Williams, 1991, Theorem 14.2)) implies that \( \beta_t \to \mathbb{E}[\gamma \mid y] \) a.s. \((\mu)\). By Condition 1, \( \gamma \) is equivalent to a \( \mathcal{U} \)-measurable function a.s. \((\mu)\) and it follows that \( \mathbb{E}[\gamma \mid y] = \gamma \) a.s. \((\mu)\). Hence, \( 1 = \mu (\{ (\theta, y) : \beta_t(\theta, y) \to \gamma(\theta, y) \}) = \int P_\theta (\{ y : \beta_t(y) \to \zeta(\theta) \}) \lambda(d\theta) \). □
Remark 1. Condition 1 is an identifiability condition. Since the Wiener system is inherently unidentifiable (i.e. different \( \theta \) can give rise to the same input-output relation), we focus on a class of identifiable functionals \( \zeta \). If \( \zeta \) is such that, given an infinite amount of data, it can be uniquely determined (i.e. it is \( \mathcal{F} \)-measurable), then the Bayes estimator of \( \zeta \) is strongly consistent a.s. \( \lambda \). The identity function \( \zeta(\theta) = \theta \) is not contained in this class for a Wiener system, but that is not necessarily an issue. Indeed, for instance, the one-step predictor is generally identifiable, even when \( \theta \) is not. Hence, if the model is to be used for making predictions, the predictor offered by the Bayes estimator is consistent a.s. \( \lambda \).

6 Numerical illustrations

In this section we apply the proposed method to identify two synthetic Wiener systems. In both cases, a bootstrap CPF-AS with \( N = 15 \) particles is used in the PGAS sampler. We use a Matérn kernel for the GP, as recommended by Stein (1999),

\[
k(z, z') = \alpha \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\sqrt{2\nu \Delta z}}{\ell} \right)^\nu K_\nu \left( \frac{\sqrt{2\nu \Delta z}}{\ell} \right),
\]

with \( \Delta z = |z - z'| \). In (Rasmussen and Williams, 2006), it is recommended to set \( \nu = 3/2 \) or \( \nu = 5/2 \). Here, we use the latter, which gives a slightly smoother prior. The hyperparameters \( \alpha \) and \( \ell \) govern the amplitude and the length-scale of the kernel, respectively. We set \( \eta \triangleq \{ \log \alpha, \log \ell \} \) and place an improper flat prior on the hyperparameter \( p(\eta) \propto 1 \).

We compare the proposed algorithm with two standard methods from the literature: the semiparametric average derivative method (ADM) (Greblicki and Pawlak, 2008) and a fully parametric prediction-error method (PEM) (Ljung, 1999). ADM estimates a parametric FIR model of the linear block. We set the order \( p \) of the FIR model based on the true impulse responses so that any coefficient above \( p \) is smaller than \( 0.01 \) times the first coefficient. The FIR model is then transformed into an LTI model of the same order as the true system, using a balanced reduction. The nonlinear block is given by a nonparametric Nadaraya-Watson estimate. For PEM, we use an output-error model for the linear block, with the same order as the true system. The nonlinearity is parameterized differently in the two examples (see below).

We evaluate the estimates using \( H_2 \) and \( \ell_2 \) errors. Let \( \hat{G} \) and \( \hat{h} \) be the estimates of the transfer function \( G \) and the nonlinearity \( h \), respectively, for one of the methods. The aforementioned errors are then given by

\[
H_2 : \left( \frac{1}{2\pi} \int_{-\pi}^{\pi} |G(i\omega) - \hat{G}(i\omega)|^2 d\omega \right)^{1/2},
\]

\[
\ell_2 : \left( \frac{1}{z_+ - z_-} \int_{z_-}^{z_+} |h(z) - \hat{h}(z)|^2 dz \right)^{1/2}.
\]

In the latter expression, we integrate over a finite interval \([z_-, z_+]\) since the nonparametric estimates can only be computed over the range of the data.
6.1 6th-order system with saturation

Consider a 6th-order linear dynamical system according to (1) where the system matrices \((A, B, C)\) conform with the transfer function

\[
G(q) = \frac{c_1q^{-1} + \cdots + c_6q^{-6}}{1 + a_1q^{-1} + \cdots + a_6q^{-6}},
\]

with \(\bar{a} = (a_1, \ldots, a_6), \bar{c} = (c_1, \ldots, c_6)\) and

\[
\bar{a} = (-2.67, 2.96, -2.01, 0.914, -0.181, -0.0102),
\]

\[
\bar{c} = (-0.467, 1.12, -0.925, 0.308, -0.0364, 0.00110).
\]

The system is excited by a known input signal \(u_t\), which is taken as a realization of a white Gaussian noise with variance 1. The process noise and measurement noise (co)variances are given by \(Q = 0.5^2I_6\) and \(R = 0.1^2\), respectively. The nonlinear mapping \(h\) is given by a saturation,

\[
h(z) = \begin{cases} 
1 & \text{if } z \geq 0.5, \\
2z & \text{if } -0.5 \leq z < 0.5, \\
-1 & \text{if } z < -0.5.
\end{cases}
\]

We generate \(T = 1000\) samples from the system and apply the proposed method (Algorithm 2) for 20,000 MCMC iterations\(^2\) (out of which 10,000 iterations are considered as burnin). The model order is fixed to the true value \(n_x = 6\) and we thus use the MNIW prior. The hyperparameters are set as described in Appendix A.

We compare Algorithm 2 with ADM and PEM. For ADM, the order of the FIR model is set as described above, which gives \(p = 13\). For PEM, we use a 6th order output-error model for the linear block. For the nonlinearity, we exploit the knowledge that \(h\) is a saturation and parameterize the function accordingly.

The results are given in Figure 2, showing the Bode diagram of the linear system and the static nonlinearity. For comparison, to account for the inherent unidentifiability of the system, the estimates from all methods are rescaled so that the linear systems have the same \(H_2\)-norms. The shaded areas illustrate the 99% Bayesian credibility regions, computed from the posterior PDFs. In the legends of the figure we also report the \(H_2\) and the \(\ell_2\) errors, respectively, for each method.

All methods capture the main resonance peak of \(G\), but are less accurate at low frequencies (likely due to a lack of excitation). ADM results in a larger error than the other two methods, especially noticeable in the phase and in the estimate of the nonlinearity. Also for PEM, there is a quite large error in the estimate of \(h\), despite the fact that PEM uses a parametric function of the correct form. A possible reason for this is that PEM does not take the process noise into account, which results in biased estimates. The proposed method provides an accurate nonparametric estimate of the nonlinearity, despite the fact that \(h\) is non-differentiable and the GP is a smoothness prior. The uncertainty about the

\(^2\)For simplicity, we run the chain for a fixed number of iterations, chosen based on visual inspection of the trace plots. In practice, some convergence diagnostic could be used instead, e.g. the Raftery-Lewis test (Raftery and Lewis, 1992).
Figure 2: Bode diagram of the 6th-order linear system (left) and the nonlinear mapping (right) with estimates for ADM, PEM and the proposed method. The red line is the posterior mean of the Bode diagram and the shaded area is the 99 % Bayesian credibility interval. The integration interval for the $\ell_2$ error is $[z_-, z_+] = [-1.2, 1.2]$.

nonlinearity gets larger for $|z| \gtrsim 1.5$, reflecting the fact that there are few samples ($\approx 2 \%$) in these regions available in the process underlying the observed data.

6.2 4th-order system with non-monotone nonlinearity

To highlight the flexibility of the GP model we consider a model with a non-monotonic function $h$, shown in Figure 3. For this example, we use a 4th-order linear dynamical system with transfer function,

$$G(q) = \frac{c_1q^{-1} + \cdots + c_4q^{-4}}{1 + a_1q^{-1} + \cdots + a_4q^{-4}},$$

(35)

where $\bar{a} = (a_1, \ldots, a_4)$, $\bar{c} = (c_1, \ldots, c_4)$ and

$$\bar{a} = (1, 0.1, -0.49, 0.01), \quad \bar{c} = (0.368, 0.888, 0.524, 0.555).$$

The process noise and measurement noise (co)variances are given by $Q = 0.25^2I_4$ and $R = 0.1^2$, respectively. We excite the system by a white Gaussian input signal with variance $0.5^2$ and generate $T = 1000$ measurements. We apply the proposed identification method for 20,000 MCMC iterations (again, discarding 10,000 iterations as burnin). However, we now assume that the model order is unknown and that we wish to infer it alongside the parameters. Therefore, we employ the GH sparseness prior by overparameterizing the model and assuming a model order of $n_x = 10$. We use the specific choice $a = 0$ in (6). For this choice, the GIG distribution reduces to an inverse-Gamma distribution, which means that the GH prior corresponds to the so called automatic relevance determination (ARD) prior (MacKay, 1994; Neal, 1996).

Again, we compare the method with ADM and PEM. The order of the FIR model in ADM is set based on the true impulse response, as described above, resulting in $p = 97$. The FIR model is then reduced to an LTI system with the same order as the true system, $n_x = 4$. PEM uses a 4th order output-error model for the linear block and a piecewise
affine model (with 10 segments) for the nonlinearity.

Figure 3 shows the Bode diagram of the linear system and the static nonlinearity. The nonmonotonicity of \( h \) gives rise to an ambiguity of the value of \( z_t \) for a given observation \( y_t \). Basically, for any observation \( y_t \) in the range \([-0.3, 0.3]\) there are three possible values for \( z_t \) which describe the observation equally well statically. Despite this, the proposed method accurately captures the function \( h \), whereas both ADM and PEM fail in this respect. The linear system is also accurately estimated. Interestingly, PEM also finds a good model of \( G \), despite the poor estimate of \( h \).

To analyze the effect of the GH prior and the ability to automatically determine the model order, we provide box plots of the GH precisions \( \tau_{-1} \) for \( j = 1, \ldots, n_x + n_u \) over the 10 000 MCMC iterations (taken after burnin). These are given in Figure 4. Recall from (5) that a large value of \( \tau_{-1} \) implies that the \( j \)th column of \( \Gamma \) is pushed to zero and that the corresponding state component in effect is switched off. It is clear that the effective model order is indeed 4, as 6 of the precision parameters take on much larger values than the remaining ones. Note that the last column of \( \Gamma \), i.e. for \( j = 11 \), corresponds to the input signal \( u_t \). These results indicate that sparsity-promoting priors (such as ARD) can be useful for automatic order determination of state-space models. Note, however, that there is no guarantee that the correct model order is found and further evaluation is needed in order to assess the accuracy and the robustness of this approach.

6.3 Discussion

Compared to ADM and PEM, the new algorithm resulted in more accurate estimates, in particular of the nonlinearity \( h(\cdot) \). We believe that ADM and PEM both suffer from the facts that (i) the data are affected by process noise; (ii) only \( T = 1000 \) samples were used in the simulations. Our results suggest that the new method handles these difficulties better than the alternatives. It should be noted, however, that both ADM and PEM are considerably faster than the proposed method in terms of computation. For \( T \gg 1000 \),
the computational complexity of the proposed method may be prohibitive. Hence, we believe that the proposed method is of particular interest when data are scarce and/or noisy. However, it is also worth exploring parallel and distributed implementations of our algorithms; note in particular that particle filtering lends itself naturally to distribution across particles.

7 Conclusions and future work

We have presented a Bayesian semiparametric method for Wiener system identification, using a state-space representation of the linear dynamical system $G$ and a GP model for the static nonlinearity $h(\cdot)$. We considered two alternative priors for $G$; first, a conjugate prior which is applicable when the model order is fixed (which is the case if, for instance, the order is found by cross-validation); second, a sparsity-promoting prior which can be used to automatically determine the model order. This is done by over-parameterizing the model and switching unnecessary state components off.

The new algorithm was profiled on two examples with good results. Compared to existing methods, we believe that the algorithm is of particular interest when data are scarce and/or noisy. Indeed, a concern with the proposed method is that it does not scale well with the number of measurements $T$, since the computational complexity of evaluating the posterior GP is cubic in $T$. However, this is a fairly well-studied problem in the GP literature and existing approaches can be used to mitigate this issue. Alternatively, a different type of nonparametric regression function can be used, e.g. based on the Dirichlet process mixture of generalized linear models (Hannah et al., 2011).

We have found that sparsity-promoting priors can be useful for automatic order selection in state-space models. However, further evaluation is needed to determine the performance and the robustness of this approach. There are also alternative ways to do automatic order selection. For instance, reversible jump MCMC (Green, 1995) can be used to infer parameters in spaces of varying dimensions. We could thus use a reversible jump sampler to include the model order $n_x$ as a parameter of the model, and update the sizes

Figure 4: ARD precision parameters $\tau_j^{-1}$ for $j = 1, \ldots, 11$. The rightmost box plot corresponds to the input signal.
of the system matrices accordingly when the value of this parameter is changed. This re-
quires a way to incorporate reversible jump moves in PMCMC, which is a topic for future
work.

A different line of future work is to leave the class of Wiener systems and use PMCMC
for fully nonparametric identification of general nonlinear dynamical systems. This can,
for instance, be done by modeling both the dynamical equation and the measurement
equation of a state-space model as Gaussian processes.

Appendix

A Choosing the hyperparameters

We use an approach known as empirical Bayes, in which the observed data are used to
set the hyperparameters of the priors. For the MNIW prior, the following heuristic is
used. First, we run a subspace identification algorithm on the input/output data (see e.g.
Van Overschee and De Moor (1996)). The resulting model is transformed into observer
canonical form. We set the mean $M$ of the MN prior (2a) to the resulting $[A\ B]$-matrix.
The covariance $L^{-1}$ is set to identity. This choice allows for a considerable variability
around the mean. For the IW priors (2b) and (3) we use the same heuristic as Fox (2009,
p. 156–160), based on the empirical covariance of the observations $y_{1:T}$. For the ARD
prior, we instead need to set the hyperparameters $\nu$ and $b$ for the inverse-Gamma prior
governing the variance vector $\tilde{\tau}$. We follow Fox et al. (2011) and set $\nu = n_x$ and $b =
\nu \times 10^{-3}$. This choice fixes the prior mean to $10^{-3}$, encouraging a sparse solution, and
aims to provide a prior which is equally informative for different choices of $n_x$. 
Bibliography


Paper D

An efficient stochastic approximation EM algorithm using conditional particle filters

Authors: Fredrik Lindsten

Edited version of the paper:

An efficient stochastic approximation EM algorithm using conditional particle filters

Fredrik Lindsten
Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
lindsten@isy.liu.se

Abstract

I present a novel method for maximum likelihood parameter estimation in nonlinear/non-Gaussian state-space models. It is an expectation maximization (EM) like method, which uses sequential Monte Carlo (SMC) for the intermediate state inference problem. Contrary to existing SMC-based EM algorithms, however, it makes efficient use of the simulated particles through the use of particle Markov chain Monte Carlo (PMCMC) theory. More precisely, the proposed method combines the efficient conditional particle filter with ancestor sampling (CPF-AS) with the stochastic approximation EM (SAEM) algorithm. This results in a procedure which does not rely on asymptotics in the number of particles for convergence, meaning that the method is very computationally competitive. Indeed, the method is evaluated in a simulation study, using a small number of particles, with promising results.

1 Introduction

State-space models (SSMs) are commonly used in statistical signal processing to model dynamical systems. Methods such as sequential Monte Carlo (SMC), have emerged to allow inference beyond the linear Gaussian case (Doucet and Johansen, 2011; Cappé et al., 2005). However, estimation of fixed model parameters remains a challenging problem. We consider here a general, discrete-time SSM with state $x_t \in X$ and observation $y_t \in Y$, parameterized by some unknown parameter $\theta \in \Theta$,

$$x_{t+1} \sim f_\theta(x_{t+1} \mid x_t), \quad y_t \sim g_\theta(y_t \mid x_t).$$

We observe a batch of measurements $y_{1:T} = \{y_1, \ldots, y_T\}$ and seek to identify $\theta$ offline. Methods addressing this problem are often iterative, in the sense that they iterate between updating $\theta$ and updating/estimating the latent states $x_{1:T}$. Examples are the expectation maximization (EM) algorithm (Dempster et al., 1977) for maximum likelihood (ML) inference and Gibbs sampling (Geman and Geman, 1984) for Bayesian inference. SMC can naturally be used within these methods to address the intermediate state inference problem at each iteration. For instance, particle smoothing (PS) has been used within the
EM algorithm (PSEM) for challenging identification problems (Schön et al., 2011; Olsson et al., 2008; Cappé et al., 2005).

However, if used in a standard way, a large number of particles is typically required to obtain accurate state inference results. Since this has to be done at each iteration of the top level identification algorithm, the resulting method will be very computationally intensive. Recently, however, a framework for Bayesian inference referred to as particle Markov chain Monte Carlo (PMCMC) has been developed (Andrieu et al., 2010). PMCMC uses SMC within MCMC, but do so in a way which ensures that the methods are, in some sense, exact, for any number of particles (see Andrieu et al. (2010) for further discussion). Furthermore, a certain branch of PMCMC, based on so called conditional particle filters (CPFs), has been found to make very efficient use of the simulated particles (Lindsten et al., 2012; Lindsten and Schön, 2012; Whiteley et al., 2010). This is achieved by propagating information from one iteration to the next, by conditioning the PF on previously simulated particles.

The purpose of this contribution is to illustrate that these attractive methods are not exclusive to the Bayesian. Indeed, we develop a method for ML inference, i.e. the problem of finding $\hat{\theta}_{\text{ML}} = \arg\max_{\theta} p_\theta (y_{1:T})$. The method is a combination of stochastic approximation EM (SAEM) (Delyon et al., 1999) and the conditional PF with ancestor sampling (CPF-AS) (Lindsten et al., 2012). There have been previous contributions on combining PMCMC with SAEM (Andrieu and Vihola, 2011; Donnet and Samson, 2011). However, these methods differ from the present contribution, in that they are based on the particle independent Metropolis-Hastings kernel, which is not able to reuse information across iterations, as is done in CPF-AS.

