Some contributions to particle Markov chain Monte Carlo algorithms

A thesis presented for the degree of
Doctor of Philosophy of Imperial College London
and the
Diploma of the Imperial College (DIC)

by

Adam Ford Persing

Department of Mathematics
South Kensington Campus
Imperial College London
London, SW7 2AZ

FEBRUARY 27, 2014
I certify that this thesis and research are the product of my own work, and any ideas or quotations from the work of others are acknowledged according to standard referencing practices.

Adam Ford Persing
Copyright

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.
To my friends and family
Abstract

Hidden Markov models (HMMs) [13] and discrete time stopped Markov processes [23, Section 2.2.3] are used to model phenomena in a wide range of fields. However, as practitioners develop more intricate models, analytical Bayesian inference becomes very difficult. In light of this issue, this work focuses on sampling from the posteriors of HMMs and stopped Markov processes using sequential Monte Carlo (SMC) ([29], [28], [43]) and, more importantly, particle Markov chain Monte Carlo (PMCMC) [2].

The thesis consists of three major contributions, which enhance the performance of PMCMC. The first work focuses on HMMs, and it begins by introducing a new SMC smoothing ([10], [36]) estimate of the HMM’s normalising constant; we prove the estimate’s unbiasedness and a central limit theorem. We use this estimate to develop new PMCMC algorithms that, under certain algorithmic settings, require less computational time than the algorithms of [2]. Our new estimate also leads to the discovery of an optimal setting for the smoothers of [10] and [36]. As this setting is not available for the general class of HMMs, we develop three algorithms for approximating it.

The second major work builds from [51] and [91] to develop new SMC and PMCMC algorithms that draw from HMMs whose observations have intractable density functions. While these types of algorithms have appeared before (see [51], [52] and [67]), this work uses twisted proposals as in [91] to reduce the variance of SMC estimates of the normalising constant to improve the convergence of PMCMC in some scenarios.

Finally, the third project is concerned with inferring the unknown parameters of stopped Markov processes that are only observed upon reaching their terminal sets. Bayesian inference has not been attempted on this class of problems before. The parameters are inferred through two new adaptive and non-adaptive PMCMC algorithms.
Acknowledgements

First and foremost, I would like to thank my principal supervisor, mentor, and friend, Ajay Jasra. I would also like to thank Leonardo Bottolo and Dan Crisan for providing supervision throughout the course of my degree.

I express my sincere gratitude to Arnaud Doucet for significant contributions to the work appearing in Chapter 3 and Nikolas Kantas for collaborating on the work appearing in Chapter 5.

The Department of Mathematics at Imperial College London was kind enough to fund my tuition fees beyond the first year of my degree through a Roth Studentship.

Adam Ford Persing
# Table of contents

## Abstract

5

## List of Publications

13

## 1 Introduction and Motivation

1.1 Bayesian parameter inference ........................................... 15
1.2 Some applications of the posterior density ............................... 16
1.3 Difficulty in computing the posterior density ............................ 17
1.4 Numerical integration .................................................. 18
1.5 Monte Carlo and Markov chain Monte Carlo methods .................... 19
1.6 Contributions of this thesis ........................................... 20