## 2 The EM, MCEM and SAEM algorithms

To introduce the methods that we will be working with we consider a general missing data model. The observed variable is denoted $y$ and the latent variable is denoted $z$. In the state-space setting, we thus have $y = y_{1:T}$ and $z = z_{1:T}$. Let $p_\theta (y)$ be the likelihood of the data, parameterized by $\theta \in \Theta$. For each $\theta$, the complete data likelihood is given by $p_\theta (z, y)$ and the posterior of $z$ given $y$ is $p_\theta (z | y) = p_\theta (z, y) / p_\theta (y)$.

Define, $Q(\theta, \theta') = \int \log p_\theta (z, y) p_{\theta'} (z | y) \, dz$. The EM algorithm (Dempster et al., 1977) is an iterative method, which maximizes $p_\theta (y)$ by iteratively maximizing the auxiliary quantity $Q(\theta, \theta')$. It is useful when maximization of $\theta \mapsto Q(\theta, \theta')$, for fixed $\theta'$, is simpler than direct maximization of the likelihood, $\theta \mapsto p_\theta (y)$. The procedure is initialized at some $\theta_0 \in \Theta$ and then iterates between two steps, expectation (E) and maximization (M),

- (E) Compute $Q(\theta, \theta_{k-1})$.
- (M) Compute $\theta_k = \arg\max_{\theta \in \Theta} Q(\theta, \theta_{k-1})$.

The resulting sequence $\{\theta_k\}_{k \geq 0}$ will, under weak assumptions, converge to a stationary point of the likelihood $p_\theta (y)$.

We shall throughout this work assume that the M-step can be carried out straightforwardly, which is the case for many models encountered in practice. For the E-step, however, we
note that we have to compute an expectation under the posterior \( p_{\theta'}(z \mid y) \). In many situations, this computation is complicated or even intractable. One way to address this issue is to compute the E-step using Monte Carlo integration, leading to the MCEM algorithm (Wei and Tanner, 1990). Assume that it is possible to simulate from the posterior \( p_{\theta'}(z \mid y) \). Then, at iteration \( k \), the E-step is replaced by the following;

\[
(E') \quad \text{Generate } M_k \text{ realizations } \{z_j\}_{j=1}^{M_k} \text{ from } p_{\theta_{k-1}}(z \mid y) \text{ and compute, } \hat{Q}_k(\theta) = M_k^{-1} \sum_{j=1}^{M_k} \log p_{\theta}(z_j, y).
\]

The M-step is left unchanged, but now the Monte Carlo approximation \( \hat{Q}_k(\theta) \) is maximized in place of \( Q(\theta, \theta_{k-1}) \).

The MCEM algorithm can be very useful in situations where the E-step of the EM algorithm is intractable. A problem with MCEM, however, is that it relies on the number of simulations \( M_k \) to increase with \( k \) to be convergent (Cappé et al., 2005; Fort and Moulines, 2003). That is, the method can be thought of as doubly asymptotic, since it requires the number of iterations to tend to infinity, \( k \to \infty \), as well as the number of simulations, \( M_k \to \infty \). Furthermore, a complete set of simulated values \( \{z_j\}_{j=1}^{M_k} \) has to be generated at each iteration of the algorithm. After making an update of the parameter, these values are discarded and a new set has to be simulated at the next iteration.

To be able to make more efficient use of the simulated variables, a related method, referred to as stochastic approximation EM (SAEM) was proposed by Delyon et al. (1999). This method uses a stochastic approximation update of the auxiliary quantity \( Q \),

\[
\hat{Q}_k(\theta) = (1 - \gamma_k)\hat{Q}_{k-1}(\theta) + \gamma_k \left( \frac{1}{m_k} \sum_{j=1}^{m_k} \log p_{\theta}(z_j, y) \right).
\]

The E-step is thus replaced by the following;

\[
(E'') \quad \text{Generate } m_k \text{ realizations } \{z_j\}_{j=1}^{m_k} \text{ from } p_{\theta_{k-1}}(z \mid y) \text{ and update } \hat{Q}_k(\theta) \text{ according to (2)}.
\]

In (2), \( \{\gamma_k\}_{k \geq 1} \) is a decreasing sequence of positive step sizes, satisfying the usual stochastic approximation conditions, \( \sum_k \gamma_k = \infty \) and \( \sum_k \gamma_k^2 < \infty \). In SAEM, all simulated values contribute to \( \hat{Q}_{k-1}(\theta) \), but they are down-weighted using a forgetting factor given by the step size. Under appropriate assumptions, SAEM can be shown to converge for fixed \( m_k \) (e.g. \( m_k \equiv 1 \)), as \( k \to \infty \) (Delyon et al., 1999; Cappé et al., 2005). When the simulation step is computationally involved, there is a considerable computational advantage of SAEM over MCEM (Delyon et al., 1999).

### 3 Conditional particle filter SAEM

We now turn to the new procedure; SAEM using conditional particle filters (CPF). We refer to this method as particle SAEM (PSAEM).
3.1 Markovian stochastic approximation

Let us return to our original problem; inference in nonlinear state-space models. The latent variable posterior is then given by $p_\theta(x_{1:T} \mid y_{1:T})$, i.e. by the joint smoothing distribution. This distribution is intractable to compute, as well as to sample from, for the models under study. We can thus not employ MCEM or SAEM directly. To address this issue, it has been suggested to use SMC, i.e. particle smoothers (PS), to compute the E-step of the EM algorithm (Schön et al., 2011; Olsson et al., 2008; Cappé et al., 2005). This leads to an SMC-analogue of MCEM, which we refer to as PSEM. Unfortunately, PSEM inherits the drawbacks of MCEM. That is, it relies on double asymptotics for convergence and is not able to reuse the simulated values (i.e. the particles) across iterations.

For instance, assume that we use the forward filter/backward simulator particle smoother by Godsill et al. (2004) in PSEM. At iteration $k$, we use $N_k$ particles and backward trajectories, with a computational complexity of $O(N_k^2)$. For this approach to be successful, we need to take $N_k$ large (in fact $N_k \to \infty$ as $k \to \infty$) which leads to a very computationally costly E-step. If we instead take an SAEM approach, it is sufficient to generate a single sample at each iteration ($m_k \equiv 1$), reducing the computational cost to $O(N_k)$. However, we still need to take $N_k$ large to get an accurate particle approximation from the PF. Furthermore, it is not clear how the approximation error for finite $N_k$ will affect the parameter estimates in the SAEM algorithm.

To avoid this, and thus be able to reduce the computational complexity, we will use a Markovian version of stochastic approximation (Benveniste et al., 1990; Andrieu et al., 2005). It has been recognized that it is not necessary to sample exactly from the posterior distribution of the latent variables, to assess convergence of the SAEM algorithm. Instead, it is sufficient to sample from a family of Markov kernels $\{M_\theta : \theta \in \Theta\}$, leaving the family of posteriors invariant (Kuhn and Lavielle, 2004). In our case, we thus seek a family of Markov kernels on $X_T$, such that, for each $\theta \in \Theta$, $M_\theta(dx_{1:T} \mid x'_{1:T})$ leaves the joint smoothing distribution $p_\theta(x_{1:T} \mid y_{1:T})$ invariant.

Assume for the time being that this family of kernels is available. At iteration $k$ of the SAEM algorithm, let $\theta_{k-1}$ be the previous value of the parameter estimate and let $x_{1:T}[k-1]$ be the previous draw from the Markov kernel. Then, we proceed by sampling

$$x_{1:T}[k] \sim M_{\theta_{k-1}}(dx_{1:T} \mid x_{1:T}[k-1]), \quad (3)$$

and updating the auxiliary quantity according to

$$\hat{Q}_k(\theta) = (1 - \gamma_k)\hat{Q}_{k-1}(\theta) + \gamma_k \log p_\theta(y_{1:T}, x_{1:T}[k]). \quad (4)$$

This quantity is then maximized w.r.t. $\theta$ in the M-step, analogously to the standard SAEM algorithm.

3.2 Conditional particle filter with ancestor sampling

To find the sought family of Markov kernels, we will make use of PMCMC theory (Andrieu et al., 2010). More precisely, I suggest to run a conditional particle filter with ancestor sampling (CPF-AS). The CPF-AS has previously been used for Gibbs sampling in a PMCMC setting (Lindsten et al., 2012). Other options are available, e.g. to use the original CPF by Andrieu et al. (2010) or the CPF with backward simulation, originally
proposed by Whiteley (2010). However, we focus here on CPF-AS since ancestor sampling has been found to considerably improve the mixing over the basic CPF. Furthermore, it can be implemented in a forward only recursion, its computational cost is linear in the number of particles and it allows for a simple type of Rao-Blackwellization (as will be discussed later).

The CPF-AS procedure is an SMC sampler, akin to a standard PF but with the difference that one particle at each time step is specified a priori. Let these prespecified particles be denoted \( x'_{1:T} = \{ x'_1, \ldots, x'_T \} \). The method is most easily described as an auxiliary PF (see e.g. Doucet and Johansen (2011); Gustafsson (2010); Pitt and Shephard (1999) for an introduction). As in a standard auxiliary PF, the sequence of distributions \( p_\theta(x_{1:t} | y_{1:t}) \), for \( t = 1, \ldots, T \), is approximated sequentially by collections of weighted particles. Let \( \{ x^i_{1:t-1}, w^i_{t-1} \}_{i=1}^N \) be a weighted particle system targeting \( p_\theta(x_{1:t-1} | y_{1:t-1}) \). That is, the particle system defines an empirical distribution,

\[
\tilde{p}_\theta^N(dx_{1:t-1} | y_{1:t-1}) \triangleq \sum_{i=1}^N \frac{w^i_{t-1}}{\sum_{l=1}^N w^l_{t-1}} \delta_{x^i_{1:t-1}}(dx_{1:t-1}),
\]

which approximates the target. To propagate this sample to time \( t \), we introduce the auxiliary variables \( \{ a^i_{t} \}_{i=1}^N \), referred to as ancestor indices. To generate a specific particle \( x^i_t \) at time \( t \), we first sample the ancestor index with \( \mathbb{P}(a^i_t = j) \propto w^j_{t-1} \). Then, \( x^i_t \) is sampled from some proposal kernel \( q_{\theta,t} \),

\[
x^i_t \sim q_{\theta,t}(x_t | x^{a^i_t}_{t-1}, y_t).
\]

Hence, \( a^i_t \) is the index of the ancestor particle at time \( t - 1 \), of particle \( x^i_t \). The particle trajectories can then be augmented according to

\[
x^i_{1:t} = \{ x^i_{1:t-1}, x^i_t \}.
\]

In this formulation, the resampling step is implicit and it corresponds to sampling the ancestor indices.

Now, in a standard auxiliary PF, we would repeat this procedure for each \( i = 1, \ldots, N \), to generate \( N \) particles at time \( t \). In CPF-AS, however, we condition on the event that \( x^i_t \) is contained in the collection \( \{ x^i_t \}_{i=1}^N \). To accomplish this, we sample according to (6) only for \( i = 1, \ldots, N - 1 \). The \( N \)th particle is then set deterministically; \( x^N_t = x^i_t \).

To be able to construct the \( N \)th particle trajectory as in (7), the conditioned particle has to be associated with an ancestor at time \( t - 1 \). That is, we need to generate a value for the ancestor variable \( a^N_t \). In CPF-AS, this is done in a so called ancestor sampling step, in which \( a^N_t \) is sampled conditionally on \( x^i_t \). From Bayes’ rule, it follows that \( p_\theta(x_{t-1} | x^i_t, y_{1:t}) \propto f_\theta(x^i_t | x_{t-1}) p_\theta(x_{t-1} | y_{1:t-1}) \). By plugging (5) into the above expression, we arrive at the approximation,

\[
\tilde{p}_\theta^N(dx_{t-1} | x^i_t, y_{1:t}) = \sum_{i=1}^N \frac{w^i_{t-1} f_\theta(x^i_t | x^i_{t-1})}{\sum_{l=1}^N w^l_{t-1} f_\theta(x^l_t | x^l_{t-1})} \delta_{x^i_{t-1}}(dx_{t-1}).
\]

To sample an ancestor particle for \( x^i_t \), we draw from this empirical distribution. That is, we sample the ancestor index with \( \mathbb{P}(a^N_t = j) \propto w^j_{t-1} f_\theta(x^j_t | x^j_{t-1}) \).
Finally, all the particles, for \( i = 1, \ldots, N \), are assigned importance weights, analogously to a standard auxiliary PF; \( w_t^i = W_{\theta,t}(x_t^i, x_{t-1}^i) \), where the weight function is given by,

\[
W_{\theta,t}(x_t, x_{t-1}) = \frac{g_{\theta}(y_t | x_t) f_\theta(x_t | x_{t-1})}{q_{\theta,t}(x_t | x_{t-1}, y_t)}.
\]  

This results in a new weighted particle system \( \{x_{1:t}^i, w_t^i\}_{i=1}^N \), targeting the joint smoothing distribution at time \( t \). The method is initialised by sampling from a proposal density \( x_1^i \sim q_{\theta,1}(x_1 | y_1) \) for \( i = 1, \ldots, N-1 \) and setting \( x_{1}^N = x_1^1 \). The initial particles are assigned weights \( w_1^i = W_{\theta,1}(x_1^i) \) where the weight function is given by \( W_1^\theta(x_1) = g_{\theta}(y_1 | x_1) p_{\theta}(x_1) / q_{\theta,1}(x_1 | y_1) \). The CPF-AS is summarized in Algorithm 1.

**Algorithm 1** CPF-AS, conditioned on \( x_{1:T}^i \)

1: Draw \( x_1^i \sim q_{\theta,1}(x_1 | y_1) \) for \( i = 1, \ldots, N-1 \).
2: Set \( x_1^N = x_1^1 \).
3: Set \( w_1^i = W_{\theta,1}(x_1^i) \) for \( i = 1, \ldots, N \).
4: for \( t = 2 \) to \( T \) do
5: \( \quad \text{Draw } a_t^i \text{ with } \mathbb{P}(a_t^i = j) \propto w_{t-1}^i \) for \( i = 1, \ldots, N-1 \).
6: \( \quad \text{Draw } x_t^i \sim q_{\theta,t}(x_t | x_{t-1}^i, y_t) \) for \( i = 1, \ldots, N-1 \).
7: \( \quad \text{Draw } a_t^N \text{ with } \mathbb{P}(a_t^N = j) \propto w_{t-1}^N f_\theta(x_t^i | x_{t-1}^i) \).
8: Set \( x_t^N = x_t^1 \).
9: Set \( x_{1:t}^i = \{x_{1:t-1}^i, x_t^i\} \) for \( i = 1, \ldots, N \).
10: Set \( w_t^i = W_{\theta,t}(x_{1:t}^i, x_t^i) \) for \( i = 1, \ldots, N \).
end for

It might not be obvious why it is attractive to condition the PF on a prespecified set of particles. The reason for why this is useful is that it implies an invariance property which is key to our development. To state this more formally, we first make a standard assumption on the support of the proposal kernels used in the PF.

(A1) For any \( \theta \in \Theta \) and any \( t \in \{1, \ldots, T\} \), \( S_t^\theta \subseteq Q_t^\theta \) where,

\[
S_t^\theta = \{x_{1:t} \in X^t : p_\theta(x_{1:t} \mid y_{1:t}) > 0\},  
Q_t^\theta = \{x_{1:t} \in X^t : q_{\theta,t}(x_t \mid x_{t-1}, y_t) p_\theta(x_{1:t-1} \mid y_{1:t-1}) > 0\}.
\]

The key property of CPF-AS can now be stated as follows.

**Proposition 1.** Assume (A1). Then, for any \( \theta \in \Theta \) and any \( N \geq 2 \), the procedure;

(i) Run Algorithm 1 conditionally on \( x_{1:T}^i \);
(ii) Sample \( x_{1:T}^* \) with \( \mathbb{P}(x_{1:T}^* = x_{1:T}^i) \propto w_{T}^i \);

defines a \( p \)-irreducible and aperiodic Markov kernel on \( X_T \), with invariant distribution \( p_\theta(x_{1:T} \mid y_{1:T}) \).

**Proof:** The invariance property follows by the construction of the CPF-AS by Lindsten.
et al. (2012), and the fact that the law of $x^*_{1:T}$ is independent of permutations of the particle indices. This allows us to always place the conditioned particles at the $N$th position. Irreducibility and aperiodicity follows from (Andrieu et al., 2010, Theorem 5).

In other words, if $x'_{1:T} \sim p_\theta(x_{1:T} \mid y_{1:T})$ and we sample $x^*_{1:T}$ according to the procedure in Proposition 1, then, for any number of particles $N$, it holds that $x^*_{1:T} \sim p_\theta(x_{1:T} \mid y_{1:T})$. To understand this result, it can be instructive to think about the extreme cases, $N = 1$ and $N = \infty$, respectively. For $N = 1$, since we condition on $x'_{1:T}$, the sampling procedure will simply return $x^*_{1:T} = x'_{1:T}$. Since $x'_{1:T}$ is distributed according to the exact smoothing distribution, then so is $x^*_{1:T}$ (though, with correlation 1 between $x^*_{1:T}$ and $x'_{1:T}$). For $N = \infty$, on the other hand, the conditioning will have a negligible effect on Algorithm 1. That is, the CPF-AS reduces to a standard PF, using an infinite number of particles. Since the PF in this case exactly recovers the joint smoothing distribution, it again holds that $x^*_{1:T} \sim p_\theta(x_{1:T} \mid y_{1:T})$, but now $x^*_{1:T}$ is independent of $x'_{1:T}$.