## 2 Literature Review

2.1 Notation ................................................................. 25
2.2 Hidden Markov models ................................................. 27
   2.2.1 Filtering ......................................................... 29
   2.2.2 Smoothing ...................................................... 31
   2.2.3 Towards the approximation of hidden Markov models .............. 32
2.3 Stopped Markov processes ............................................. 33
   2.3.1 The Wright-Fisher model and Kingman’s coalescent ............. 36
2.4 Basics of Monte Carlo ................................................. 42
2.5 Importance Sampling .................................................. 43
   2.5.1 Random weight importance sampling ............................. 44
2.6 Sequential importance sampling ....................................... 45
2.7 Basics of sequential Monte Carlo .................................... 47
   2.7.1 A note on resampling .......................................... 48
   2.7.2 Block-sampling within sequential Monte Carlo .................. 50
   2.7.3 High dimensionality .......................................... 50
2.8 Sequential Monte Carlo filtering techniques for hidden Markov models ........................................... 52
   2.8.1 Comparison to Kalman filters .................................. 53
2.9 Sequential Monte Carlo smoothing techniques for hidden Markov models ........................................... 55
   2.9.1 Approximation of FFBS ...................................... 55
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.9.2 Generalised two-filter smoothing</td>
<td>58</td>
</tr>
<tr>
<td>2.10 HMM inference using approximate Bayesian computation and SMC</td>
<td>61</td>
</tr>
<tr>
<td>2.11 Sequential Monte Carlo techniques for stopped Markov processes</td>
<td>65</td>
</tr>
<tr>
<td>2.11.1 Backward importance sampling for stopped Markov processes</td>
<td>66</td>
</tr>
<tr>
<td>2.12 Twisted particle filters</td>
<td>69</td>
</tr>
<tr>
<td>2.13 Sequential Monte Carlo samplers</td>
<td>71</td>
</tr>
<tr>
<td>2.14 Basics of Markov chain Monte Carlo</td>
<td>73</td>
</tr>
<tr>
<td>2.14.1 Metropolis-Hastings algorithm</td>
<td>73</td>
</tr>
<tr>
<td>2.14.2 Gibbs sampler</td>
<td>75</td>
</tr>
<tr>
<td>2.14.3 Convergence of Markov chain Monte Carlo</td>
<td>76</td>
</tr>
<tr>
<td>2.15 Block updating for Markov chain Monte Carlo</td>
<td>78</td>
</tr>
<tr>
<td>2.16 Parallel tempering for Markov chain Monte Carlo</td>
<td>78</td>
</tr>
<tr>
<td>2.17 Exact approximation of Markov chain Monte Carlo</td>
<td>79</td>
</tr>
<tr>
<td>2.18 Particle Markov chain Monte Carlo</td>
<td>81</td>
</tr>
<tr>
<td>2.18.1 Extensions of particle Markov chain Monte Carlo</td>
<td>84</td>
</tr>
<tr>
<td>2.19 Markov chain Monte Carlo moves within sequential Monte Carlo</td>
<td>87</td>
</tr>
<tr>
<td>2.20 Parallel computing</td>
<td>89</td>
</tr>
<tr>
<td>2.21 Final remarks</td>
<td>90</td>
</tr>
<tr>
<td>3 Generalised Two-filter Smoothing and Particle Markov chain Monte Carlo</td>
<td>91</td>
</tr>
<tr>
<td>3.1 Unbiased estimates of the marginal likelihood for HMMs</td>
<td>93</td>
</tr>
<tr>
<td>3.1.1 Estimate of complexity $O(N^2)$</td>
<td>93</td>
</tr>
<tr>
<td>3.1.2 Estimate of complexity $O(N)$</td>
<td>95</td>
</tr>
<tr>
<td>3.2 Numerical comparison of $O(N)$ estimates</td>
<td>98</td>
</tr>
<tr>
<td>3.2.1 Linear Gaussian model</td>
<td>100</td>
</tr>
<tr>
<td>3.2.2 Stochastic volatility model</td>
<td>106</td>
</tr>
<tr>
<td>3.2.3 Primary findings</td>
<td>108</td>
</tr>
<tr>
<td>3.3 Optimal pseudo-priors</td>
<td>109</td>
</tr>
<tr>
<td>3.4 Application of generalised two-filter smoothing in PMCMC</td>
<td>110</td>
</tr>
<tr>
<td>3.4.1 Samplers of complexity $O(N^2)$</td>
<td>111</td>
</tr>
<tr>
<td>3.4.2 Samplers of complexity $O(N)$</td>
<td>113</td>
</tr>
<tr>
<td>3.5 Implementation of the $O(N)$ PMMH and PG algorithms</td>
<td>117</td>
</tr>
<tr>
<td>3.5.1 Linear Gaussian model</td>
<td>120</td>
</tr>
<tr>
<td>3.5.2 Stochastic volatility model</td>
<td>125</td>
</tr>
<tr>
<td>3.5.3 Primary findings</td>
<td>131</td>
</tr>
<tr>
<td>3.6 New smoothing algorithms</td>
<td>132</td>
</tr>
<tr>
<td>3.6.1 Parametric and marginal smoothers</td>
<td>133</td>
</tr>
<tr>
<td>3.6.2 Random weight smoothing</td>
<td>134</td>
</tr>
<tr>
<td>3.7 Implementation of the new smoothing algorithms</td>
<td>138</td>
</tr>
<tr>
<td>3.7.1 Linear Gaussian model</td>
<td>141</td>
</tr>
<tr>
<td>3.7.2 Stochastic volatility model</td>
<td>147</td>
</tr>
<tr>
<td>3.8 Discussion</td>
<td>149</td>
</tr>
</tbody>
</table>
## 4 Twisting the Alive Particle Filter

4.1 Notation and definitions ........................................... 160
4.2 Alive twisted sequential Monte Carlo .......................... 162
4.3 Optimal change in measure ........................................ 165
4.4 Implementation of alive twisted SMC on a linear Gaussian model .......................... 168
4.5 Alive twisted particle marginal Metropolis-Hastings ............. 173
4.6 Implementation of alive twisted PMMH on a stochastic volatility model ............ 176
4.7 Discussion .......................................................... 184

## A Proof of the main result from Section 4.3 .................... 186

## B Table of kernel and operator notation ......................... 202

## 5 Inference for Partially Observed Stopped Markov Processes 203

5.1 Multi-level PMMH: fixed $\mathcal{F}$–sets .......................... 204
5.2 Multi-level PMMH: adaptive $\mathcal{F}$–sets ......................... 205
5.3 Implementation of the multi-level PMMH algorithms ............ 207
5.3.1 Coalescent with mutation ...................................... 209
5.3.2 Coalescent with mutation and migration ..................... 214
5.4 Discussion .......................................................... 219

## 6 Summary and Future Work 221

6.1 Review of Chapter 3 ................................................. 221
6.2 Review of Chapter 4 ................................................. 222
6.3 Review of Chapter 5 ................................................. 223

## References 225
## List of Figures

<table>
<thead>
<tr>
<th>Section</th>
<th>Figure Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1</td>
<td>Example of the coalescent with mutation</td>
<td>39</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Comparing estimates of $Z_\theta$: linear Gaussian HMM, small $N$, fixed $t$</td>
<td>103</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Comparing estimates of $Z_\theta$: linear Gaussian HMM, large $N$, fixed $t$</td>
<td>104</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Comparing estimates of $Z_\theta$: linear Gaussian HMM, large $N$, variable $t$</td>
<td>105</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Comparing estimates of $Z_\theta$: stochastic volatility HMM, fixed $t$</td>
<td>108</td>
</tr>
<tr>
<td>3.5.1</td>
<td>PG applied to a linear Gaussian HMM: Trial 1</td>
<td>122</td>
</tr>
<tr>
<td>3.5.2</td>
<td>PG applied to a linear Gaussian HMM: Trial 2</td>
<td>123</td>
</tr>
<tr>
<td>3.5.3</td>
<td>PG applied to a linear Gaussian HMM: Trial 3</td>
<td>124</td>
</tr>
<tr>
<td>3.5.4</td>
<td>PMMH applied to a stochastic volatility HMM: Trial 1, convergence</td>
<td>126</td>
</tr>
<tr>
<td>3.5.5</td>
<td>PMMH applied to a stochastic volatility HMM: Trial 1, trace plot</td>
<td>127</td>
</tr>
<tr>
<td>3.5.6</td>
<td>PMMH applied to a stochastic volatility HMM: Trial 2, convergence</td>
<td>128</td>
</tr>
<tr>
<td>3.5.7</td>
<td>PMMH applied to a stochastic volatility HMM: Trial 2, trace plot</td>
<td>129</td>
</tr>
<tr>
<td>3.5.8</td>
<td>PMMH applied to a stochastic volatility HMM: Trial 3, convergence</td>
<td>130</td>
</tr>
<tr>
<td>3.5.9</td>
<td>PMMH applied to a stochastic volatility HMM: Trial 3, trace plots</td>
<td>131</td>
</tr>
<tr>
<td>3.7.1</td>
<td>ESS of SMC smoothers: linear Gaussian HMM, small $N$</td>
<td>143</td>
</tr>
<tr>
<td>3.7.2</td>
<td>Variance of $\hat{Z}_{\theta,1:T}$ of SMC smoothers: linear Gaussian HMM, small $N$</td>
<td>144</td>
</tr>
<tr>
<td>3.7.3</td>
<td>ESS of SMC smoothers: linear Gaussian HMM, large $N$</td>
<td>145</td>
</tr>
<tr>
<td>3.7.4</td>
<td>Variance of $\hat{Z}_{\theta,1:T}$ of SMC smoothers: linear Gaussian HMM, large $N$</td>
<td>146</td>
</tr>
<tr>
<td>3.7.5</td>
<td>ESS of SMC smoothers: stochastic volatility HMM</td>
<td>148</td>
</tr>
<tr>
<td>3.7.6</td>
<td>Variance of $\hat{Z}_{\theta,1:T}$ of SMC smoothers: stochastic volatility HMM</td>
<td>149</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Comparing alive SMC to alive twisted SMC with few particles</td>
<td>171</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Comparing alive SMC to alive twisted SMC with many particles</td>
<td>172</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Comparing alive SMC to alive twisted SMC with small $\nu$ and $\tau$</td>
<td>173</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Daily closing index value of S&amp;P 500 and the daily logarithmic returns</td>
<td>177</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Comparing alive PMMH to alive twisted PMMH, $\alpha = 1.75$: ACF plots</td>
<td>178</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Comparing alive PMMH to alive twisted PMMH, $\alpha = 1.75$: trace plots</td>
<td>179</td>
</tr>
<tr>
<td>4.6.4</td>
<td>Comparing alive PMMH to alive twisted PMMH, $\alpha = 1.95$: ACF plots</td>
<td>181</td>
</tr>
<tr>
<td>4.6.5</td>
<td>Comparing alive PMMH to alive twisted PMMH, $\alpha = 1.95$: trace plots</td>
<td>182</td>
</tr>
<tr>
<td>5.3.1</td>
<td>ML-PMMH applied to the coalescent: estimated pdfs</td>
<td>212</td>
</tr>
<tr>
<td>5.3.2</td>
<td>ML-PMMH applied to the coalescent: ACF plots</td>
<td>213</td>
</tr>
</tbody>
</table>
5.3.3 ML-PMMH applied to the coalescent: trace plots . . . . . . . . . . . . . 214
5.3.4 Dataset for the coalescent with migration . . . . . . . . . . . . . . . . 215
5.3.5 ML-PMMH applied to the coalescent with migration: estimated pdfs . . 217
5.3.6 ML-PMMH applied to the coalescent with migration: ACF plots . . . . 218
5.3.7 ML-PMMH applied to the coalescent with migration: trace plots . . . . 219
List of Tables

2.1 Distributions used throughout the thesis . . . . . . . . . . . . . . . . . . . 27
2.2 Acronyms used throughout the thesis . . . . . . . . . . . . . . . . . . . . 28
B1 Kernel and operator notation used throughout Chapter 4 . . . . . . . . . . . 202
List of publications

Most of the author’s research can be found after the second chapter of this thesis. Some of the research presented here can also be found in the following:


“We introduce an estimate for the likelihood of hidden Markov models (HMMs) using sequential Monte Carlo (SMC) approximations of the generalized two-filter smoothing decomposition [10]. This estimate is unbiased and a central limit theorem (CLT) is established. The new estimate is also investigated from a numerical perspective.”

An electronic copy may be found at:
http://dx.doi.org/10.1016/j.spl.2013.02.005


“We consider Bayesian parameter inference associated to partially-observed stochastic processes that start from a set $B_0$ and are stopped or killed at the first hitting time of a known set $A$. Such processes occur naturally within the context of a wide variety of applications. The associated posterior distributions are highly complex and posterior parameter inference requires the use of advanced Markov chain Monte Carlo (MCMC) techniques. Our approach uses a recently introduced simulation methodology, particle Markov chain Monte Carlo (PMCMC) [2], where sequential Monte Carlo (SMC) ([28], [65]) approximations are embedded within MCMC. However,
when the parameter of interest is fixed, standard SMC algorithms are not always appropriate for many stopped processes. In [17] and [23], the authors introduce SMC approximations of multi-level Feynman-Kac formulae, which can lead to more efficient algorithms. This is achieved by devising a sequence of sets from $B_0$ to $A$ and then performing the resampling step only when the samples of the process reach intermediate sets in the sequence. The choice of the intermediate sets is critical to the performance of such a scheme. In this paper, we demonstrate that multi-level SMC algorithms can be used as a proposal in PMCMC. In addition, we introduce a flexible strategy that adapts the sets for different parameter proposals. Our methodology is illustrated on the coalescent model with migration.”

An electronic copy may be found at:
http://www.springerlink.com/content/h4706675h265221w/

• Adam Persing and Ajay Jasra. Twisting the alive particle filter. (submitted), 2014.

“This work focuses on sampling from hidden Markov models [13] whose observations have intractable density functions. We develop a new sequential Monte Carlo ([29], [28], [43]) algorithm and a new particle marginal Metropolis-Hastings [2] algorithm for these purposes. We build from [51] and [91] to construct the sequential Monte Carlo (SMC) algorithm (which we call the alive twisted particle filter). Like the alive particle filter of [51], our new SMC algorithm adopts an approximate Bayesian computation [87] estimate of the HMM. Our alive twisted particle filter also uses a twisted proposal as in [91] to obtain a low-variance estimate of the HMM normalising constant. We demonstrate via numerical examples that, in some scenarios, this estimate has a much lower variance than that of the estimate obtained via the alive particle filter. The low variance of this normalising constant estimate encourages the implementation of our SMC algorithm within a particle marginal Metropolis-Hastings (PMMH) scheme, and we call the resulting methodology ‘alive twisted PMMH’. We numerically demonstrate on a stochastic volatility model how our alive twisted PMMH can converge faster than the standard alive PMMH of [51].”