Intuitively, using a fixed $N$ can be though of as an interpolation between these two results. The invariance property will hold for any $N$, but the larger we take $N$, the smaller the correlation will be between $x^*_{1:T}$ and $x'_{1:T}$. However, it has been experienced in practice that the correlation drops of very quickly as $N$ increases (Lindsten et al., 2012; Lindsten and Schön, 2012), and for many models a moderate $N$ (e.g. in the range 5–20) is enough to get a rapidly mixing kernel.

### 3.3 Final identification algorithm

From Proposition 1, if follows that CPF-AS defines a Markov kernel with the invariance properties needed for the SAEM algorithm. Before stating the final identification algorithm, however, we note that it is possible to reuse all $N$ particle trajectories when updating the auxiliary quantity (4). That is, we update $\hat{Q}_k$ according to,

$$
\hat{Q}_k(\theta) = (1 - \gamma_k)\hat{Q}_{k-1}(\theta) + \gamma_k \sum_{i=1}^N \frac{w_i^T}{N} \log p_\theta(y_{1:T}, x^i_{1:T}).
$$

(9)

Let $J$ be the random index of the extracted particle trajectory in Proposition 1, i.e. $x^*_{1:T} = x^J_{1:T}$. Then, (9) is simply a Rao-Blackwellization over $J$. Consequently, the variance of (9) will be lower than that of (4). It should be noted, however, that the variance reduction in general will be quite small, due to path degeneracy of the CPF-AS algorithm.

We summarize the proposed method for maximum likelihood inference in nonlinear state-space models, PSAEM, in Algorithm 2. The method is run until convergence, as checked by some standard convergence criterion.

### 4 Numerical illustration

To start with, we evaluate PSAEM on a 1st order linear system, for which the exact ML estimator (MLE) is readily available. The system is given by $x_{t+1} = ax_t + v_t$ and $y_t = x_t + e_t$ with $a = 0.9$ and where $e_t$ and $v_t$ are independent white Gaussian sequences, with variances $\sigma^2_e = \sigma^2_v = 1$. We simulate 100 batches of data from the system, each with
Algorithm 2 PSAEM

1: Set $\theta_0$ and $x_{1:T}[0]$ arbitrarily. Set $\hat{Q}_0(\theta) \equiv 0$.
2: for $k \geq 1$ do
3: \hspace{1em} Generate $\{x^i_{1:T}, w^i_T\}_{i=1}^N$ by running Algorithm 1, conditioned on $x_{1:T}[k-1]$ and targeting $p_{\theta_{k-1}}(x_{1:T} \mid y_{1:T})$.
4: \hspace{1em} Compute $\hat{Q}_k(\theta)$ according to (9).
5: \hspace{1em} Compute $\theta_k = \arg \max_{\theta \in \Theta} \hat{Q}_k(\theta)$.
6: \hspace{1em} Sample $J$ with $P(J = i) \propto w^i_T$ and set $x_{1:T}[k] = x^J_{1:T}$.
7: end for

$T = 100$. We then run Algorithm 2 for each batch to identify $\theta = (a, \sigma^2_v, \sigma^2_e)$. We use $N_{CPF} = 15$ particles and a bootstrap proposal kernel in the CPF-AS. We let $\gamma_k \equiv 1$ for $k \leq 100$, and $\gamma_k \sim k^{-0.7}$ for $k > 100$. This allows for a rapid change in the parameter estimates during the initial iterations, followed by a convergent phase. We are interested in comparing the estimates with the true MLE, $\hat{\theta}_{ML}$ (this is computed by running an exact EM algorithm for 10000 iterations). We thus compute the differences $\theta_k - \hat{\theta}_{ML}$ for $k \in \{10^2, 10^3, 10^4, 10^5\}$. Box plots of these differences, over the 100 data batches, are given in Figure 1. Despite the fact that we use a fixed (and small) number of particles, PSAEM converges to the true MLE as $k$ increases.

As a second, more challenging, example we consider the standard nonlinear time series model, used among others by Schön et al. (2011) and Cappé (2009),

\begin{align}
    x_{t+1} &= 0.5x_t + 25 \frac{x_t}{1 + x_t^2} + 8 \cos(1.2t) + v_t, \quad (10a) \\
    y_t &= 0.05x_t^2 + e_t, \quad (10b)
\end{align}

where $v_t$ and $e_t$ are independent white Gaussian sequences, with variances $\sigma^2_v = 1$ and $\sigma^2_e = 0.1$, respectively. We use PSAEM with $N_{CPF} = 15$ particles to identify $\theta = (\sigma^2_v, \sigma^2_e)$. The step size is set as above. As a comparison, we use the PSEM algorithm (Schön et al., 2011), based on a forward filtering/backward simulation (FFBSi) smoother (Godsill et al., 2004), with $N_{PS} = 1500$ forward filter particles and $M_{PS} = 300$ backward
trajectories.

It is interesting to note that the computational complexity of PSAEM scales like $O(N_{CPF}T)$ per iteration. Similarly, the complexity of PSEM is $O(N_{PS}M_{PS}T)$. There is a striking difference. If we neglect all computational overhead, PSAEM is a factor $M_{PS}N_{PS}/N_{CPF} = 30,000$ less costly than PSEM!

Despite this difference, we compare the methods over equally many iterations. We generate 20 independent batches of data, each consisting of $T = 1500$ observations. For each data set, we run PSAEM and PSEM for 2000 iterations. The methods are initialized uniformly at random, $\theta_0 \in [1, 2]^2$. The results are given in Figure 2. There is a clear difference between the methods, where PSAEM outperforms PSEM in both variance and bias (and, most notably, in computational time). Despite the fact that we use a fixed number of particles $N_{CPF} = 15$ at each iteration, PSAEM converges as we increase the number of iterations. This is not the case for PSEM, as this would require $N_{PS} \to \infty$ and $M_{PS} \to \infty$.

5 Conclusions

Conditional particle filters (CPFs) provide an elegant way of using SMC to construct Markov kernels which leave the exact joint smoothing distribution invariant. This holds true for any number of particles. With ancestor sampling, it is also possible to obtain a rapidly mixing kernel with a very modest number of particles. This is a key observation, meaning that we have to revise the common notion that SMC necessarily implies a high computational cost. In this contribution, the CPF with ancestor sampling has been combined with a stochastic approximation EM (SAEM) algorithm. The resulting method, PSAEM, was shown to be an efficient method for maximum likelihood parameter estimation in nonlinear/non-Gaussian state-space models. Indeed, PSAEM outperformed a state of the art particle-based EM algorithm in a simulation study, both in terms of bias, variance and computation time.

\footnote{Asymptotically (as $N_{PS} \to \infty$), this can be reduced to $O(N_{PS}T)$ by using a rejection-sampling-based FFBSi (Douc et al., 2011). In practice, however, the constants can be quite large and the actual gain limited (Taghavi et al., 2013; Bunch and Godsill, 2013).}
Figure 2: Parameter estimates over 20 iterations for PSAEM (left) and PSEM (right). Each line corresponds to one realization of data. The true values are $\sigma_v = 1$ and $\sigma_e = \sqrt{0.1}$, respectively.
Bibliography


Paper E

Rao-Blackwellized particle smoothers for mixed linear/nonlinear state-space models

Authors: Fredrik Lindsten, Pete Bunch, Simon J. Godsill and Thomas B. Schön

Edited version of the paper:

Rao-Blackwellized particle smoothers for mixed linear/nonlinear state-space models

Fredrik Lindsten*, Pete Bunch**, Simon J. Godsill** and Thomas B. Schön*

*Dept. of Electrical Engineering, Linköping University, SE–581 83 Linköping, Sweden {lindsten,schon}@isy.liu.se
**Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK {pb404,sjg30}@cam.ac.uk

Abstract

We consider the smoothing problem for a class of conditionally linear Gaussian state-space (CLGSS) models, referred to as mixed linear/nonlinear models. In contrast to the better studied hierarchical CLGSS models, these allow for an intricate cross dependence between the linear and the nonlinear parts of the state vector. We derive a Rao-Blackwellized particle smoother (RBPS) for this model class by exploiting its tractable substructure. The smoother is of the forward filtering/backward simulation type. A key feature of the proposed method is that, unlike existing RBPS for this model class, the linear part of the state vector is marginalized out in both the forward direction and in the backward direction.

1 Introduction

Particle filters (PF) and particle smoothers (PS) are useful for state inference in nonlinear state-space models (SSM) (Doucet and Johansen, 2011; Gustafsson, 2010). It is well recognized that the Rao-Blackwellized PF (RBPF) (Schön et al., 2005; Chen and Liu, 2000; Doucet et al., 2000), can be used to address the filtering problem in conditionally linear Gaussian state-space (CLGSS) models. By exploiting the tractable substructure present in these models, the RBPF results in more accurate estimators than a standard PF (Lindsten et al., 2011; Chopin, 2004) and it can therefore be used for filtering in even more challenging, e.g. high-dimensional, models.

However, Rao-Blackwellization has not been as well explored for smoothing. Most Rao-Blackwellized particle smoothers (RBPS) have been focused on a type of hierarchical CLGSS models, for which the “nonlinear state” is Markovian (Särkkä et al., 2012; Whiteley et al., 2010; Fong et al., 2002). Here, we consider instead a class of mixed linear/nonlinear SSMs, given by

\[
\begin{align*}
  u_{t+1} &= g(u_t) + B(u_t)z_t + G(u_t)v_t^u, \\
  z_{t+1} &= f(u_t) + A(u_t)z_t + F(u_t)v_t^z, \\
  y_t &= h(u_t) + C(u_t)z_t + e_t,
\end{align*}
\]
where \( v_t^u \sim \mathcal{N}(0, I) \), \( v_t^z \sim \mathcal{N}(0, I) \) and \( e_t \sim \mathcal{N}(0, R(u_t)) \). We assume that \( Q(u_t) \triangleq G(u_t)G(u_t)^T \) and \( R(u_t) \) are invertible, but we do not assume invertibility of \( F(u_t)F(u_t)^T \).

The state consists of two parts, \( x_t = (u_t, z_t) \), where there is a nonlinear dependence on \( u_t \) (referred to as the \textit{nonlinear state}) and an affine dependence on \( z_t \) (referred to as the \textit{linear state}).

This model is CLGSS, but it is more involved than a hierarchical CLGSS models, since there is a cross-dependence between the linear and the nonlinear states. That is, the nonlinear state process alone is non-Markovian. This class of models arise, for instance, when the observations depend nonlinearly on a subset of the states in a system with linear dynamics. An RBPF for the mixed linear/nonlinear model was derived and used for terrain-aided aircraft navigation by Schön et al. (2005). In this contribution, we derive a novel RBPS, akin to the recent contributions for hierarchical CLGSS models by Särkkä et al. (2012); Whiteley et al. (2010), but applicable to the model (1).

A key property of the proposed method is that it only samples the nonlinear part of the state, both in the forward and backward directions, as opposed to, for instance Fong et al. (2002); Lindsten and Schön (2011), who sample the full state in the backward direction.

For a vector \( \mu \) and a positive semidefinite matrix \( \Omega \succeq 0 \), we write \( \|\mu\|_2^2 \triangleq \mu^T \Omega \mu \). We write \( |A| \) for matrix determinant and \( \mathcal{N}(\mu, \Sigma) \) and \( \mathcal{N}(x; \mu, \Sigma) \) for the Gaussian distribution and probability density function (PDF), respectively.

## 2 Background

### 2.1 Particle filtering and smoothing

Consider first a standard, Markovian SSM: \( x_{t+1} = f(x_t) + v_t \) and \( y_t = h(x_t) + e_t \), where \( f \) and \( h \) are nonlinear functions, and \( v_t \) and \( e_t \) have known, tractable densities. A particle filter is a sequential Monte Carlo algorithm used to approximate the intractable filtering density, representing it with a set of weighted particles \( \{x_{1:t}, w^i_t\}_{i=1}^N \), each of which is a state trajectory \( x_{1:t} \),

\[
\hat{p}^N(dx_{1:t} | y_{1:t}) \triangleq \sum_{i=1}^N w^i_t \delta_{x^i_{1:t}}(dx_{1:t}). \tag{2}
\]

In the simplest particle filter, the \( t \)-th set of particles are formed by sampling \( x_{1:t-1} \) from the previous distribution and then \( x_t \) from an importance distribution. A weight is assigned to each particle to account for the difference between the proposal and the target density. Note that an approximation to \( p(x_t | y_{1:t}) \) is obtained by marginalization of (2), which equates to simply discarding \( x_{1:t-1} \).

The term “smoothing” encompasses a number of related inference problems, but here we focus on the estimation of the complete joint smoothing distribution, \( p(x_{1:T} | y_{1:T}) \). This distribution is approximated at the final step of the particle filter (Kitagawa, 1996). However, this approximation suffer from the problem of path degeneracy, i.e. the number of unique particles decreases rapidly for \( t \ll T \). A diverse set of particles may be generated
by sampling state trajectories using the forward filtering/backward simulation (FFBSi) algorithm (Godsill et al., 2004).

FFBSi exploits a sequential factorization of the joint smoothing density

$$p(x_{1:T} \mid y_{1:T}) = p(x_T \mid y_{1:T}) \prod_{t=1}^{T-1} p(x_t \mid x_{t+1:T}, y_{1:T}).$$

A final state, $x_T'$, is first sampled from the particle filter approximation $\hat{p}^N(dx_T' \mid y_{1:T})$. Then, working sequentially backwards from time $T$, each subsequent state $x_t'$ is sampled from the backwards kernel, $p(x_t \mid x_{t+1:T}, y_{1:T})$. The resulting trajectory \(\{x_t'\}_{t=1}^T\) is then a sample from the smoothing distribution.

Using the Markov property, the backward kernel may be expanded as

$$p(x_t \mid x_{t+1:T}, y_{1:T}) \propto p(x_{t+1} \mid x_t)p(x_t \mid y_{1:t}). \quad (3)$$

The second factor is approximated using the particle filter, leading to the representation,

$$\hat{p}^N(dx_t \mid x_{t+1:T}, y_{1:T}) \triangleq \sum_{i=1}^N \tilde{w}_{t|T}^i \delta_{x_t^i} (dx_t), \text{ with } \tilde{w}_{t|T}^i \propto w_t^i p(x_{t+1} \mid x_t).$$

### 2.2 Rao-Blackwellized particle filtering

The structure of a CLGSS model (such as (1)) can be exploited by using an RBPF, by exploiting the factorization $p(u_{1:t}, z_t \mid y_{1:t}) = p(z_t \mid u_{1:t}, y_{1:t})p(u_{1:t} \mid y_{1:t})$. Since the model is CLGSS, it holds that

$$p(z_t \mid u_{1:t}, y_{1:t}) = \mathcal{N}(z_t; \tilde{z}_{t|t}(u_{1:t}), P_{t|t}(u_{1:t})). \quad (4)$$

A PF is used to estimate only the nonlinear state marginal density while conditional Kalman filters, one for each particle, are used to compute the moments for the linear state in (4). The resulting RBPF approximation is given by

$$\hat{p}^N(du_{1:t}, dz_t \mid y_{1:t}) = \sum_{i=1}^N w_t^i \mathcal{N}(dz_t; \tilde{z}_{t|t}^i, P_{t|t}^i(u_{1:t}))\delta_{u_{1:t}^i}(du_{1:t}).$$

The particle weights are given by the ratio of the Gaussian density $p(y_t, u_t \mid u_{1:t-1}, y_{1:t-1})$ and the importance density. See Schön et al. (2005) for details. The reduced dimensionality of the particle approximation results in a reduction in variance of associated estimators (Lindsten et al., 2011; Chopin, 2004).

### 3 Rao-Blackwellized particle smoothing

The new RBPS for the model (1) is an FFBSi which uses the RBPF as a forward filter. The novelty lies in the construction of a backward simulator which samples only the nonlinear state in the backward pass. Difficulty arises because the nonlinear state process is non-Markovian. Practically, this means that the backward kernel cannot be expressed in a simple way, as in (3). Furthermore, it implies that the measurement likelihood depends on the complete history $u_{1:t}$; we must therefore sample whole trajectories produced by the RBPF. More precisely, let $u_{1:t+1:T}^i$ be a partial, nonlinear backward trajectory. To extend this trajectory to time $t$, we draw one of the RBPF particles \(\{u_{1:t}^i\}_{i=1}^N\) (with probabilities
computed below), set \( u_{t:T}^i = \{ u_t^i, u_{t+1:T}^i \} \) and discard \( u_{t-1:T}^i \). This procedure is repeated for each time \( t = T - 1, \ldots, 1 \), resulting in a complete backward trajectory.

To compute the backward sampling probabilities, we note that

\[
p(u_{1:t} | u_{t+1:T}, y_{1:T}) \propto p(y_{t+1:T}, u_{t+1:T} | u_{1:t}, y_{1:t}) p(u_{1:t} | y_{1:t}). \tag{5}
\]

The second factor in this expression can be approximated by the forward RBPF, analogously to a standard FFBSi. This results in a point-mass approximation of the backward kernel, given by

\[
\hat{p}^N (du_{1:t} | u_{t+1:T}, y_{1:T}) = \sum_{i=1}^{\infty} \tilde{w}_{t/T}^i \delta_{u_{1:t}^i} (du_{1:t}), \tag{6}
\]

with

\[
\tilde{w}_{t/T}^i \propto w_t^i p(y_{t+1:T}, u_{t+1:T}^i | u_{1:t}^i, y_{1:t}). \tag{7}
\]

It remains to find an expression for the predictive PDF in this expression (up to proportionality). In fact, this PDF can be computed straightforwardly by running a conditional Kalman filter from time \( t \) up to time \( T \). However, using such an approach to calculate the weights at time \( t \) would require \( N \) separate Kalman filters to run over \( T - t \) time steps, resulting in a total computational complexity scaling quadratically with \( T \). To avoid this, we seek an efficient recursion for the weights \( (7) \). This is accomplished by propagating a set of statistics backward in time, as the trajectory \( u_{1:T}^i \) is generated. The idea is similar to that of Gerlach et al. (2000); Whiteley et al. (2010); Särkkä et al. (2012) but our derivation is adapted to the mixed linear/nonlinear model \( (1) \). To start with, we express the predictive PDF as

\[
p(y_{t+1:T}, u_{t+1:T} | u_{1:t}, y_{1:t}) = \int p(y_{t+1:T}, u_{t+1:T} | z_t, u_t) p(z_t | u_{1:t}, y_{1:t}) dz_t, \tag{8}
\]

where the second factor of the integrand is given by the RBPF \( (4) \). Hence, we seek an expression for the first factor of the integrand. The following two propositions, which will be used alternately in the backward simulation, provide the updating equations for a set of sufficient statistics for this PDF. For brevity, we write \( A_t \) for \( A(u_t) \), etc.