An electronic copy may be found at:
http://arxiv.org/abs/1308.4462
Chapter 1

Introduction and Motivation

The main aim of this thesis is to make contributions to a recently discovered class of numerical methods which are used in computational Bayesian statistics, called particle Markov chain Monte Carlo (PMCMC) algorithms [2]. Before delving into the specifics of PMCMC (a review of which is given in Chapter 2), we begin here with a brief review of Bayesian inference problems and their usefulness (see also [46, Chapter 11]). We observe that analytical solutions to Bayesian inference problems are not always available in real world applications, thereby motivating the need for numerical methods.

1.1 Bayesian parameter inference

The following discussion is limited to continuous random variables, with the extension to discrete random variables being obvious.

Suppose that, in advance of observing realisations of a real-valued (possibly multi-dimensional) random variable \( Y \in \mathbb{R}^{d_Y} \), one establishes a model (with input parameters collectively denoted by \( \theta \)) that assigns measures of uncertainty to the values that \( Y \) might take. In Bayesian statistics, those measures of uncertainty are probabilities, and they are defined on scales from zero to one. Thus, under our model, the probability

\[
P( Y \in E \mid \theta ) = \int_{E} \pi( y \mid \theta ) \, dy,
\]
with density $\pi$, is a measure of the belief that $Y$ will take a value in the set $E \subseteq \mathbb{R}^d_Y$, given some parameter $\theta$. Just as the exact value of $Y$ is unknown, the exact value of $\theta$ may also be unknown. In this case, $\theta$ becomes another random variable $\theta \in \mathbb{R}^d_\theta$, and the belief that this parameter takes a certain value in $F \subseteq \mathbb{R}^d_\theta$ is given by

$$\mathbb{P}(\theta \in F) = \int_F \pi(\theta) \, d\theta.$$ 

Note that this measure of belief may also condition on other random variables (called hyper-parameters). For example, we may write $\pi(\theta) = \pi(\theta | \epsilon, \nu)$ where $\epsilon \sim \xi(\cdot)$ and $\nu \sim \phi(\cdot)$.

Let us assume that $\pi(\theta)$ only conditions upon fixed hyperparameters. As $\pi(\theta)$ is ultimately used to give a measure of the belief of the value of $\theta$ before a value of $Y$ is observed, we say that $\pi(\theta)$ is a prior density. After observing a value $Y = y$ (hereafter, $y$ is called the data), updating the belief of the value of $\theta$ amounts to constructing $\pi(\theta | Y = y)$. This density is named the posterior density, and its exact expression is given by Bayes’ theorem:

$$\pi(\theta | Y = y) = \frac{\pi(\theta) \, \pi(Y = y | \theta)}{\pi(Y = y)} \propto \pi(\theta) \, \pi(Y = y | \theta). \quad (1.1.1)$$

In this context, $\pi(Y = y | \theta)$ is referred to as a likelihood and the expression $\pi(Y = y)$ is a normalising constant.

### 1.2 Some applications of the posterior density

The expression (1.1.1) is useful when a practitioner is trying to infer the parameter $\theta$ for a particular model after having observed a data set $y$. Consider the real-valued function $h(\theta)$ and the integral

$$\int_{-\infty}^{\infty} h(\theta) \pi(\theta | Y = y) \, d\theta. \quad (1.2.1)$$

When $h(\theta) = \theta$ for all $\theta \in \mathbb{R}^d_\theta$, then (1.2.1) is the conditional expectation of $\theta$ with respect to the posterior, and it is denoted as $\mathbb{E}_\pi(\theta | Y = y)$. This expectation is also known as the first moment, and the $i^{th}$ moment can be calculated when $h(\theta) = \theta^i$ for all $\theta \in \mathbb{R}^d_\theta$. Alternatively, by setting $h(\theta) = [\theta - \mathbb{E}_\pi(\theta | Y = y)]^i$ for all $\theta \in \mathbb{R}^d_\theta$, (1.2.1) becomes the $i^{th}$ central moment with respect to the posterior. With this definition of $h(\theta)$, one obtains
The conditional variance of $\theta$ when $i = 2$.

The expression (1.2.1) arises when trying to determine an appropriate estimate of $\theta$ (denoted $\hat{\theta}$). Consider a loss function $L(\theta, \hat{\theta})$, such as the squared error function. The Bayes risk of $\hat{\theta}$ is defined as the expected value of the loss function with respect to the density $\pi(\theta)$, or $\mathbb{E}_\pi[L(\theta, \hat{\theta})]$. Any estimator that minimises the Bayes risk is known as a Bayes estimator. If one’s loss function is the squared error function, then the Bayes estimator can be calculated as $\mathbb{E}_\pi(\theta \mid Y = y)$.

1.3 Difficulty in computing the posterior density

The density (1.1.1) is useful, for as the last section shows, it enables us to update our model parameters as we observe new datasets. However, the posterior is only analytically tractable in a limited number of specific scenarios. One such scenario is when a conjugate prior $\pi(\theta)$ exists for $\pi(y \mid \theta)$. In this case, one can find a closed form expression for (1.1.1), and in fact, $\pi(\theta \mid Y = y)$ will be in the same family of densities as the prior. Unfortunately, the conjugate families only apply in some applications of interest (i.e., they are mainly available for exponential family problems), and one can find many more examples where there is no known analytically tractable expression for (1.1.1). Chapter 2 gives a more detailed review of some of those specific models for which the posterior density can be difficult to compute. However, to gain some appreciation for the problem at hand now, we review a real world model here.