**Proposition 1 (Backward prediction).** Given \( \hat{\Omega}_{t+1} \) and \( \hat{\lambda}_{t+1} \) as in Proposition 2, for any \( 1 \leq t \leq T - 1 \),

\[
p(y_{t+1:T}, u_{t+1:T} | z_t, u_t) \propto Z_t \exp \left( -\frac{1}{2} (z_T^T \Omega_t z_t - 2\lambda_t^T z_t) \right),
\]

where \( Z_t, \Omega_t \succeq 0 \) and \( \lambda_t \) depend on \( u_t \), but are independent of \( z_t \), and the proportionality is w.r.t. \((u_t, z_t)\). The updated statistics are given by,

\[
Z_t = |M_t|^{-1/2} |Q_t|^{-1/2} \exp \left( -\frac{1}{2} \tau_t \right),
\]

\[
\Omega_t = A_t^T \left( \hat{\Omega}_{t+1} - \hat{\Omega}_{t+1} F_t M_t^{-1} F_t^T \hat{\Omega}_{t+1} \right) A_t + B_t^T Q_t^{-1} B_t,
\]

\[
\lambda_t = A_t^T \left( I - \hat{\Omega}_{t+1} F_t M_t^{-1} F_t^T \right) m_t + B_t^T Q_t^{-1} (u_{t+1} - g_t),
\]
We thus have a recursion for updating the statistics \( Z_t \).

As for a standard FFBSi, the backward simulation is typically repeated \( M \) times, to generate a set of backward trajectories \( \{ u_{1:T}^{(j)} \}_{j=1}^M \) which can be used to approximate \( p(u_{1:T} | y_{1:T}) \). If we seek smoothed estimates of the linear states, these can be computed, e.g. by running a modified Bryson-Frazier (Bierman, 1973) or a Rauch-Tung-Striebel (RTS) (Rauch et al., 1965) smoother for each backward trajectory (alternatively, we can run conditional Kalman filters and fuse the filter estimates with the backward statistics).

Proof: See Section 4.

**Proposition 2 (Update).** Given \( \Omega_t \) and \( \lambda_t \) as in Proposition 1, for any \( 1 \leq t \leq T-1 \),

\[
p(y_{t:T} | u_{t+1:T}, z_t, u_t) \propto \exp \left( -\frac{1}{2} \left( z_t^T \hat{\Omega}_t z_t - 2\hat{\lambda}_t z_t \right) \right),
\]

where \( \hat{\Omega}_t \geq 0 \) and \( \hat{\lambda}_t \) depend on \( u_t \), but are independent of \( z_t \), and the proportionality is w.r.t. \( z_t \). The updated statistics are given by,

\[
\hat{\Omega}_t = \Omega_t + C_t^T R_t^{-1} C_t,
\]

\[
\hat{\lambda}_t = \lambda_t + C_t^T R_t^{-1} (y_t - h_t).
\]

Furthermore, at time \( T \), it holds that

\[
p(y_T | z_T, u_T) \propto \exp \left( -\frac{1}{2} \left( z_T^T \hat{\Omega}_T z_T - 2\hat{\lambda}_T z_T \right) \right),
\]

with \( \hat{\Omega}_T = C_T^T R_T^{-1} C_T \) and \( \hat{\lambda}_T = C_T^T R_T^{-1} (y_T - h_T) \).

Proof: See Section 4.

We thus have a recursion for updating the statistics \( Z_t, \Omega_t \) and \( \lambda_t \). Using these quantities, together with (4), we can solve the integral (8). This is formalized in the next proposition.

**Proposition 3.** Let \( \tilde{z}_{t|t} \) and \( P_{t|t} = \Gamma_{t|t} \Gamma_{t|t}^T \) be given as in (4) and let \( Z_t, \Omega_t \) and \( \lambda_t \) be given as in Proposition 1. Then,

\[
p(y_{t+1:T} | u_{t+1:T}, u_{1:t}, y_{1:t}) \propto Z_t | \Lambda_t |^{-1/2} \exp \left( -\frac{1}{2} \eta_t \right),
\]

where the proportionality is w.r.t. \( u_{1:t} \) and where,

\[
\eta_t = \| \tilde{z}_{t|t} \|_{\tilde{\Omega}_{t|t}}^2 - 2\lambda_t^T \tilde{z}_{t|t} - \| \Gamma_{t|t}^T (\lambda_t - \Omega_t \tilde{z}_{t|t}) \|_{\lambda_{t|t}}^2,
\]

\[
\Lambda_t = \Gamma_{t|t}^T \Omega_t \Gamma_{t|t} + I.
\]

Proof: See Section 4.

By plugging this result into (7), we obtain an expression for the backward sampling weights. The resulting RBPS is given in Algorithm 1.
Algorithm 1 Rao-Blackwellized FFBSi

1: **Forward filter:** Run an RBPF for time \( t = 1, \ldots, T \). For each \( t \), store \( \{u^i_t, w^i_t, z^i_t, \Gamma^i_t\}_{i=1}^N \).

2: **Initialize:** Draw \( u'_T = u^i_T \) with probability \( w^i_T \). Compute \( \hat{\Omega}_T \) and \( \hat{\lambda}_T \) as in Proposition 2.

3: for \( t = T - 1 \) to 1 do

4: for \( i = 1 \) to \( N \) do

5: Compute \( \{Z^i_t, \Omega^i_t, \lambda^i_t\} \) as in Proposition 1.

6: Compute \( \{\Lambda^i_t, \eta^i_t\} \) as in Proposition 3.

7: Compute \( \tilde{W}^i_t = w^i_T Z^i_t | \Lambda^i_t |^{-1/2} \exp \left( -\frac{1}{2} \eta^i_t \right) \).

8: end for

9: Normalize the weights, \( \tilde{w}^i_t | T = \tilde{W}^i_t / \sum_i \tilde{W}^i_t \).

10: Set \( J = i \) with probability \( \tilde{w}^i_t | T \).

11: Set \( u'_{t,T} = \{u^J_t, u'_{t,T}\} \) and \( \{\Omega_t, \lambda_t\} = \{\Omega^J_t, \lambda^J_t\} \).

12: Compute \( \{\hat{\Omega}_t, \hat{\lambda}_t\} \) as in Proposition 2.

13: end for

The total computational complexity of generating \( M \) backward trajectories, using \( N \) forward filter particles, is \( O(NMT) \) (i.e. the same as for a standard FFBSi). The computational cost can be reduced by using the rejection-sampling-based FFBSi by Douc et al. (2011) or the Metropolis-Hastings-based FFBSi by Bunch and Godsill (2013).

4 Proofs

In this section, we prove Propositions 1–3. We start with a useful lemma.

**Lemma 1.** Let \( \xi \sim \mathcal{N}(0, I) \) and let \( z = c + Ax + \Gamma \xi \), for some constant vectors \( c \) and \( x \) and matrices \( A \) and \( \Gamma \) of appropriate dimensions. Let \( \Omega \succeq 0 \) and \( \lambda \) be a constant matrix and vector, respectively. Then \( \mathbb{E} \left[ \exp \left( -\frac{1}{2} \left( z^T \Omega z - 2 \lambda^T z \right) \right) \right] = |M|^{-1/2} \exp \left( -\frac{1}{2} \gamma \right) \) with,

\[
\gamma = \|A x\|^2_{\Omega - \Omega \Gamma M^{-1} \Gamma^T \Omega} - 2 x^T A^T \left( I - \Omega \Gamma M^{-1} \Gamma^T \right) m + \|c\|^2_{\Omega} - 2 \lambda^T c - \|\Gamma^T m\|^2_{M^{-1}},
\]

where \( m = \lambda - \Omega c \) and \( M = \Gamma^T \Omega \Gamma + I \).

**Proof:** A detailed proof is omitted for brevity. The result follows by plugging in the expression for \( z \) and carrying out the integration w.r.t. \( \xi \). \( \square \)

Propositions 1 and 2 are given by induction. The initialization at time \( T \) in Proposition 2 follows directly from (1c),

\[
p(y_T \mid z_T, u_T) = \mathcal{N}(y_T; h_T + C_T z_T, R_T) \alpha \exp \left( -\frac{1}{2} \left( \|C_T z_T\|^2_{R_T^{-1}} - 2 z_T^T C_T R_T^{-1} (y_T - h_T) \right) \right). \quad (9)
\]
Hence, assume that Proposition 2 holds at time $t + 1$. We have,

$$
p(y_{t+1:T}, u_{t+1:T} | z_t, u_t) = p(u_{t+1} | z_t, u_t) \int p(y_{t+1:T}, u_{t+2:T} | z_{t+1}, u_{t+1}) p(z_{t+1} | z_t, u_t) \, dz_{t+1}.
$$

The first factor is a Gaussian PDF, given by (1a),

$$
p(u_{t+1} | z_t, u_t) = \mathcal{N}(u_{t+1} | g_t + B_t z_t, Q_t)
$$

$$
\propto |Q_t|^{-1/2} \exp \left( -\frac{1}{2} \left( \|u_{t+1} - g_t\|^2_{Q_t^{-1}} + \|B_t z_t\|^2_{Q_t^{-1}} - 2 z_t^T B_t^T Q_t^{-1} (u_{t+1} - g_t) \right) \right).
$$

To compute the integral in (10), we use the induction hypothesis and (1b). We then apply Lemma 1 with $c = f_t, A = A_t, x = z_t, \Gamma = F_t, \Omega = \hat{\Omega}_{t+1} + \hat{\lambda}_{t+1}$. Proposition 1 then follows by collecting terms from the two factors.

Next, to prove Proposition 2 for $t < T$, we assume that Proposition 1 holds at time $t$. We have,

$$
p(y_{t:T}, u_{t+1:T} | z_t, u_t) = p(y_t | z_t, u_t) p(y_{t+1:T}, u_{t+1:T} | z_t, u_t).
$$

The first factor is given by (1c), analogously to (9), and the second factor is given by Proposition 1. The result follows by collecting terms from the two factors.

Finally, to prove Proposition 3 we note that the sought density is given by (8) where the two factors of the integrand are given by Proposition 1 and by (4), respectively. The result follows by applying Lemma 1 with $c = \hat{z}_{t|t}, x = 0, \Gamma = \Gamma_{t|t}, \Omega = \Omega_t$ and $\lambda = \lambda_t$.

## 5 Numerical results

We evaluate the proposed RBPS by comparing its performance with alternative smoothers. The following methods are considered:

- **FFBSi**: A non-Rao-Blackwellized FFBSi (Godsill et al., 2004).
- **RB-FF/JBS**: Rao-Blackwellized forward filter/joint backward simulator (Lindsten and Schön, 2011).
- **RB-FFBSi**: The proposed method (Algorithm 1).

For all methods, a bootstrap PF (Gordon et al., 1993) or RBPF (Schön et al., 2005) is used in the forward direction.

The RB-F/S consists of running an RBPF and storing the nonlinear state trajectories. Smoothed linear state estimates are then computed by running constrained RTS smoothers, conditionally on these nonlinear trajectories. The RB-FF/JBS is the “joint backward simulator with constrained RTS smoothing” by Lindsten and Schön (2011) (see also Fong et al. (2002)). In this method, we run an RBPF in the forward direction, but sample both the nonlinear and the linear states in the backward direction. The method relies on having access to the linear state samples in order to compute the backward sampling probabilities.
However, once the backward simulation is complete, the linear parts of the trajectories are discarded. Refined linear state estimates are then computed by, again, running constrained RTS smoothers, one for each nonlinear backward trajectory.

We consider a 5th order mixed linear/nonlinear system. The nonlinear part is given by the time series,

\[ u_{t+1} = 0.5u_t + \theta_t \frac{u_t}{1 + u_t^2} + 8 \cos(1.2t) + 0.071v_t^u, \tag{11a} \]
\[ y_t = 0.05u_t^2 + e_t, \tag{11b} \]

for some process \( \{\theta_t\}_{t \geq 1} \). The case with a static \( \theta_t \equiv 25 \) has been studied, among others, by Andrade Netto et al. (1979); Gordon et al. (1993). Here, we assume instead that \( \theta_t \) is a time varying parameter with known dynamics, given by the output from a 4th order linear system,

\[ z_{t+1} = \begin{pmatrix} 3 & -1.691 & 0.849 & -0.3201 \\ 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{pmatrix} z_t + 0.1v_t^z, \tag{12a} \]
\[ \theta_t = 25 + (0.04 \ 0.044 \ 0.008^T) z_t, \tag{12b} \]

with poles in \( 0.8 \pm 0.1i \) and \( 0.7 \pm 0.05i \). Combined, (11) and (12) is a mixed linear/nonlinear system. The noises are assumed to be white, Gaussian and mutually independent; \( v_t^u \sim \mathcal{N}(0, 1) \), \( v_t^z \sim \mathcal{N}(0, I) \) and \( e_t \sim \mathcal{N}(0, 0.1) \).

We generate 1000 batches of data from the system, each with \( T = 100 \) samples. We run the smoothers two times, first with \( N = 300 \) and then with \( N = 30 \) particles. The backward-simulation-based methods use \( M = N/3 \) backward trajectories, based on the recommendation to set \( M \lesssim N \) (Lindsten and Schön, 2013). Table 1 summarizes the results, in terms of the time averaged RMSE values for the nonlinear state \( u_t \) and for the time varying parameter \( \theta_t \) (note that \( \theta_t \) is a linear combination of the four linear states \( z_t \)).

<table>
<thead>
<tr>
<th>Smoother</th>
<th>( N = 300 )</th>
<th>( N = 30 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( u_t )</td>
<td>( \theta_t )</td>
</tr>
<tr>
<td>FFBSi</td>
<td>0.499</td>
<td>0.782</td>
</tr>
<tr>
<td>RB-F/S</td>
<td>0.424</td>
<td>0.660</td>
</tr>
<tr>
<td>RB-FF/JBS</td>
<td>0.399</td>
<td>0.579</td>
</tr>
<tr>
<td>RB-FFBSi</td>
<td>0.398</td>
<td>0.564</td>
</tr>
</tbody>
</table>

The proposed RB-FFBSi gives the most accurate results among the considered smoothers, both for \( N = 300 \) and \( N = 30 \). The difference between RB-FFBSi and RB-FF/JBS is quite small, but standard statistical hypothesis tests indeed indicate a clear statistical significance. In fact, these two methods are in many respects similar. They use similar forward and backward recursions and they both use conditional RTS smoothers to compute...
smoothed estimates of the linear states. Hence, in terms of implementation and computational complexity, they are almost identical. With this in mind, and from the fact that the results in Table 1 are in favor of RB-FFBSi, we believe that the RB-FFBSi indeed is the preferred method of choice, between these two smoothers. Furthermore, in the authors’ opinion, RB-FFBSi makes use of a more intuitively correct Rao-Blackwellization, since the marginalization is done both in the forward direction and in the backward direction.

For further comparison, Figure 1 shows the estimates of $\theta_t$ for one specific batch of data, using $N = 300$ and $M = 100$. This reveals a clear difference between the methods’ abilities of accurately representing the posterior distribution of $\theta_t$. For FFBSi and RB-F/S (the top row), there is a clear degeneracy in the trajectories. For RB-F/S, this is expected, as it is a direct effect of the path degeneracy of the RBPF. For the (non-Rao-Blackwellized) FFBSi, the degeneracy is caused by the fact that $N = 300$ particles is insufficient to represent the posterior in all five dimensions, resulting in that only a few particles get significantly non-zero weights. This will cause the backward simulator to degenerate, in the sense that many backward trajectories will coincide. The Rao-Blackwellized backward simulators (bottom row) perform much better in this respect, as there is a much larger diversity among the backward trajectories.

6 Conclusion

A new smoother for a class of mixed linear/nonlinear state-space models has been presented. The method is a forward filter/backward simulator which uses Rao-Blackwellization to exploit the conditionally linear Gaussian structure of the model. In contrast to previously developed algorithms for this model, the new smoother samples the linear state component in neither the forward nor the backward direction. Instead, a recursion has been derived which allows efficient calculation of backward sampling probabilities. Simulations have been used to demonstrate that the smoother functions well, with improvements in RMSE over previous algorithms.
Figure 1: Estimates of $\theta_t$ for $t = 1, \ldots, T$. From top left to bottom right; FFBSi, RB-F/S, RB-FF/JBS and RB-FFBSi. Each curve corresponds to one particle trajectory ($\theta_{1:T}$ for FFBSi and $\bar{\theta}_{1:T|T}'$ for the remaining smoothers). The true value is shown as a thick black line.
Bibliography


A non-degenerate Rao-Blackwellised particle filter for estimating static parameters in dynamical models

Authors: Fredrik Lindsten, Thomas B. Schön and Lennart Svensson

Edited version of the paper:

A non-degenerate Rao-Blackwellised particle
filter for estimating static parameters in
dynamical models

Fredrik Lindsten*, Thomas B. Schön* and Lennart Svensson**

*Dept. of Electrical Engineering, Linköping University, SE–581 83 Linköping, Sweden
{lindsten, schon}@isy.liu.se

**Division of Signals and Systems, Chalmers University, SE–412 96 Gothenburg, Sweden
lennart.svensson@chalmers.se

Abstract

The particle filter (PF) has emerged as a powerful tool for solving nonlin-
ear and/or non-Gaussian filtering problems. When some of the states en-
ter the model linearly, this can be exploited by using particles only for the
“nonlinear” states and employing conditional Kalman filters for the “linear”
states; this leads to the Rao-Blackwellised particle filter (RBPF). However,
it is well known that the PF fails when the state of the model contains some
static parameter. This is true also for the RBPF, even if the static states are
marginalised analytically by a Kalman filter. The reason is that the posterior
density of the static states is computed conditioned on the nonlinear particle
trajectories, which are bound to degenerate over time. To circumvent this
problem, we propose a method for targeting the posterior parameter density,
conditioned on just the current nonlinear state. This results in an RBPF-like
method, capable of recursive identification of nonlinear dynamical models
with affine parameter dependencies.

1 Introduction

We consider the filtering problem for a certain type of nonlinear dynamical state-space
models, with static state components. The typical case for when such models arise is
when the state is augmented with some unknown, static parameter. This is common in
e.g. simultaneous localisation and mapping (Thrun and Leonard, 2008) and in recursive,
Bayesian parameter estimation (Ljung and Söderström, 1983).

Let \( \{x_t\}_{t \geq 1} \) be the state process in a state-space model (SSM). That is, \( \{x_t\}_{t \geq 1} \) is a
discrete-time Markov process evolving according to a transition density \( p(x_{t+1} \mid x_t, \theta) \).
The states are hidden, but observed through the measurements \( y_t \), according to the obser-
vation density \( p(y_t \mid x_t, \theta) \). Here, \( \theta \) is an unknown static parameter with prior density
\( p(\theta) \). For the purpose of joint estimation of \( x_t \) and \( \theta \), we augment the state with a static
component $\theta_t \equiv \theta$. Hence, the SSM is described by,

$$x_{t+1} \sim p(x_{t+1} \mid x_t, \theta_t), \quad (1a)$$
$$\theta_{t+1} = \theta_t, \quad (1b)$$
$$y_t \sim p(y_t \mid x_t, \theta_t). \quad (1c)$$

Let $\xi_t = \{x_t, \theta_t\}$. We are interested in finding the joint filtering density $p(\xi_t \mid y_{1:t})$, i.e. the posterior density of the state $x_t$ and the parameter $\theta_t$ given a sequence of measurements $y_{1:t} \triangleq \{y_1, \ldots, y_t\}$. Let $p(\xi_1)$ be the joint prior density of the state and the parameter. The filtering density is then given by the Bayesian filtering recursions,

$$p(\xi_t \mid y_{1:t}) \propto p(y_t \mid \xi_t)p(\xi_t \mid y_{1:t-1}),$$
$$p(\xi_{t+1} \mid y_{1:t}) = \int p(\xi_{t+1} \mid \xi_t)p(\xi_t \mid y_{1:t}) d\xi_t,$$

for any $t \geq 1$, using the convention $p(\xi_1 \mid y_{1:0}) = p(\xi_1)$.

Despite the simplicity of these expressions, they are known to be intractable for basically any model, except linear Gaussian state-space (LGSS) models and models with finite state-space. For general, nonlinear and/or non-Gaussian models, some approximate method for computing the filtering density is required. One popular approach is to use sequential Monte Carlo (SMC) methods, commonly referred to as particle filters (PFs); see e.g. Doucet and Johansen (2011); Gustafsson (2010); Cappé et al. (2007). However, it is well known that the PF will fail when the state contains some static parameter (Andrieu et al., 2004; Cappé et al., 2007). The reason is that the exploration of the parameter space is restricted to the first time instant. Once the particles are initiated, their positions are fixed. At consecutive time points, the particles will be reweighted and resampled, but not moved to new positions.

In this paper we shall study the filtering problem for a special case of (1). More precisely, we assume that model is Gaussian with an affine dependence on the parameters,

$$x_{t+1} = f(x_t) + A(x_t)\theta_t + v_t, \quad (2a)$$
$$\theta_{t+1} = \theta_t, \quad (2b)$$
$$y_t = h(x_t) + C(x_t)\theta_t + e_t, \quad (2c)$$

where the process noise and measurement noise are white and Gaussian. Hence, conditioned on the trajectory $x_{1:t}$, the $\theta$-process is given by an LGSS model. For each time $t \geq 0$, the model can be seen as an LGSS with state $\theta_t$, if we fix the state trajectory $x_{1:t}$ up to that time. Hence, the conditional filtering density of $\theta_t$ given $x_{1:t}$ is Gaussian and available through the Kalman filter (KF). Models with this property are known as conditionally linear Gaussian state-space (CLGSS) models. It is worth to note that the method proposed in subsequent sections is applicable to any CLGSS model, and can be of interest also when the “linear state” is non-static, but slowly mixing (see Lindsten (2011) for details). However, we have chosen to present the method for the special case (2) for clarity.

The conditionally linear Gaussian substructure in a CLGSS model can be exploited when addressing the filtering problem using SMC methods. This leads to the Rao-Blackwellised
Degeneracy of the RBPF – the motivation for a new approach

2 Degeneracy of the RBPF – the motivation for a new approach

The presence of a conditionally linear Gaussian substructure can be exploited when addressing the filtering problem using SMC methods, leading to the RBPF by Doucet et al. (2000); Schön et al. (2005). The RBPF utilises the fact that the joint filtering density can be expressed according to

\[ p(x_t, \theta_t | y_{1:t}) = \int p(\theta_t | x_{1:t}, y_{1:t}) p(x_{1:t} | y_{1:t}) \, dx_{1:t-1} \]  

Now, since the model under study is CLGSS, the first factor of the integrand above is Gaussian and analytically tractable, using the KF. More precisely, it holds that

\[ p(\theta_t | x_{1:t}, y_{1:t}) = N(\theta_t; \hat{\theta}_{t|t}(x_{1:t}), P_{t|t}(x_{1:t})) \]  

for some (tractable) sequence of mean and covariance functions, \( \hat{\theta}_{t|t} \) and \( P_{t|t} \), respectively. Note that these functions are of the state trajectory \( x_{1:t} \). Clearly, they also depend on the measurement sequence, but we shall not make that dependence explicit.

The second factor of the integrand in (3), referred to as the smoothing density, is targeted with an SMC sampler. This is done by generating a sequence of weighted particle systems \( \{x^{i}_{1:t}, \omega^{i}_{t}\}_{i=1}^{N} \) for \( t = 1, 2, \ldots \), each defining an empirical point-mass distribution approximating the smoothing distribution at time \( t \) according to

\[ p(dx_{1:t} | y_{1:t}) \approx \sum_{i=1}^{N} \omega^{i}_{t} \delta_{x^{i}_{1:t}} (dx_{1:t}) \]  

where the importance weights \( \{\omega^{i}_{t}\}_{i=1}^{N} \) are normalised to sum to one.

There exist a vast amount of literature, concerning how to generate such particle systems; see e.g. Doucet and Johansen (2011); Gustafsson (2010); Cappé et al. (2007) for an in-
depth treatment. The basic procedure is as follows. Assume that we have generated a weighted particle system \( \{x_{1:t-1}^i, \omega_{i-1}^i\}_{i=1}^N \) targeting the smoothing distribution at time \( t-1 \). We then proceed to time \( t \) by proposing new particles from a (quite arbitrary) proposal kernel \( x_t^i \sim r_t(x_t \mid x_{1:t-1}, y_{1:t}) \) for \( i = 1, \ldots, N \). These samples are appended to the existing particle trajectories, i.e., \( x_{1:t}^i = \{x_{1:t-1}^i, x_t^i\} \). The particles are then assigned importance weights according to

\[
\omega_t^i \propto \omega_{i-1}^i \frac{p(y_t \mid x_{1:t}^i, y_{1:t-1})p(x_{1:t-1}^i \mid x_{1:t-1}^i, y_{1:t-1})}{r_t(x_t^i \mid x_{1:t-1}^i, y_{1:t})},
\]

where the weights are normalised to sum to one. For a CLGSS model, the densities involved in the expression (6) are of known form. In particular, for the model (2) the densities in the numerator are both Gaussian; see Schön et al. (2005) for details. Finally, when the sampling procedure outlined above is iterated over time, it is crucial to complement it with a resampling stage to avoid weight depletion (Cappé et al., 2007). This has the effect of discarding particles with low weights and duplicating particles with high weights.

As indicated by (5), the SMC sampler does in fact generate weighted particle trajectories targeting the smoothing density \( p(x_t \mid y_{1:t}) \). However, due to the consecutive resampling steps, the particle trajectories will suffer from degeneracy; see e.g. Cappé et al. (2007). This means that the SMC method underlying the RBPF in general only can provide good approximations of the marginal filtering density \( p(x_t \mid y_{1:t}) \), or a fixed-lag smoothing density with a short enough lag. Hence, we are not able to provide any good approximation of the smoothing density, which in turn means that we do not have all the components required to approximate the joint filtering density by using (3).

To get around this, one often relies on the mixing of the system. More precisely, the linear state at time \( t \) is supposed to be more or less independent of \( x_{t-\ell} \), if the lag \( \ell \) is large enough. If this is the case, we can obtain an accurate representation of the linear states despite the degeneracy of the nonlinear particle trajectories. Clearly, the success of this approach heavily depends on how good the mixing assumption is. In our case, where the linear state is static, the dependence of \( \theta_t \mid \{x_{1:t}, y_{1:t}\} \) on \( \{x_s, s \leq t - \ell\} \) can be substantial. That is, if the approximation of the density \( p(x_{1:t-\ell} \mid y_{1:t}) \) is poor, using (3) to compute the joint filtering density can give very poor results. We illustrate the RBPF degeneracy problem in Example 1.

---

**Example 1: RBPF for a partially static system**

The first order LGSS system,

\[
x_{t+1} = ax_t + v_t, \quad v_t \sim \mathcal{N}(0, 0.1),
\]

\[
y_t = x_t + e_t, \quad e_t \sim \mathcal{N}(0, 1),
\]

where \( a = -0.8 \), is simulated for \( T = 10000 \) time steps. We wish to estimate the parameter \( a \) recursively and we therefore augment the state with \( \theta_t \equiv a \). The prior distribution of \( a \) is taken as Gaussian with mean \( \theta_{1|0} = 1 \) and covariance \( \hat{P}_{1|0} = 3 \). A bootstrap RBPF using \( N = 100 \) particles is applied to the data and the results is shown in Figure 1.

The initial distribution of \( \theta_1 \) should have negligible effect at time \( T = 10000 \). Hence, we
expect the estimate to converge to the “true” value $-0.8$. As can be seen in the figure, this is not the case. Also, the estimated confidence interval is way too small, i.e. the covariance of the linear state is underestimated. The intuitive explanation is that the RBPF particle trajectories, in some sense, degenerate faster than the estimate of $\theta_t$ converges. As a comparison, we also show the estimate provided by the proposed method, given in the consecutive section\footnote{We emphasise that this example is provided as an illustration of the concept, and not an evaluation of the proposed method. We have only considered one realisation of data, which of course is not enough to draw any general conclusions. A more rigorous numerical evaluation is given in Section 4.}.

3 A non-degenerate RBPF for models with static parameters

In this paper, we propose an alternative to (3), which is to factorise the joint filtering density as,

$$p(x_t, \theta_t \mid y_{1:t}) = p(\theta_t \mid x_t, y_{1:t})p(x_t \mid y_{1:t}).$$

(7)

The marginal filtering density $p(x_t \mid y_{1:t})$ can be approximated using SMC without suffering from degeneracy. Thus, an approximation of the joint filtering density based on the factorisation (7), does not rely as strongly on the mixing properties of the system. However, as opposed to $p(\theta_t \mid x_{1:t}, y_{1:t})$ given in (4), the density

$$p(\theta_t \mid x_t, y_{1:t}),$$

(8)

is in general non-Gaussian and intractable. The problem we face is thus to find an appropriate way to approximate (8), while still enjoying the benefits of a Rao-Blackwellised setting.

This approach resembles an RBPF, but it is based on the marginal density $p(x_t, \theta_t \mid y_{1:t})$ rather than on the density $p(x_{1:t}, \theta_t \mid y_{1:t})$. Hence, it will be referred to as the Rao-Blackwellised marginal particle filter (RBMPF). We start the presentation of the RBMPF...
with a discussion on how to sample from the marginal filtering density. After this, we turn to the more central problem of approximating the conditional filtering density (8).

3.1 Sampling from the marginals

As indicated by (7), we wish to target the marginal filtering density \( p(x_t \mid y_{1:t}) \) with an SMC sampler. In fact, one way to do this is to perform the sampling exactly as in the RBPF, and then simply discard the particle trajectories up to time \( t - 1 \). However, here we outline a different approach, inspired by the marginal particle filter (MPF) (Klaas et al., 2005).

Assume that we have completed the sampling at time \( t - 1 \). We have thus generated a weighted particle system \( \{x_{t-1}^j, \omega_{t-1}^j \}_{j=1}^N \) targeting \( p(x_{t-1} \mid y_{1:t-1}) \). Similarly to Klaas et al. (2005)\(^2\), we then construct a proposal as a mixture density

\[
    r_t'(x_t \mid y_{1:t}) = \sum_{j=1}^N \omega_{t-1}^j r_t(x_t \mid x_{t-1}^j, y_{1:t}), \tag{9}
\]

from which we draw a set of new particles \( \{x_t^i\}_{i=1}^N \). To compute the importance weights, i.e. the quotient between the target and the proposal densities, we note that the target density can be expanded according to,

\[
p(x_t \mid y_{1:t}) = \int \frac{p(y_t \mid x_{t-1:t}, y_{1:t-1}) p(x_t \mid x_{t-1}, y_{1:t-1})}{p(y_t \mid y_{1:t-1})} p(x_{t-1} \mid y_{1:t-1}) dx_{t-1}. \tag{10}
\]

By approximating \( p(x_{t-1} \mid y_{1:t-1}) \), using the weighted particles given at time \( t - 1 \), we get

\[
p(x_t \mid y_{1:t}) \approx \sum_{j=1}^N \omega_{t-1}^j \frac{p(y_t \mid x_{t-1}^j, x_t, y_{1:t-1}) p(x_t \mid x_{t-1}^j, y_{1:t-1})}{p(y_t \mid y_{1:t-1})}. \tag{11}
\]

Using the above approximation, we can compute the importance weights according to,

\[
    \omega_t^i \propto \frac{\sum_{j=1}^N \omega_{t-1}^j p(y_t \mid x_{t-1}^j, x_t^i, y_{1:t-1}) p(x_t \mid x_{t-1}^j, y_{1:t-1})}{\sum_{j=1}^N \omega_{t-1}^j r_t(x_t^i \mid x_{t-1}^j, y_{1:t})}, \tag{12}
\]

where the weights are normalised to sum to one.

3.2 Gaussian mixture approximation

We now turn to the more central problem in the RBMPF, namely to find an approximation of the density (8). The general idea that we will employ is to approximate it as Gaussian. Hence, let us assume that, for some \( t \geq 2 \),

\[
p(\theta_{t-1} \mid x_{t-1}, y_{1:t-1}) \approx \hat{p}(\theta_{t-1} \mid x_{t-1}, y_{1:t-1})
\]

\[
\triangleq \mathcal{N}\left(\theta_{t-1}; \bar{\theta}_{t-1|t-1}(x_{t-1}), P_{t-1|t-1}(x_{t-1})\right), \tag{13}
\]

\(^2\)The difference is that Klaas et al. (2005) targets \( p(\xi_t \mid y_{1:t}) \) rather than \( p(x_t \mid y_{1:t}) \).
for some mean and covariance functions, \( \bar{\theta}_{t-1|t-1} \) and \( P_{t-1|t-1} \), respectively. At time \( t = 2 \), no approximation is needed, since (13) then coincides with (4).

Just as in the standard RBPF, if we augment the conditioning on the nonlinear state to \( x_{t-1:t} \), and make a time update and a measurement update of (13), we obtain

\[
\hat{p}(\theta_t \mid x_{t-1}, x_t, y_{1:t}) = N \left( \bar{\theta}_t; \hat{\theta}_{t|t}(x_{t-1:t}), \hat{P}_{t|t}(x_{t-1:t}) \right),
\]

(14)

for some mean and covariance functions, \( \hat{\theta}_{t|t} \) and \( \hat{P}_{t|t} \), respectively. The problem is that once we “remove” the conditioning on \( x_{t-1} \), the Gaussianity is lost. Hence, to obtain a recursion, i.e. to end up with (13) with time index \( t - 1 \) replaced by \( t \), we need to find a Gaussian approximation of \( p(\theta_t \mid x_t, y_{1:t}) \) based on (14).