The random effects model is a standard linear statistical model that is used for study in a wide array of fields, including epidemiology ([1], [70]) and fMRI research [82]. The intuition behind the model is easy to grasp, but in practice, the model can become difficult to work with analytically. In a random effects model, one might have a multi-dimensional $Y$, whose real-valued elements $\{Y_{ij}\}$ are independent of one another and have distributions which depend on a hierarchy of different random factors (i.e., $Y_{ij} = \mu + A_i + B_{ij}$). The parameter $\mu$ is an expected value common to all elements of $Y$, $A_w$ follows some distribution that is only common to $Y_{ij}$’s where $i = w$, and $B_{wz}$ follows some distribution that is only common to $Y_{ij}$’s where $i = w$ and $j = z$. In more complex models, $Y$ could be of an even higher dimension (e.g., $Y_{ijkl} = \mu + A_i + B_{ij} + C_{ijk} + D_{ijkl}$).
For $Y_{ij} = \mu + A_i + B_{ij}$, suppose $i \in \{1, \ldots, 2\}$ and $j \in \{1, \ldots, 3\}$. We additionally set $\mu = 0$, with $A_i$ being normally distributed with mean $\mu_{A_i}$ and variance $\sigma_{A_i}^2$ and $B_{ij}$ being normally distributed with mean $\mu_{B_{ij}}$ and variance $\sigma_{B_{ij}}^2$. If the elements of $\theta = (\mu_{A_{1:2}}, \sigma_{A_{1:2}}^2, \mu_{B_{1:2,1:3}}, \sigma_{B_{1:2,1:3}}^2)$ independently follow exponential priors with common parameter $\lambda$, then the density (1.1.1) is

$$
\pi(\theta \mid Y_{1:2,1:3} = y_{1:2,1:3}) \propto \exp \left[ -\lambda \sum_{i=1}^{2} \left( \mu_{A_i} + \sigma_{A_i}^2 + \sum_{j=1}^{3} \mu_{B_{ij}} + \sum_{j=1}^{3} \sigma_{B_{ij}}^2 \right) \right] \times 
\exp \left[ \sum_{i=1}^{2} \sum_{j=1}^{3} -\frac{(y_{ij} - \mu_{A_i} - \mu_{B_{ij}})^2}{2(\sigma_{A_i}^2 + \sigma_{B_{ij}}^2)} \right]
$$

Direct analytical computations with respect to $\pi(\theta \mid Y_{1:2,1:3} = y_{1:2,1:3})$ are not possible.

### 1.4 Numerical integration

When a probability density is analytically intractable, computations of the form (1.2.1) must be estimated. Numerical integration offers one class of solutions for such approximations. A complete review of numerical integration is covered in [79], but here, we give a brief introduction to a subset of those techniques (i.e., Newton-Cotes approximations) to give the reader a basic idea of the methods.

To begin, consider a real-valued function $f(\theta) : \mathbb{R} \to \mathbb{R}$ with $\theta \in \mathbb{R}$, which is to be integrated in a one-dimensional, finite region $E \subset \mathbb{R}$. In the Newton-Cotes approximation methods (or, quadrature methods), the definite integral is divided into segments of width $w$, and $f$ is approximated by a polynomial at each segment. The areas under the polynomials are taken to be approximations of the integral at each segment, and summing those areas provides an estimate of the entire integral of interest. There are many forms of the Newton-Cotes methods that use polynomials of different degrees. One such basic form uses straight lines, and each segment between points $\theta_1 \in E$ and $\theta_2 \in E$ is approximated as

$$
\int_{\theta_1}^{\theta_2} f(\theta) d\theta \approx \frac{w}{2} \left[ f(\theta_1) + f(\theta_2) \right],
$$

where $\zeta \in E$ is a point satisfying $\theta_1 \leq \zeta \leq \theta_2$ (this formula is called the trapezoidal rule).
Using quadratic polynomials gives Simpson’s rule:

\[
\int_{\theta_1}^{\theta_2} f(\theta) d\theta \approx \frac{w}{6} \left[ f(\theta_1) + 4f(\zeta) + f(\theta_2) \right],
\]

where \( \theta_1 < \zeta < \theta_2 \), and the three points are equally spaced apart. In general, as the degree of the polynomials increases, they more appropriately fit the function \( f \) and the error of the approximation of the integral at each segment shrinks. Furthermore, as the entire integral is divided into more segments, the approximation becomes more fine.

In the specific case of approximating a finite integral where \( f(\theta) = h(\theta)\pi(\theta | Y = y) \) and the normalising constant of \( \pi \) is unknown, one needs to perform two approximation steps. In the first step,

\[
\int_E h(\theta)\pi(\theta | Y = y) d\theta
\]

is approximated using some quadrature rule. Then, as the normalising constant of \( \pi \) is unknown, the integral

\[
\int_E \pi(\theta | Y = y) d\theta
\]

must also be approximated using a quadrature method. The first estimate is divided by the second estimate to obtain a normalised approximation of the integral of interest. For the reasons mentioned above, we know that this scheme will not be without error.

When the integrals of interest are multi-dimensional, methods such as the trapezoidal rule and Simpson’s rule have to be repeated across those multiple dimensions. The evaluations, therefore, can become very complex. Furthermore, in the case of computing (1.2.1), where the limits of integration are not finite, numerical integration can be prone to many errors because only a finite number of segments can actually be approximated.