To achieve this, we start by noting that the sought density (8) can be written,

\[
p(\theta_t \mid x_t, y_{1:t}) = \int p(\theta_t \mid x_{t-1:t}, y_{1:t})p(x_{t-1} \mid x_t, y_{1:t}) \, dx_{t-1}
= \int p(\theta_t \mid x_{t-1:t}, y_{1:t}) \frac{p(y_t \mid x_{t-1:t}, y_{1:t=1})p(x_{t} \mid x_{t-1:t}, y_{1:t=1})}{p(x_{t} \mid y_{1:t=1})} \, dx_{t-1}.
\]

(15)

At time \( t - 1 \), we have acquired a weighted particle system \( \{x^j_{t-1}, \omega^j_{t-1}\}_{j=1}^N \) targeting the marginal filtering density \( p(x_{t-1} \mid y_{1:t-1}) \). By plugging this into (15), conditioned on \( x^i_t \), we obtain,

\[
p(\theta_t \mid x^i_t, y_{1:t}) \approx \sum_{j=1}^N \gamma^j_{t,i} p(\theta_t \mid x^j_{t-1}, x^i_t, y_{1:t}),
\]

(16a)

with,

\[
\gamma^j_{t,i} \triangleq \frac{\omega^j_{t-1} p(y_t \mid x^j_{t-1}, x^i_t, y_{1:t=1})p(x^j_{t} \mid x^j_{t-1}, y_{1:t=1})}{\sum_{k=1}^N \omega^k_{t-1} p(y_t \mid x^k_{t-1}, x^i_t, y_{1:t=1})p(x^k_{t} \mid x^k_{t-1}, y_{1:t=1})}.
\]

(16b)

Furthermore, by the Gaussianity assumption (14), we see that (16) is a Gaussian mixture model (GMM). Recall that we seek to approximate the left hand side of (16a) with a single Gaussian. To keep the full GMM representation is generally not an option, since this would result in a mixture with a number of components increasing exponentially over time. Hence, we propose to approximate the GMM with a single Gaussian, by using moment matching. From (14), the mean and covariance of the GMM (16a) are given by,

\[
\bar{\theta}_t \triangleq \sum_{j=1}^N \gamma^j_{t,i} \hat{\theta}^j_{t|i},
\]

(17a)

\[
P_t(x^i_t) = \sum_{j=1}^N \gamma^j_{t,i} \left( \hat{P}^j_{t|i} + (\hat{\theta}^j_{t|i} - \bar{\theta}_t)(\hat{\theta}^j_{t|i} - \bar{\theta}_t)^T \right),
\]

(17b)

respectively. Here we have used the shorthand notation \( \hat{\theta}^j_{t|i} \) instead of \( \hat{\theta}_{t|t}(x^j_{t-1}, x^i_t) \), etc. In conclusion, the above results provide a Gaussian approximation of (8) according to,

\[
\hat{p}(\theta_t \mid x^i_t, y_{1:t}) \triangleq N \left( \bar{\theta}_t; \hat{\theta}_{t|t}(x^i_t), P_t(x^i_t) \right).
\]
3.3 Resulting Algorithm

The procedure outlined in the previous two sections provides an RBPF-like method targeting the filtering density using the factorisation (7). To be able to carry out the steps of this method, we require

\[ p(y_t \mid x_{t-1}, x_t, y_{1:t-1}) \text{ and } p(x_t \mid x_{t-1}, y_{1:t-1}) \]

to be available for evaluation, since these are used in (12) and in (16b), to compute the particle weights and the mixing weights, respectively (note the similarity between the two expressions).

Strictly speaking, these densities are not analytically tractable in the general case, since we condition on just \( x_{t-1:t} \) and not the full nonlinear state trajectory \( x_{1:t} \) (cf. the densities appearing in the RBPF weight expression (6), which are tractable for any CLGSS model).

However, this will in fact not be an issue in the RBMPF setting. The reason is that the conditional filtering density for the linear state is approximated by a Gaussian at time \( t-1 \), according to (13). Given this approximation, conditioning on just \( x_{t-1} \) in the RBMPF, will have the “same effect” as conditioning on \( x_{1:t-1} \) in the RBPF. Hence, the densities appearing in (12) and (16b) will indeed be available for evaluation, under this Gaussianity approximation (see Lindsten (2011) for details). It can be said that the whole idea with the RBMPF, is to replace the conditioning on the nonlinear state trajectory, with a conditioning on the nonlinear state at a single time point. We summarise the RBMPF method in Algorithm 1.

Algorithm 1 RBMPF (one time step)

1: Sample particles, \( \{x_i^t\}_{i=1}^N \) from (9).
2: for \( i = 1 \) to \( N \) do
3:     for \( j = 1 \) to \( N \) do
4:         Compute the mean \( \hat{\theta}_{j,i}^{t-1} \) and the covariance \( \hat{P}_{j,i}^{t-1} \) of the density (14), conditioned on \( \{x_j^{t-1}, x_i^t\} \), using RBPF time and measurement updates.
5:     Compute the mixture weights \( \gamma_{j,i}^t \) according to (16b). The involved densities are available from the RBPF time and measurement updates.
6: end for
7: Compute the mean \( \bar{\theta}_{i}^{t} \) and covariance \( \bar{P}_{i}^{t} \) of the GMM according to (17).
8: Compute the importance weights \( \{\omega_i^t\}_{i=1}^N \) according to (12).
9: end for

4 Numerical illustration

In this section we evaluate the RBMPF method for recursive identification on simulated data. We will consider the first order nonlinear system,

\[
\begin{align*}
x_{t+1} &= ax_t + b \frac{x_t}{1 + x_t^2} + c \cos(1.2t) + v_t, \quad (18a) \\
y_t &= dx_t^2 + e_t,
\end{align*}
\]

with \( v_t \sim \mathcal{N}(0, 0.01) \) and \( e_t \sim \mathcal{N}(0, 0.1) \). The initial state of the system is \( x_1 \equiv 0 \). The true parameters are given by \( \theta^* = (a \ b \ c \ d)^T = (0.5 \ 25 \ 8 \ 0.05)^T \). This system has been studied e.g. by Gordon et al. (1993) and has become something of a
benchmark example for nonlinear filtering. By augmenting the model with the static parameter state $\theta_t \equiv (a \ b \ c \ d)^T$, we obtain a fifth order mixed linear/nonlinear system, where four of the states are conditionally linear. We can thus employ the RBMPF given in Algorithm 1 for recursive parameter estimation.

The RBMPF is compared with the RBPF, where the latter uses jittering noise as discussed in Remark 1. As suggested by Schön and Gustafsson (2003), we apply Gaussian jittering noise with decaying variance on both states and parameters. These artificial noise sources are Gaussian with time-decaying variances, $\sigma^2_x/t$ and $(\sigma^2_\theta/t)I_4$, respectively. Of course, the jittering noises are internal to the RBPF and are not used when simulating data from the system.

The evaluation was made by a Monte Carlo study over 100 realisations of data $y_{1:T}$ from the system (18), each consisting of $T = 200$ measurements. The parameters were modeled as Gaussian random variables, $\theta_1 \sim N(\bar{\theta}_{1|0}, \text{diag}(0.5, 25, 8, 0.05))$. Here, $\bar{\theta}_{1|0}$ corresponds to the prior mean of the parameter. This vector was chosen randomly for each Monte Carlo simulation, uniformly over the intervals $\pm 50\%$ from the true parameter values. The RBMPF and four versions of the RBPF were run in parallel, all using $N = 500$ particles. The first RBPF did not use any jittering noise, whereas the remaining three versions used jittering noise with $(\sigma^2_x = \sigma^2_\theta = \sigma^2_x, \sigma^2 = 0.01, \sigma^2 = 0.1$ and $\sigma^2 = 1$, respectively. Furthermore, to increase the numerical robustness, a weight threshold was implemented in the filters. That is, if the sum of the unnormalised importance weights was below a certain threshold, here $10^{-12}$, the particles were discarded, the filter “rewinded” a few time step and new particles were generated.

Table 1 summarises the results from the different filters, in terms of the Monte Carlo means and standard deviations for the parameter estimates extracted at the final time point $t = T = 200$. Also, the rightmost column of the table shows the percentage of data realisations, in which the weight threshold (as mentioned above) was hit at least once. We conclude that jittering noise with $\sigma^2 = 0.1$ provides the best tuning for the RBPF, among the values considered here. The results from this filter and from the RBMPF, over the 100 realisations of data, are given in Figures 2 and 3. It is clear that the jittering noise in the RBPF introduces extra variance to the estimates and also that it slows down the convergence, when compared to the RBMPF. Furthermore, from Table 1 we see that the accuracy of the RBPF is highly dependent on the variance of the jittering. Tuning of this parameter can be problematic in a real world scenario. The absence of a jittering noise which needs to be tuned properly, is one of the main advantages with the RBMPF over the RBPF.

5 Discussion and future work

One of the main drawbacks with the RBMPF method is that it has quadratic complexity in the number of particles, as opposed to the RBPF, which has linear complexity. In fact, just as the RBPF can be seen as using $N$ parallel Kalman filters, the RBMPF uses $N^2$ Kalman filters. In this way, by viewing each particle as a separate model, the RBMPF very much resembles the 2nd order, generalised pseudo-Bayesian (GPB2) multiple model filter. Guided by this insight, we could also derive an RBMPF similar to the
Table 1: Monte Carlo means and standard deviations

<table>
<thead>
<tr>
<th>Method</th>
<th>a ($\times 10^{-1}$)</th>
<th>b</th>
<th>c</th>
<th>d ($\times 10^{-2}$)</th>
<th>Rew.</th>
</tr>
</thead>
<tbody>
<tr>
<td>True value ($\theta^{\star}$)</td>
<td>5</td>
<td>25</td>
<td>8</td>
<td>5</td>
<td>52</td>
</tr>
<tr>
<td>RBPF ($\sigma^2 = 0$)</td>
<td>4.99 ± 0.12</td>
<td>24.9 ± 3.84</td>
<td>7.92 ± 0.61</td>
<td>5.26 ± 1.06</td>
<td>52</td>
</tr>
<tr>
<td>RBPF ($\sigma^2 = 0.01$)</td>
<td>4.95 ± 0.09</td>
<td>24.6 ± 1.60</td>
<td>7.95 ± 0.26</td>
<td>5.21 ± 0.44</td>
<td>4</td>
</tr>
<tr>
<td>RBPF ($\sigma^2 = 0.1$)</td>
<td>4.93 ± 0.48</td>
<td>22.7 ± 0.91</td>
<td>7.60 ± 0.20</td>
<td>5.84 ± 0.67</td>
<td>1</td>
</tr>
<tr>
<td>RBPF ($\sigma^2 = 1$)</td>
<td>4.73 ± 0.29</td>
<td>18.8 ± 0.95</td>
<td>6.45 ± 0.30</td>
<td>8.60 ± 0.66</td>
<td>0</td>
</tr>
<tr>
<td>RBMPF</td>
<td>5.00 ± 0.03</td>
<td>25.2 ± 1.00</td>
<td>8.05 ± 0.15</td>
<td>4.94 ± 0.24</td>
<td>11</td>
</tr>
</tbody>
</table>

GPB1 filter (see Bar-Shalom et al. (2001) for the two GPB filters). This would reduce the complexity to grow linearly with $N$, but at the cost of coarser approximations, likely to degrade the performance of the filter. A third approach in this direction, is to start from the interacting multiple model (IMM) filter by Blom and Bar-Shalom (1988), which is a popular multiple model filter, since it has lower complexity than GPB2 (still quadratic, but smaller constants), but is known to have similar performance (Blom and Bar-Shalom, 1988). However, it is not clear that the ideas underlying the IMM filter, can be straightforwardly generalised to the RBMPF. This issue requires further attention.

Another way to reduce the complexity of the algorithm is by numerical approximations of the mixture models. Due to the exponential decay of the Gaussian components, truncation might aid in making fast, sufficiently accurate, evaluations of the GMM moments. A related approach, which could be adapted to the RBMPF, is used by Gray and Moore (2001, 2003) for fast, nonparametric density estimation. Also, fast summation methods, similar to the ideas underlying the fast Gauss transform by Greengard and Strain (1991); Greengard and Sun (1998), might be of use. However, as discussed by Boyd (2010), truncation methods should in general have more to offer than fast summation methods, for Gaussian components which are quickly decaying.

Finally, another option is of course to seek alternative approximations of the conditional filtering density (8), not based on a GMM as in (16). By doing so, one can possibly find good approximations, which can be evaluated more efficiently than the ones presented here.

6 Conclusions

The application of particle filters (PFs) for estimating static parameters is a well known and challenging problem. For models where the parameters enter linearly, they can be marginalised out analytically, by using conditional Kalman filters, leading to the Rao-Blackwellised particle filter (RBPF). However, this will not remedy the problem, as the degeneracy of the particle trajectories in the RBPF will result in erroneous parameter estimates. This issue can be traced back to the expression of the filtering density (3), which is the basis for the RBPF. When this form is used to approximate the filtering density, good accuracy is obtained only when the model under study is mixing sufficiently fast; this is not the case when the state is augmented with static parameters. Here, we have proposed
a different factorisation of the filtering density, according to (7). By using a particle representation of the marginal filtering distribution for the nonlinear state, a Gaussian mixture arises as the natural approximation of the conditional filtering density for the parameters. To obtain a recursive method, we propose to approximate this mixture density with a single Gaussian, by using moment matching. This results in an RBPF-like method, suitable for recursive identification of nonlinear dynamical systems with affine parameter dependence. The main drawback with the proposed method is its computational complexity, which grows quadratically with the number of particles. However, we believe that this can be reduced significantly by using truncation techniques, motivated by the exponential decay of the Gaussian components.
Figure 2: Estimates of the parameters $a$, $b$, $c$ and $d$ (from top to bottom) for the RBPF using jittering noise with $\sigma^2_x = \sigma^2_\theta = 0.1$. The blue lines illustrate the estimates for the 100 realisations of data. The true parameter values are shown as thick black lines.

Figure 3: Estimates of the parameters $a$, $b$, $c$ and $d$ (from top to bottom) for the RBMPF. The blue lines illustrate the estimates for the 100 realisations of data. The true parameter values are shown as thick black lines.


An explicit variance reduction expression for the Rao-Blackwellised particle filter

Authors: Fredrik Lindsten, Thomas B. Schön and Jimmy Olsson

Edited version of the paper:

An explicit variance reduction expression for the Rao-Blackwellised particle filter

Fredrik Lindsten*, Thomas B. Schön* and Jimmy Olsson**

*Dept. of Electrical Engineering, Linköping University, SE–581 83 Linköping, Sweden
{lindsten,schon}@isy.liu.se

**Center of Mathematical Sciences, Lund University, SE–221 00 Lund, Sweden
jimmy@maths.lth.se

Abstract

Particle filters (PFs) have been shown to be very potent tools for state estimation in nonlinear and/or non-Gaussian state-space models. For certain models, containing a conditionally tractable substructure (typically conditionally linear Gaussian or with finite support), it is possible to exploit this structure in order to obtain more accurate estimates. This has become known as Rao-Blackwellised particle filtering (RBPF). However, since the RBPF is typically more computationally demanding than the standard PF per particle, it is not always beneficial to resort to Rao-Blackwellisation. For the same computational effort, a standard PF with an increased number of particles, which would also increase the accuracy, could be used instead. In this paper, we have analysed the asymptotic variance of the RBPF and provide an explicit expression for the obtained variance reduction. This expression could be used to make an efficient discrimination of when to apply Rao-Blackwellisation, and when not to.

1 Introduction and related work

Many important problems in various fields of science are related to state estimation in general state-space models, based on noisy observations. If a prior distribution is assumed for the initial state, the optimal filter is given by the Bayesian filtering recursions. In a few special cases, basically for linear Gaussian state-space (LGSS) models and finite state-space models, the optimal filtering problem is analytically tractable. However, many interesting problems do not exhibit such nice properties, but are both nonlinear and/or non-Gaussian. In these cases, the optimal filter needs to be approximated in some way. Sequential Monte Carlo (SMC) methods, or particle filters (PFs), have shown to be very powerful tools when addressing such intractable models. Since the introduction of the PF by Gordon et al. (1993), we have experienced a vast amount of research in the area. For instance, many improvements and extensions have been introduced to increase the accuracy of the filter, see e.g. Doucet and Johansen (2011) for an overview of recent developments.
One natural idea is to exploit any tractable substructure in the model, see e.g. Doucet et al. (2000b); Schön et al. (2005). More precisely, if the model, conditioned on one partition of the state, behaves like e.g. an LGSS or a finite state-space model it is sufficient to employ particles for the intractable part and make use of the analytic tractability for the remaining part. Inspired by the Rao-Blackwell theorem, this has become known as the Rao-Blackwellised particle filter (RBPF).

The motivation for the RBPF is to improve the accuracy of the filter, i.e. any estimator derived from the RBPF will intuitively have lower variance than the corresponding estimator derived from the standard PF. Informally, the reason for this is that in the RBPF, the particles are spread in a lower dimensional space, yielding a denser particle representation of the underlying distribution. The improved accuracy is also something that is experienced by practitioners. However, it can be argued that it is still not beneficial to resort to Rao-Blackwellisation in all cases. The reason is that the RBPF is typically more computationally expensive per iteration, compared to the standard PF (e.g. for an RBPF targeting a conditional LGSS model, each particle is equipped with a Kalman filter, all which needs to be updated at each iteration). Hence, for a fixed computational effort, we can choose to either use Rao-Blackwellisation or to run a standard PF, but instead increase the number of particles. Both these alternatives will reduce the variance of the estimators. Hence, it is important to understand and to be able to quantify how large variance reduction we can expect from the RBPF, in order to make suitable design choices for any given problem.

In this paper we shall study the asymptotic (in the number of particles) variances for the RBPF and the standard PF. We provide an explicit expression for the difference between the variance of an estimator derived from the PF and the variance of the corresponding estimator derived from the RBPF. Doucet et al. (2000b) motivates the RBPF by concluding that the weight variance will be lower than for the PF, but they do not consider the variances of any estimators. This is done by Chopin (2004), who, under certain assumptions, concludes that the variance of an estimator based on the PF always is at least as high as for the RBPF. However, no explicit expression for the difference is given, and the test functions considered are restricted to certain dimensions of the state-space. Doucet et al. (2000a) also analyse the RBPF and the reduction of asymptotic variance. However, they only consider an importance sampling setting and neglect the important resampling step. Karlsson et al. (2005) studies the problem empirically, by running simulations on a specific example. Here, they have also analysed the number of computations per iteration in the RBPF and the PF, respectively.