### 1.5 Monte Carlo and Markov chain Monte Carlo methods

Monte Carlo (MC) and Markov chain Monte Carlo (MCMC) methods offer a second class of solutions to the approximation problem (see [12] and [81]). In these types of algorithms, samples are drawn from intractable densities known point wise up to a normalising constant, and those samples are used to approximate any analytical computations that oth-
1.6 Contributions of this thesis

We are concerned with sampling from (i.e., “targeting”) complex posterior densities whose state spaces include the random variables \( X_{1:n} \in \mathbb{R}^{d_1} \) with \( n \geq 1 \) where each \( X_n \) corresponds to the state of a Markov chain. Recall that in a Markov chain, \( \mathbb{P} (X_n = x_n \mid x_{1:n-1}) = \mathbb{P} (X_n = x_n \mid x_{n-1}) \). The state spaces will also sometimes contain a static random variable \( \theta \in \mathbb{R}^{d_\theta} \) which is invariant to changes in \( n \). These types of targets arise in many models/processes where Bayesian inference can be difficult, and our work focuses on a specific subset of those models and processes. That subset is properly reviewed in the beginning of the next chapter, but briefly, it consists of:

- Hidden Markov models (see Section 2.2): a hidden Markov model (HMM) with static parameter \( \theta \) essentially consists of discrete observations \( y_{1:n} \) from which the unobservable Markov random variables \( X_{1:n} \) must be inferred through sequences of “filtering densities” of the form \( \pi_\theta (x_n \mid y_{1:n}) \) or “smoothing densities” of the form

MC and MCMC methods can be particularly attractive over numerical integration procedures in the cases where the boundaries of integration are very complicated. MC and MCMC methods can also be easier to implement in multi-dimensional analyses. Their general applicability to a wider range of problems is one of the main reasons that MC and MCMC methods are studied by so many researchers. For large or complex models, numerical integration methods typically only perform well when they are carefully tailored to the problem at hand.

As increasing amounts of data become available to practitioners, real world models are becoming more complex and their dimensionality is increasing; see [21], [48] and [93] for examples. Therefore, in this thesis, we choose to focus on studying the more general class of numerical methods. That is, we study MC and MCMC methods with the ultimate goal of performing Bayesian inference.
\( \pi_\theta(x_{n_1:n_2} \mid y_{1:T}) \) for \( 1 \leq n_1 \leq n_2 < T \). The parameter \( \theta \) is also sometimes unknown, in which case we aim to infer it and the Markov process through the density \( \pi(\theta, x_{1:n} \mid y_{1:n}) \) and its marginals.

- **Discrete time stopped Markov processes** (see Section 2.3): these Markov processes \( \{X_n\}_{n \geq 1} \) require a parameter \( \theta \) and are initialised in some set \( B_0 \). They evolve in discrete time until they reach a target set \( A \) at some random time \( \tau \) and terminate.

  In this thesis, the only observation is \( y = x_\tau \), and inference relies on the density \( \pi(\theta, x_{1:\tau}, \tau \mid y) \). One major application of such processes is the modelling of phenomena in population genetics.

The structure for the thesis is as follows. Chapter 2 opens by introducing the models and Markov processes of interest and formulating their inference problems. We then review the MC and MCMC techniques that can be used to sample from the posteriors of these models/processes. The review takes a special interest in the major advancements leading to sequential Monte Carlo ([29],[28],[43]) and PMCMC. Sequential Monte Carlo (SMC) is a methodology tailored to sampling from the sequence

\[
\pi(x_1), \pi(x_{1:2}), \ldots, \pi(x_{1:n}), \ldots
\]

where the \( X_n \)'s do not necessarily have to evolve from a Markov process. In the case of the HMM, SMC can also be used to infer \( X_{n_1:n_2} \) using \( \pi_\theta(x_{n_1:n_2} \mid y_{1:T}) \). PMCMC is suited to sample from challenging densities of the general form \( \pi(\theta, x_{1:n}) \); again, the \( X_n \)'s do not necessarily have to evolve from a Markov process. PMCMC algorithms make use of SMC in their sampling steps, and we will explain exactly how that is done in the next chapter in Section 2.18.

The later chapters of the thesis build from the methodologies reviewed in Chapter 2 and include our original work. The basic strategy that we follow in our original work is to identify an SMC algorithm that performs with low variance for a particular model (when the parameters for the model are known) and then embed that SMC technique in a PMCMC algorithm. An SMC algorithm will yield an unbiased estimate of its targeted model’s normalising constant, and some PMCMC algorithms actually rely on this unbiased
estimate in order to run. We will explain exactly how these samplers work in the next chapter, but we give this level of detail to the reader now to make it clear why the thesis has a heavy focus on the bias of estimates of normalising constants.

Chapter 3 focuses on the hidden Markov model, and it begins by introducing a new unbiased estimate of the HMM’s normalising constant (also known as the marginal likelihood function). That estimate (for which we also prove a central limit theorem) can be calculated using existing SMC smoothing methods. A study of the estimate leads to

1. the development of new PMCMC algorithms that can be used to target densities of the general form $\pi(\theta, x_{1:n} \mid y_{1:n})$. As part of the algorithms’ settings, a practitioner needs to input an arbitrary sequence of densities $\{\xi_{n,\theta}(x_n)\}$. The choice of sequence will impact the performance of the algorithms, and when each $\xi_{n,\theta}(x_n) = \pi_\theta(x_n \mid y_{1:n-1})$, our new techniques require much less computational time than PMCMC algorithms appearing in the original paper [2];

2. the discovery of a new SMC smoothing technique that has a very low variance across multiple runs because it allows for exact sampling of the target HMM. The technique also requires the algorithmic setting $\xi_{n,\theta}(x_n) = \pi_\theta(x_n \mid y_{1:n-1})$.

The major issue with our proposed methods is that that setting $\xi_{n,\theta}(x_n) = \pi_\theta(x_n \mid y_{1:n-1})$ is typically not analytically available in a real world application of interest; if one can calculate $\pi_\theta(x_n \mid y_{1:n-1})$, then there is likely no need to resort to computational methods to begin with. However, if one has access to a good approximation of $\pi_\theta(x_n \mid y_{1:n-1})$, then our methods can potentially outperform existing SMC and PMCMC strategies. Thus, the latter part of Chapter 3 includes an exploration of alternative smoothing algorithms that approximate the ideal SMC smoothing technique mentioned in point 2 above by approximating $\pi_\theta(x_n \mid y_{1:n-1})$. These alternative smoothing methods unfortunately fail to achieve the desired low variance.

Chapter 4 introduces improved SMC and PMCMC algorithms that enable one to perform Bayesian inference for HMMs whose observations are drawn from distributions with intractable or unknown probability density functions. These types of algorithms have appeared in the literature before (see [51], [52], and [67]), but in our work, we introduce a change of measure on the algorithms of [51] to improve upon them in terms of precision.
and speed (similar to as in [91]). Our methods work particularly well when the true variances of the targeted HMM are very low. Unfortunately, the change in measure that we introduce is often times not analytically available in a real world application (as in [91]), and so it must be approximated when possible. We do show via numerical simulation (of a stochastic volatility HMM with S&P 500 daily logarithmic returns) that the approximation must be very accurate in order for our methods to outperform the methods of [51].

Chapter 5 is entirely concerned with inferring the unknown parameter $\theta$ of a stopped Markov process that is only observed when it reaches the terminal set $A$. Bayesian inference has not been attempted on this class of problems before, and we introduce computational strategies that enable the scientific community to perform such inference for the first time. We infer $\theta$ by developing two new types of PMCMC sampling algorithms. One type adapts the sampling scheme as new values for $\theta$ are drawn, while the other does not. Our numerical studies show that our new methods vastly outperform standard PMCMC methods when one is attempting inference on models from the population genetics literature. Finally, Chapter 6 concludes the thesis with a summary and a discussion of future work.
Chapter 2

Literature Review

This literature review provides the background for the models and numerical techniques that are studied throughout the later chapters of the thesis. The descriptions of the models and algorithms offer details on how they are developed and, in the case of the algorithms, outline their specific steps.

This chapter is organised as follows. Section 2.1 introduces our notation. Sections 2.2 and 2.3 review hidden Markov models and stopped Markov processes, respectively. The review of the models is followed by a review of Monte Carlo techniques, which follows a programme that is similar to that of [29]. We discuss the basics of Monte Carlo methods in Section 2.4, importance sampling in Section 2.5, and sequential importance sampling in Section 2.6. By establishing the principles of these methods, we can introduce sequential Monte Carlo in Section 2.7. Specific sequential Monte Carlo methods tailored for hidden Markov models are reviewed in Sections 2.8, 2.9 and 2.10, and specific methods tailored for stopped Markov processes are discussed in Section 2.11. We look at more advanced sequential Monte Carlo algorithms in Sections 2.12, 2.13 and 2.19.

We also cover Markov chain Monte Carlo methods, with the basic concepts discussed in Section 2.14. The basic methods can struggle to perform well for complex models, and we discuss what this means in more technical detail and introduce some advanced solutions in Sections 2.15, 2.16 and 2.17. Section 2.17 additionally serves the purpose of establishing the theoretical groundwork for particle Markov chain Monte Carlo, which is reviewed in Section 2.18.
Finally, the literature review closes with some notes on implementing Monte Carlo and Markov chain Monte Carlo in parallel computing architectures in Section 2.20. Final remarks are given in Section 2.21.

2.1 Notation

The conventions $\sum_{i=n}^{n-1} = 0$ and $\prod_{\emptyset} = 1$ are used throughout, and $a \land b$ denotes the minimum between the two real numbers $a$ and $b$. When we write $a \ll b$, we mean to say that $a$ is much less than $b$. $\mathbb{Z}$ denotes the set of all integers, and $\mathbb{R}$ denotes the real numbers. The brackets $\lfloor \text{\_} \rfloor$ are used to denote rounding down to the nearest integer.

Consider a random variable $X_k \in \mathbb{R}^{d_x}$ (with index $k$) which may take a value $x_k$. The vector of all $X_k$’s corresponding to all $k \in \{1, \ldots, n\}$ for $n \geq 1$ will be designated $X_{1:}\!n$, the joint density of $X_{1:}\!n$ will be written $\pi_{\theta}(x_{1:}\!n)$.

Each joint density $\pi_{\theta}(x_{1:}\!n)$ may be decomposed as

$$\pi_{\theta}(x_{1:}\!n) = \frac{\gamma_{\theta}(x_{1:}\!n)}{Z_{\theta,1:}\!n},$$

where $Z_{\theta,1:}\!n$ (sometimes written $Z_{\theta}$) is a normalising constant. When a density conditions on a sequence of random variables (and that sequence of random variables is obvious), we may interchangeably use $\pi_{\theta}(x_1 | x_{2:}\!n) = \pi_{\theta}(x_1 | \cdots)$. Any approximations to any density $\pi_{\theta}(x_{1:}\!n)$ will be denoted $\hat{\pi}_{\theta}(x_{1:}\!n)$, and the approximation of the normalising constant will be similarly written as $\hat{Z}_{\theta,1:}\!n$.

A collection of $N$ samples with a common density $\pi_{\theta}(x_{1:}\!n)$ will be written as $\{x_{1:}\!n\}_{i \in \{1, \ldots, N\}}, \{x_{1:}\!N\}$ or $x_{1:}\!n$. We will sometimes assign the value $a_k \in \{1, \ldots, N\}$ to be the index of a sampled value for $X_k$, and in that instance, we allow $x^a_{k} = x^{a(i)}_{k}$. This notation will essentially mean that $x^i_{k+1}$ was sampled dependent on the value of $x^a_{k}$ (i.e., $x^{a(i)}_{k}$ is the ancestor of $x^i_{k+1}$). A sequence of index assignments will be denoted $a_{1:}\!n$. For example, suppose $x^3_3$’s sampling depends on $x^2_2$ and $x^2_2$’s sampling depends on $x^1_1$. Then the ordered
list of indices \((l, j, i)\) can be written \(a_{i,j}^l\).

Some samples will have a weight assigned to them (e.g., the weight of \(x_{1:n}^i\) would be written \(W_n^i (x_{1:n}^i)\)). We will use “\(W_n^i (x_{1:n}^i)\)” and “\(W_n^i\)” interchangeably.