The rest of this paper is organised as follows. In Section 2 we present the SMC framework and define the PF and the RBPF. We thereafter introduce the natural estimators that can be derived from the two filters, and discuss their asymptotic properties in Section 3. The main result is given in Section 4, where we provide an explicit expression for the difference in asymptotic variance for the two estimators. In Section 5 this expression is studied for a special case and in Section 6 we discuss how it can be used in the design of an SMC sampler. Finally, in Section 7 we draw conclusions.
2 Background

2.1 Notation

All random variables are defined on a common probability space \((\Omega, \mathcal{F}, \mathbb{P})\). For a measurable space \((\mathcal{X}, \mathcal{X})\), we denote by \(\mathbb{F}(\mathcal{X})\) the set of all \(\mathcal{X}\)-measurable functions from \(\mathcal{X}\) to \(\mathbb{R}\). For a measure \(\mu\) on \(\mathcal{X}\) and \(f \in \mathbb{F}(\mathcal{X})\), satisfying \(\int |f| d\mu < \infty\), we denote by \(\mu(f)\) the integral \(\int f d\mu\). For \(p \geq 1\), \(L^p(\mathcal{X}, \mu)\) denotes the set of functions \(f \in \mathbb{F}(\mathcal{X})\) such that \(\int |f|^p d\mu < \infty\).

Let \((\mathcal{X}, \mathcal{X})\) and \((\mathcal{Y}, \mathcal{Y})\) be two measurable spaces. A kernel \(V\) from \(\mathcal{X}\) to \(\mathcal{Y}\) is a map \(V : (\mathcal{X}, \mathcal{Y}) \rightarrow \mathbb{R}_+\), such that (i) for each \(x \in \mathcal{X}\), the map \(A \mapsto V(x, A)\) is a measure on \(\mathcal{Y}\) (ii) for each \(A \in \mathcal{Y}\), the map \(x \mapsto V(x, A)\) is \(\mathcal{X}\)-measurable. A kernel is called a transition kernel if \(V(x, Y) = 1\) for any \(x \in \mathcal{X}\). We shall sometimes write \(V(A | x)\) instead of \(V(x, A)\). With \(f : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}\), \(f(x, \cdot) \in L^1(\mathcal{Y}, V(x, \cdot))\) we let \(V(f)\) denote the function \(V(f)[x] = \int f(x, y)V(x, dy)\).

For two measures \(\mu\) and \(\nu\), we say that \(\nu\) is absolutely continuous with respect to \(\mu\) (written \(\nu \ll \mu\)) if \(\mu(A) = 0 \Rightarrow \nu(A) = 0\). By \(\mathcal{N}(m, \sigma^2)\) we denote the Gaussian distribution with mean \(m\) and variance \(\sigma^2\). Finally, convergence in distribution and convergence in probability are denoted by \(\xrightarrow{D}\) and \(\xrightarrow{P}\), respectively.

2.2 Particle filtering

Let \(\{X_t, t \in \mathbb{N}\}\) be a discrete-time Markov process on the state-space \((\mathcal{X}, \mathcal{X})\) (typically some power of the real line with the corresponding Borel \(\sigma\)-algebra). The process is hidden, but observed through the measurement sequence \(\{Y_t, t \in \mathbb{N}\}\) defined on \((\mathcal{Y}, \mathcal{Y})\). For a given sequence of measurements \(Y_{1:t} \triangleq (Y_1, \ldots, Y_t)\) (a similar notation shall be used for other sequences as well), the joint smoothing distribution is defined as

\[
\phi_t(A) \triangleq \mathbb{P}(X_{1:t} \in A \mid Y_{1:t}), \quad A \in \mathcal{X}^t. \tag{1}
\]

In particle filtering, we seek to approximate this distribution by a weighted particle system \(\{x_{1:t}^N, w_{1:t}^N\}_{i=1}^N\). Observe that this is a triangular array of random variables and, though cumbersome, it is important to keep the dependence on \(N\) in the notation. There are many different ways to generate such a particle system (see e.g. Doucet and Johansen (2011) for an overview), but common to most are the concepts of sequential importance sampling and resampling. Importance sampling is used to propagate a weighted sample, targeting the smoothing distribution at time \(t - 1\), into a weighted sample targeting the smoothing distribution at time \(t\). This is done by sampling new particles from a proposal kernel (a transition kernel from \(X_{t-1}\) to \(X_t\)),

\[
R_{t-1}(\tilde{x}_{1:t-1}, dx_{1:t}). \tag{2}
\]

The proposal kernel is chosen such that \(\phi_t \ll \pi_t\), where we have defined the probability measure on \(\mathcal{X}^t\),

\[
\pi_t(dx_{1:t}) \triangleq \int_{\mathcal{X}_{t-1}} \phi_{t-1}(d\tilde{x}_{1:t-1})R_{t-1}(\tilde{x}_{1:t-1}, dx_{1:t}). \tag{3}
\]
Besides from the absolute continuity condition, we shall not assume any restrictions on the choice of proposal kernel. For instance, it can depend on the measurement sequence, but we shall not make such dependence explicit. The distribution $\pi_t$ can be seen as the “proposed smoothing distribution” at time $t$ under the proposal kernel $R_{t-1}$.

To compensate for the fact that we sample from the wrong distribution, the samples are weighted. For this purpose we introduce the weight function

$$W_{t-1}(x_{1:t}) \triangleq \frac{d\phi_t}{d\pi_t}(x_{1:t}). \quad (4)$$

A well known problem in particle filtering is weight depletion, see e.g. Doucet et al. (2000b); Doucet and Johansen (2011). To remedy this the particle system can be resampled, i.e. particles with high weights are duplicated and particles with low weights are discarded. As previously mentioned, there are many options available to the user when designing a particle filter, e.g. in the choice of resampling scheme, if the number of particles shall be fixed or varying and when the resampling shall be performed. In this paper, the results will be given for the particle filter presented in Algorithm 1 below. Briefly, we consider arbitrary proposal kernels (2) (under the absolute continuity assumption) and multinomial resampling which is performed at each iteration of the algorithm. However, results similar to those presented in Section 4 could be obtained for other types of PFs as well, such as the auxiliary PF (Pitt and Shephard, 1999) and PFs with more sophisticated resampling schemes.

**Algorithm 1** Particle filter

**Input:** A weighted sample $\{x_{1:t-1}^N, w_{t-1}^N\}_{i=1}^N$ targeting $\phi_{t-1}$.

**Output:** A weighted sample $\{x_{1:t}^N, w_t^N\}_{i=1}^N$ targeting $\phi_t$.

1. **Resampling:** Sample $N$ indices from a discrete distribution, i.e. for $i = 1, \ldots, N$, $\mathbb{P}(I_{N,i} = j \mid \{x_{1:t-1}^N, w_{t-1}^N\}_{k=1}^N) = w_{N,j}^t / \sum_{l=1}^N w_{N,l}^t$. Set $\tilde{x}_{1:t-1}^N \sim x_{1:t-1}^N$, $i = 1, \ldots, N$. The equally weighted sample $\{\tilde{x}_{1:t-1}^{N,i}, 1\}_{i=1}^N$ targets $\phi_{t-1}$.

2. **Importance sampling:** Choose a proposal kernel according to (2). Sample new particles according to $x_{1:t}^N, i \sim R_{t-1}(\tilde{x}_{1:t-1}^N, dx_{1:t})$ for $i = 1, \ldots, N$. Compute the weights, using the weight function (4), $w_t^N = W_{t-1}(x_{1:t}^N)$.

The SMC sampler presented in this section, i.e. which targets the “full” joint smoothing distribution (1), will throughout the remainder of this paper be denoted the particle filter (PF).

### 2.3 Rao-Blackwellised particle filter

The idea underlying the Rao-Blackwellised particle filter is to exploit any tractable substructure in the targeted Markov process, if such a structure is present. In doing so, one can hope to obtain better particle approximations for a fixed number of particles. Hence, assume that each $X_t$ can be partitioned according to $X_t = \{\Xi_t, Z_t\}$ and $X = X_{\xi} \times X_z$. Furthermore, assume that $\phi_t$ factorises according to

$$\phi_t(dx_{1:t}) = \phi^m_t(dx_{1:t})\phi^z_t(dx_{1:t} \mid \xi_{1:t}), \quad (5)$$
where \{\xi_t, z_t\} identifies to \(x_t\). Here, \(\phi_t^p\) is the marginal smoothing distribution of \(\Xi_{1:t}\) and \(\phi_t^c\) is the conditional smoothing distribution of \(Z_{1:t}\) given \(\Xi_{1:t} = \xi_{1:t}\). The conditional distribution is assumed to be analytically tractable, typically Gaussian or with finite support.

**Remark 1.** More precisely, \(\phi_t^c\) is a kernel from \(X_t^c\) to \(X_t^c\). For each fixed \(\xi_{1:t}\), \(\phi_t^c(\cdot \mid \xi_{1:t})\) is a measure on \(X_t^c\), and it can hence be viewed as a conditional distribution. In the notation introduced in (5), the meaning is that \(\phi_t^c\) is the product of the measure \(\phi_t^m\) and the kernel \(\phi_t^c\). In the remainder of this paper we shall make frequent use of a Fubini like theorem for such products, see e.g. Uglanov (1991).

Instead of running the PF targeting the “full” joint smoothing distribution (1), we have the option to target the marginal distribution \(\phi_t^m\) with an SMC sampler and then make use of an analytical expression for \(\phi_t^c\). Hence, we choose a proposal kernel \(R_{t-1}^m(\xi_{1:t-1}, d\xi_{1:t})\) (from \(X_{t-1}^c\) to \(X_t^c\)) such that \(\phi_t^m \ll \pi_t^m\) and define a weight function \(W_t^m(\xi_{1:t})\) analogously to (4). The measure \(\pi_t^m\) is defined analogously to (3).

A weighted particle system \(\{\xi_{1:t}^i, \omega_{t}^i \}_{i=1}^N\), targeting \(\phi_t^m\), can then be generated in the same manner as in Algorithm 1. We simply replace \(\{x_{1:t}^i, w_{t}^i\}_{i=1}^N\) with \(\{\xi_{1:t}^i, \omega_{t}^i\}_{i=1}^N\) and \(\phi_{\cdot\cdot}, R_{\cdot\cdot}, W_{\cdot\cdot}\) with \(\phi_t^m, R_t^m, W_t^m\), respectively (again, superscript \(m\) for marginal). This will be referred to as the Rao-Blackwellised particle filter (RBPF).

**Remark 2.** The most common way to present the RBPF is for conditional LGSS models. In this case, the conditional distribution \(\phi_t^c\) is Gaussian, which means that it can be computed using the Kalman filter recursions. Consequently, the Kalman filter updates are often shown as intrinsic steps in the presentation of the RBPF algorithm, see e.g. Schön et al. (2005). In this paper, we adopt a more general view and simply see the RBPF as an SMC sampler targeting the marginal distribution \(\phi_t^m\). We then assume that the conditional distribution \(\phi_t^c\) is available by some means (for the conditional LGSS case, this would of course be by the Kalman filter), but it is not important for our results what those means are.

## 3 Problem formulation

The PF and the RBPF can both be used to estimate expectations under the joint smoothing distribution. Assume that we, for some function \(f \in L^1(\mathbb{X}^t, \phi_t)\), seek the expectation \(\phi_t(f)\). For the PF we use the natural estimator,

\[
f_{PF}^N \triangleq \sum_{i=1}^N \frac{w_{t}^N \cdot f(x_{1:t}^i)}{\sum_{j=1}^N w_{t}^N \cdot f(x_{1:t}^j)}.
\]

For the RBPF we use the fact that \(\phi_t(f) = \phi_t^m(\phi_t^c(f))\), and define the estimator,

\[
f_{RBPF}^N \triangleq \sum_{i=1}^N \frac{\omega_{t}^N \cdot f(\xi_{1:t}^i)}{\sum_{j=1}^N \omega_{t}^N \cdot f(\xi_{1:t}^j)}.
\]

The question then arise, how much better is (7) compared to (6)?

One analysis of this question, sometimes seen in the literature, is to simply consider a
decomposition of variance,
\[
\text{Var}(f) = \text{Var}(\mathbb{E}[f | \Xi_{1:t}]) + \mathbb{E}[\text{Var}(f | \Xi_{1:t})].
\]
(8)

Here, the last term is claimed to be the variance reduction obtained in the RBPF. The decomposition is of course valid, the problem is that it does not answer our question. What we have in (8) is simply an expression for the variance of the test function \(f\), it does not apply to the estimators (6) and (7).

**Remark 3.** It is not hard to see why the “simplified” analysis (8) has been considered. If the PF would produce independent and identically distributed (i.i.d.) samples from the target distribution (which it does not), then the analysis would be correct. More precisely, for i.i.d. samples, the central limit theorem states that the asymptotic variance of an estimator of a test function \(f\), coincides with the variance of the test function itself (up to a factor \(1/N\)). However, as we have already pointed out, the PF does not produce i.i.d. samples. This is due to the resampling step, in which a dependence between the particles is introduced. At the end of Section 6, one of the inadequacies of (8) will be pointed out.

Hence, we are interested in the asymptotic variance of (6) and (7), respectively. To analyse this we shall borrow the concept of asymptotic normality from Douc and Moulines (2008).

**Definition G.1 (Asymptotic normality).** Let \((X, \mathcal{X})\) be a measurable space, \(A\) and \(W\) subsets of \(\mathbb{F}(X)\), \(\mu\) a probability measure and \(\gamma\) a finite measure on \(\mathcal{X}\). Let \(\sigma\) be a real nonnegative function on \(A\) and \(\{a_N\}_{N=1}^{\infty}\) a nondecreasing real sequence diverging to infinity. A weighted sample \(\{\chi^{N,i}, v^{N,i}\}_{i=1}^{N}\) is said to be asymptotically normal for \((\mu, A, W, \sigma, \gamma, \{a_N\})\) if
\[
\begin{align*}
    a_N \Omega_N^{-1} \sum_{i=1}^{N} v^{N,i} (f(\chi^{N,i}) - \mu(f)) & \xrightarrow{D} \mathcal{N}(0, \sigma^2(f)), \\
    a_N^2 \Omega_N^{-2} \sum_{i=1}^{N} (v^{N,i})^2 g(\chi^{N,i}) & \xrightarrow{P} \gamma(g), \\
    a_N \Omega_N^{-1} \max_{1 \leq i \leq N} v^{N,i} & \xrightarrow{P} 0,
\end{align*}
\]
(9)
as \(N \to \infty\), for any \(f \in A\) and any \(g \in W\), where \(\Omega_N = \sum_{j=1}^{N} v^{N,j}\).

In the following two theorems (slight modifications of what has previously been given by Douc and Moulines (2008)) we claim asymptotic normality for the weighted particle systems generated by the PF and the RBPF, respectively.

**Theorem 1 (Asymptotic normality of the PF).** Assume that the initial particle system \(\{x_1^{N,i}, w_1^{N,i}\}_{i=1}^{N}\) is asymptotically normal for \((\phi_1, A_1, W_1, \sigma_1, \phi_1, \{\sqrt{N}\})\). Define recursively the sets
\[
\begin{align*}
    A_t & \triangleq \{ f \in L^2(X^t, \phi_t) : R_{t-1}(\cdot, W_{t-1}f) \in A_{t-1}, R_{t-1}(\cdot, W_{t-1}^2f^2) \in W_{t-1} \}, \\
    W_t & \triangleq \{ f \in L^1(X^t, \phi_t) : R_{t-1}(\cdot, W_{t-1}^2|f|) \in W_{t-1} \}.
\end{align*}
\]
Assume that, for any \(t \geq 1\), \(R_t(\cdot, W_{t}^2) \in W_t\). Then, for any \(t \geq 1\), the weighted particle
system \( \{ x_{1:t}^N, \omega_t^N \}_{i=1}^N \) generated by the PF is asymptotically normal for
\[ (\phi_t, A_t, W_t, \sigma_t, \phi_t, \{ \sqrt{N} \}). \]
The asymptotic variance is, for \( f \in A_t \), given by
\[ \sigma_t^2(f) = \sigma_{t-1}^2 \left( R_{t-1} \left( \cdot, W_{t-1} \bar{f} \right) \right) + \phi_{t-1} \left[ R_{t-1} \left( \cdot, (W_{t-1} \bar{f})^2 \right) \right], \]
(10)
\[ \bar{f} = f - \phi_t(f). \]

**Proof:** See Appendix A. \( \square \)

**Theorem 2 (Asymptotic normality of the RBPF).** Under analogous conditions and definitions as in Theorem 1, for any \( t \geq 1 \) the particle system \( \{ \xi_{1:t}^N, \omega_t^N \}_{i=1}^N \) generated by the RBPF is asymptotically normal for
\[ (\phi_t^m, A_t^m, W_t^m, \tau_t, \phi_t^m, \{ \sqrt{N} \}). \]
The asymptotic variance is, for \( g \in A_t^m \), given by
\[ \tau_t^2(g) = \tau_{t-1}^2 \left( R_{t-1} \left( \cdot, W_{t-1} \bar{g} \right) \right) + \phi_{t-1}^m \left[ R_{t-1} \left( \cdot, (W_{t-1} \bar{g})^2 \right) \right], \]
(11)
\[ \bar{g} = g - \phi_t^m(g). \]

**Proof:** See Appendix A. \( \square \)

Recall from Remark 2 that the PF and the RBPF are really just two SMC samplers, targeting different distributions, hence the similarity between the two theorems above. Actually, we could have given one, more general, theorem applicable to both filters. The reason for why we have chosen to present them separately is for clarity and to introduce all the required notation.

As previously pointed out, the RBPF will intuitively produce better estimates than the PF, i.e. we expect \( \tau_t^2(\phi_t^c(f)) \leq \sigma_t^2(f) \). Let us therefore define the variance difference
\[ \Delta_t(f) \triangleq \sigma_t^2(f) - \tau_t^2(\phi_t^c(f)). \]
(12)
The problem that we are concerned with in this paper is to find an explicit expression for this quantity. This will be provided in the next section.

## 4 The main result

To analyse the variance difference (12) we shall need the following assumption (similar to what is used by Chopin (2004)).

**(A1)** For each \( \tilde{\xi}_{1:t-1} \in X_{\xi}^{t-1} \), the two measures
\[ \int_{X_{\xi}^{t-1}} \phi_{t-1}^c(d\tilde{z}_{1:t-1} \mid \tilde{\xi}_{1:t-1})R_{t-1}(\{\tilde{\xi}_{1:t-1}, \tilde{z}_{1:t-1}\}, dx_{1:t}) \]
(13a)
agree on $\mathcal{X}^t$, for some positive function $a_t: \mathcal{X}^t_x \to \mathbb{R}$ and some transition kernel $\pi_t^c$ from $\mathcal{X}^t_x$ to $\mathcal{X}^t_z$, for which $\phi_t^c(\cdot \mid \xi_{1:t}) \ll \pi_t^c(\cdot \mid \xi_{1:t})$.

The basic meaning of this assumption is to create a connection between the proposal kernels $R_{t-1}$ and $R_{m,t-1}$. It is natural that we need some kind of connection. Otherwise the asymptotic variance expressions (10) and (11) would be completely decoupled, and it would not be possible to draw any conclusions from a comparison. Still, as we shall see in the next section, assumption (A1) is fairly weak.

We are now ready to state the main result of this paper.

**Theorem 3.** Under assumption (A1), and using the definitions from Theorem 1 and Theorem 2, for any $f \in \tilde{A}_t$,

$$
\Delta_t(f) = \Delta_{t-1}(R_{t-1}(\cdot, W_{t-1} \bar{f})) + \phi_{t-1}^m \left( R_{t-1}^m \left( \cdot, \left( \frac{1-a_t}{a_t} \right) (W_{t-1}^m \bar{\psi})^2 + a_t \text{Var}_{\pi_t^c}(W_{t-1} \bar{f}) \right) \right),
$$

(14)

where

$$
\bar{\psi} = \phi_t^c(f) - \phi_t(f),
$$

(15a)

$$
\tilde{A}_t \triangleq \{ f \in \mathcal{F}(\mathcal{X}^t) : \phi_t^c(f) \in \mathcal{A}_t^m \} \cap \mathcal{A}_t.
$$

(15b)

**Proof:** See Appendix A.

## 5 Relationship between the proposals kernels

To understand the results given in the previous section, we shall have a closer look at the relationship between the proposal kernels imposed by assumption (A1). We shall do this for a certain family of proposal kernels. More precisely, assume that the kernels can be written

$$
R_{t-1}(\bar{x}_{1:t-1}, dx_{1:t}) = r_{t-1}(dx_t \mid x_{1:t-1}) \delta_{\bar{x}_{1:t-1}}(dx_{1:t-1}),
$$

(16a)

$$
R_{t-1}^m(\bar{\xi}_{1:t-1}, d\xi_{1:t}) = r_{t-1}^m(d\xi_t \mid \xi_{1:t-1}) \delta_{\bar{\xi}_{1:t-1}}(d\xi_{1:t-1}).
$$

(16b)

Informally, this means that when a trajectory $(x_{1:t}^N, i)$ or $\xi_{1:t}^N$ is sampled at time $t$, we keep the “old” trajectory up to time $t-1$ and simply append a sample from time $t$. This is the case for most PFs (when targeting the joint smoothing distribution), but not all, see e.g. the resample-move algorithm by Gilks and Berzuini (2001).

Furthermore, let $r_{t-1}$ be factorised as

$$
r_{t-1}(dx_t \mid x_{1:t-1}) = q_{t-1}^c(dx_t \mid \xi_{1:t}^c, z_{1:t-1}) q_{t-1}^m(d\xi_t \mid \xi_{1:t-1}, z_{1:t-1}).
$$

(16c)
Assume that \( q_{t-1}^m \ll r_{t-1}^m \) and define the kernel
\[
\nu_t(dz_{1:t} \mid \xi_{1:t}) \triangleq \frac{d q_{t-1}^m(\cdot \mid \xi_{1:t-1}, z_{1:t-1})}{d r_{t-1}^m(\cdot \mid \xi_{1:t-1})} (\xi_t) \\
\times \phi_{t-1}^c(dz_{1:t-1} \mid \xi_{1:t-1}) q_{t-1}^c(dz_t \mid \xi_{1:t}, z_{1:t-1}).
\] (17)
It can now be verified that the choice
\[
a_t(\xi_{1:t}) = \int \nu_t(dz_{1:t} \mid \xi_{1:t}),
\] (18)
\[
\pi_t^c(dz_{1:t} \mid \xi_{1:t}) = \frac{\nu_t(dz_{1:t} \mid \xi_{1:t})}{a_t(\xi_{1:t})},
\] (19)
satisfies assumption (A1), given that \( \phi_{t-1}^c(\cdot \mid \xi_{1:t}) \ll \pi_{t-1}^c(\cdot \mid \xi_{1:t}) \).

Hence, the function \( a_t \) takes the role of a normalisation of the kernel \( \nu_t \) to obtain a transition kernel \( \pi_t^c \). One interesting fact is that, from (14), we cannot guarantee that \( \Delta_t(f) \) is nonnegative for arbitrary functions \( a_t \). At first this might seem counterintuitive, since it would mean that the variance is higher for the RBPF than for the PF. The explanation lies in that assumption (A1), relating the proposal kernels in the two filters, is fairly weak. In other words, we have not assumed that the proposal kernels are “equally good”. For instance, say that the optimal proposal kernel is used in the PF, whereas the RBPF uses a poor kernel. It is then no longer clear that the RBPF will outperform the PF. In the next section we shall see that if both filters use their respective bootstrap proposal kernel, a case when the term “equally good” makes sense, then \( \Delta_t(f) \) will indeed be nonnegative. However, for other proposal kernels, it is not clear that there is an analogue between the PF and the RBPF in the same sense.

### 5.1 Example: Bootstrap kernels

Let \( Q(dx_t \mid x_{t-1}) \) be the Markov transition kernel of the process \( X \). In the bootstrap PF we choose the proposal kernel according to (16a) with
\[
r_{t-1}(dx_t \mid x_{1:t-1}) = Q(dx_t \mid x_{t-1}),
\] (20)
where, for \( A \in \mathcal{X} \),
\[
Q(A \mid X_{t-1}) = \mathbb{P}(X_t \in A \mid X_{t-1}) = \mathbb{P}(X_t \in A \mid X_{1:t-1}, Y_{1:t-1}).
\] (21)
The second equality follows from the Markov property of the process. In the RBPF, the analogue of the bootstrap proposal kernel is to use (16b) with
\[
r_{t-1}^m(A \mid \Xi_{1:t-1}) = \mathbb{P}(\Xi_t \in A \mid \Xi_{1:t-1}, Y_{1:t-1}),
\] (22)
for \( A \in \mathcal{X}_{\xi} \).

It can be verified that these choices fulfill assumption (A1) with \( a_t \equiv 1 \) and
\[
\pi_t^c(A \mid \Xi_{1:t}) = \mathbb{P}(Z_{1:t} \in A \mid \Xi_{1:t}, Y_{1:t-1}),
\] (23)
for \( A \in \mathcal{X}_{\xi}^t \). Hence, \( \pi_t^c \) is indeed the predictive distribution of \( Z_{1:t} \) conditioned on \( \Xi_{1:t} \) and based on the measurements up to time \( t - 1 \). In this case we can also write
\( \pi_t(dx_{1:t}) = \pi^m_t(d\xi_{1:t})\pi^c_t(dz_{1:t} | \xi_{1:t}) \), which highlights the connection between the predictive distributions in the two filters. In this case, since \( a_t = 1 \), the variance difference (14) can be simplified to

\[
\Delta_t(f) = \Delta_{t-1}(R_{t-1}(\cdot, W_t^{-1}\bar{f})) + \phi_{t-1}^m \left[ R_{t-1}^m \left( \cdot, \text{Var}_{\pi^c_t} (W_t^{-1}\bar{f}) \right) \right].
\]

(24)

Hence, \( \Delta_t(f) \) can be written as a sum (though, we have expressed it in a recursive form here) in which each term is an expectation of a conditional variance. It is thus ensured to be nonnegative.

### 6 Discussion

In Theorem 3 we gave an explicit expression for the difference in asymptotic variance between the PF and the RBPF. This expression can be used as a guideline for when it is beneficial to apply Rao-Blackwellisation, and when it is not. The variance expressions given in this paper are asymptotic. Consequently, they do not apply exactly to the variances of the estimators (6) and (7), for a finite number of particles. Still, a reasonable approximation of the accuracy of the estimator (6) is

\[
\text{Var} \left( \hat{f}^N_{PF} \right) \approx \frac{\sigma^2_t(f)}{N},
\]

(25)

and similarly for (7)

\[
\text{Var} \left( \hat{f}^N_{RBPF} \right) \approx \frac{\tau^2_t(\phi^c_t(f))}{N}.
\]

(26)

Now, assume that the computational effort required by the RBPF, using \( M \) particles, equals that required by the PF, using \( N \) particles (thus, \( M < N \) since, in general, the RBPF is more computationally demanding than the PF per particle). We then have

\[
\frac{\text{Var} \left( \hat{f}^N_{PF} \right)}{\text{Var} \left( \hat{f}^M_{RBPF} \right)} \approx \frac{M}{N} \left( 1 + \frac{\Delta_t(f)}{\tau^2_t(\phi^c_t(f))} \right).
\]

(27)

Whether or not this quantity is greater than one tells us if it is beneficial to use Rao-Blackwellisation. The crucial point is then to compute the ratio \( \Delta_t(f)/\tau^2_t(\phi^c_t(f)) \), which in itself is a challenging problem. One option is to apply an RBPF to estimate this ratio, but to sort out the details of how this can be done is a topic for future work.

As a final remark, for the special case discussed in Section 5.1, the variance difference (24) resembles the last term in (8). They are both composed of an expectation of a conditional variance. One important difference though, is that the dependence on the weight function \( W_{t-1} \) is visible in (24). As an example, if the test function is restricted to \( f \in L^1(\mathcal{X}_t, \phi^m_t) \) the gain in variance indicated by (8) would be zero (since \( \text{Var}(f(\Xi_{1:t}) | \Xi_{1:t}) \equiv 0 \)), but this is not the case for the actual gain (24).
7 Conclusions

We have analysed the Rao-Blackwellised particle filter in a fairly general setting, and provide an explicit expression for the reduction of asymptotic variance obtained from Rao-Blackwellisation. This expression is expected to be of practical use, since it can serve as an indicator for when it is beneficial to apply Rao-Blackwellisation, and when it is not. We are currently investigating efficient methods, based on the analytical expression, for estimating the obtained variance reduction.

Appendix

A Proofs

Proof of Theorem 1: In Theorem 10 in (Douc and Moulines, 2008), take

$$L_{t-1}(\tilde{x}_{1:t-1}, dx_{1:t}) = W_{t-1}(x_{1:t})R_{t-1}(\tilde{x}_{1:t-1}, dx_{1:t}),$$

(28)

which satisfies the conditions of the hypothesis. Furthermore, $\phi_{t-1}L_{t-1}(X^t) = 1$. Now, the results follow for the choice $\kappa = 0$.

In Theorem 10 in (Douc and Moulines, 2008), the asymptotic normality of a particle system obtained after resampling is considered. Compared to Theorem 1 of this paper, they thus obtain an additional term in the expression for the asymptotic variance.

Proof of Theorem 2: As the previous proof, with

$$L_{t-1}^m(\tilde{\xi}_{1:t-1}, d\xi_{1:t}) = W_{t-1}^m(\xi_{1:t})R_{t-1}^m(\tilde{\xi}_{1:t-1}, d\xi_{1:t}).$$

(29)

Proof of Theorem 3: Let assumption (A1) be satisfied. Consider

$$\phi_t(dx_{1:t}) = \int \phi_{t-1}(d\tilde{x}_{1:t-1})W_{t-1}(x_{1:t})R_{t-1}(\tilde{x}_{1:t-1}, dx_{1:t})$$

$$= a_t(\xi_{1:t})W_{t-1}(x_{1:t})\pi_t^m(d\xi_{1:t})\pi_t^c(dz_{1:t} | \xi_{1:t}),$$

(30)

where we have used (3) and (4) for the first equality, and (5) and assumption (A1) for the second equality (recall that $\pi_t^m$ is defined analogously to (3)).

However, we may also write

$$\phi_t = \frac{d\phi_t^m}{d\pi_t^m} \frac{d\phi_t^c}{d\pi_t^c} \pi_t^m \pi_t^c.$$  

(31)

Hence, we have two candidates for the Radon-Nikodym derivative of $\phi_t$ with respect to $\pi_t^m \pi_t^c$ which, $\pi_t^m \pi_t^c$-almost surely, implies

$$a_t(\xi_{1:t})W_{t-1}(x_{1:t}) = W_{t-1}^m(\xi_{1:t}) \frac{d\phi_t^c}{d\pi_t^c}(\cdot | \xi_{1:t}) (z_{1:t}).$$

(32)
Consider arbitrary \( \varphi \in \hat{\mathbb{A}}_t \). Using (5) and assumption (A1) we may write
\[
\phi_{t-1} \left[ R_{t-1} \left( \cdot, \varphi \right) \right] = \phi^m_{t-1} \left[ R^m_{t-1} \left( \cdot, a_t \pi^c_t(\varphi) \right) \right].
\] (33)
Comparing (33) and (10), we see that we can let \( \varphi \) take the role of \( (W_{t-1} \bar{f})^2 \). Hence, consider
\[
\pi^c_t ((W_{t-1} \bar{f})^2) = (\pi^c_t (W_{t-1} \bar{f}))^2 + \text{Var}_{\pi^c_t} (W_{t-1} \bar{f}),
\] (34)
where, using (32) we have \( \pi^m_t \)-almost surely,
\[
\pi^c_t (W_{t-1} \bar{f}) = \int \frac{W^m_{t-1}(\xi_{1:t})}{a_t(\xi_{1:t})} \frac{d\phi^c_t(\cdot)}{d\pi^c_t(\cdot)}(z_{1:t}) \bar{f}(\xi_{1:t}) \pi^c_t(dz_{1:t} | \xi_{1:t})
\]
\[
= \frac{W^m_{t-1}(\xi_{1:t})}{a_t(\xi_{1:t})} \phi^c_t(\bar{f}) = \frac{W^m_{t-1}(\xi_{1:t})}{a_t(\xi_{1:t})} \bar{\psi}(\xi_{1:t}).
\] (35)
Combining (34) and (35) we get, \( \pi^m_t \)-almost surely,
\[
a_t(\xi_{1:t}) \pi^c_t ((W_{t-1} \bar{f})^2) = \left( \frac{W^m_{t-1}(\xi_{1:t}) \bar{\psi}(\xi_{1:t})}{a_t(\xi_{1:t})} \right)^2 + a_t(\xi_{1:t}) \text{Var}_{\pi^c_t} (W_{t-1} \bar{f}).
\] (36)
Let \( L_{t-1} \) and \( L^m_{t-1} \) be defined as in (28) and (29), respectively. Then, \( R_{t-1} \left( \cdot, W_{t-1} \bar{f} \right) = L_{t-1} \left( \cdot, \bar{f} \right) \) and \( R^m_{t-1} \left( \cdot, W^m_{t-1} \bar{\psi} \right) = L^m_{t-1} \left( \cdot, \bar{\psi} \right) \).

Using (12), (10), (11) and the above results, the difference in asymptotic variance can be expressed as
\[
\Delta_t(f) = \sigma^2_{t-1} \left( L_{t-1} \left( \cdot, \bar{f} \right) \right) - \tau^2_{t-1} \left( L^m_{t-1} \left( \cdot, \bar{\psi} \right) \right)
\]
\[
+ \phi_{t-1} \left[ R_{t-1} \left( \cdot, (W_{t-1} \bar{f})^2 \right) \right] - \phi^m_{t-1} \left[ R^m_{t-1} \left( \cdot, (W^m_{t-1} \bar{\psi})^2 \right) \right]
\]
\[
= \sigma^2_{t-1} \left( L_{t-1} \left( \cdot, \bar{f} \right) \right) - \tau^2_{t-1} \left( L^m_{t-1} \left( \cdot, \bar{\psi} \right) \right)
\]
\[
+ \phi^m_{t-1} \left[ R^m_{t-1} \left( \cdot, \left( \frac{1}{a_t} - 1 \right) (W^m_{t-1} \bar{\psi})^2 + a_t \text{Var}_{\pi^c_t} (W_{t-1} \bar{f}) \right) \right]
\] (37)
(recall that \( \pi^m_t = \phi^m_{t-1} R^m_{t-1} \), which ensures that we, due to the expectation w.r.t. \( \phi^m_{t-1} R^m_{t-1} \) in (37), can make use of the equality in (36)).

Finally, by straightforward, but somewhat tedious manipulations
\[
\phi^c_{t-1} (L_{t-1} \left( \cdot, \bar{f} \right)) = L^m_{t-1} \left( \cdot, \bar{\psi} \right), \quad \pi^m_t \text{-almost surely.}
\] (38)
Hence,
\[
\sigma^2_{t-1} \left( L_{t-1} \left( \cdot, \bar{f} \right) \right) - \tau^2_{t-1} \left( L^m_{t-1} \left( \cdot, \bar{\psi} \right) \right) = \Delta_{t-1}(L_{t-1} \left( \cdot, \bar{f} \right)),
\] (39)
which completes the proof.
Bibliography


Paper G  An explicit variance reduction expression for the Rao-Blackwellised particle filter