Some of the algorithms reviewed in the thesis will build samples \(\{x_{1:N}^i\}\) in increments where \(n\) increases (e.g., \(\{x_{1:n}^i\}\) to \(\{x_{1:n+1}^i\}\)) or decreases (e.g., \(\{x_{n:T}^i\}\) to \(\{x_{n-1:T}^i\}\)). When it is necessary to distinguish between sampling with increasing values of \(n\) versus decreasing values of \(n\), we will use a system of rightward and leftward arrows. For example, random variables (and their respective sampled values) which are associated with algorithms that increase \(n\) will be written as \(\overrightarrow{X}_n = X_n^i\); random variables which are sampled with decreasing values of \(n\) will be written \(\overleftarrow{X}_n = \overleftarrow{X}_n^i\).

A probability space \((\Omega, \mathcal{F}, \mathbb{P}_\theta)\) consists of a sample space \(\Omega\) and a set of events \(\mathcal{F}\). \(\mathbb{P}_\theta\) is a probability measure defined for every \(\theta \in \Theta\) such that for every \(A \in \mathcal{F}\), \(\mathbb{P}_\theta (A)\) is \(\mathcal{B}(\Theta)\)–measurable. Furthermore, \(\mathcal{M}(\Omega)\) denotes the collection of measures on \(\Omega\) and \(\mathcal{B}(\Omega)\) can also denote the collection of probability measures on \(\Omega\). When clearly stated, \(\mathcal{F}\) may alternatively denote a filtration.

We use \(\delta_a (b)\) to denote the Dirac measure defined for real numbers \(a\) and \(b\) with mass at \(a\). Similarly, for the real-valued numbers \(a\) and \(b\), \(\mathbb{I}_a (b)\) will denote the indicator function that equals one when \(a = b\) and zero when \(a \neq b\). For some measurable space \((E, \mathcal{E})\), let \(B_\epsilon (a) \in \mathcal{E}\) be a ball of radius \(\epsilon\) centred on \(a \in E\). Thus, for \(b \in E\), \(\mathbb{I}_{B_\epsilon (a)} (b)\) will be an indicator function whose value is one when \(b \in B_\epsilon (a)\) and zero otherwise.

For a measurable function \(\varphi : \mathbb{R}^d \to \mathbb{R}\) such that \(\sup_{x \in \mathbb{R}^d} |\varphi(x)| < +\infty\), we write \(\varphi \in B_b (\mathbb{R}^d)\). \(B_b (\mathbb{R}^d)\) is the Banach space that is complete with respect to the norm \(\sup_{x \in \mathbb{R}^d} |\varphi(x)|\).

For a measure space \((E, \mathcal{E}, \sigma)\) and a measurable function \(\varphi(x) : E \to \mathbb{R}\), the essential supremum of \(\varphi\) will be denoted \(\text{ess sup}_x \varphi = \inf \{a \in \mathbb{R} : \sigma (\{x : \varphi(x) > a\}) = 0\}\).

The notation \(\Rightarrow\) will mean “converges almost surely”, whereas \(\Rightarrow\) will mean “converges weakly”.

At times, we will write a vector of zeros with a value of 1 in the \(i\)th position as \(e_i\). The sum of the values of all elements of any \(d\)-dimensional vector \(Y \in (\mathbb{Z}^+ \cup \{0\})^d\) is written \(|Y|\). For example, \(|e_i| = 1\). The \(m \times n\) transpose of a matrix \(A \in \mathbb{R}^{n \times m}\) is \(A^\top\). The inverse of a square matrix \(A\) will be \(A^{-1}\). The identity matrix is written as \(I\).
Chapter 2. Literature Review

Standard notation $\mathbb{E}_\pi[X]$ is adopted for the expected value of a random variable $X$ with respect to $\pi(x)$. Similarly, the variance of the same random variable is denoted $\mathbb{V}_\pi[X]$. The covariance of two random variables $X$ and $Y$ with respect to a measure $\pi$ will be $\text{Cov}_\pi[X, Y]$.

Finally, any distributions and acronyms used in the thesis can be found in Tables 2.1 and 2.2, respectively.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Parameters</th>
<th>Notation</th>
<th>Expected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamma</td>
<td>shape $\alpha &gt; 0$ and scale $\beta &gt; 0$</td>
<td>$\mathcal{G}(\alpha, \beta)$</td>
<td>$\alpha\beta$</td>
</tr>
<tr>
<td>Gaussian</td>
<td>mean $\mu \in (-\infty, \infty)$ and variance $\sigma^2 &gt; 0$</td>
<td>$\mathcal{N}(\mu, \sigma^2)$</td>
<td>$\mu$</td>
</tr>
<tr>
<td>Geometric</td>
<td>success probability $0 &lt; p \leq 1$</td>
<td>$\mathcal{G}(p)$</td>
<td>$p^{-1}$</td>
</tr>
<tr>
<td>Stable</td>
<td>stability $\alpha \in (0, 2]$, skewness $\beta \in [-1, 1]$, scale $\gamma &gt; 0$, and location $\delta \in (-\infty, \infty)$</td>
<td>$\mathcal{S}(\alpha, \beta, \gamma, \delta)$</td>
<td>$\mu$ when $\alpha &gt; 1$</td>
</tr>
<tr>
<td>Uniform</td>
<td>boundaries $-\infty &lt; a &lt; b &lt; \infty$</td>
<td>$\mathcal{U}(a, b)$</td>
<td>$\frac{1}{2}(a + b)$</td>
</tr>
</tbody>
</table>

2.2 Hidden Markov models

We begin with an introduction to the models and Markov processes used in this work, as well as an introduction to their associated inference problems. This first section offers a review of the hidden Markov model and the filtering and smoothing inference problems for HMMs. We then review stopped Markov processes, their inference problems, and their application within population genetics.

Consider an evolving discrete time Markov process. At each time point $n \geq 1$, the state of this process is denoted by the random variable $X_n \in \mathbb{R}^d$. Assume that we cannot directly observe $X_n$, but we can only indirectly observe this latent state through the random variable $Y_n \in \mathbb{R}^d$. Assume also that the observations are statistically independent of one another, conditioned upon the latent process. This model is called a hidden Markov model.
Table 2.2: Acronyms used throughout the thesis

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>Approximate Bayesian computation</td>
</tr>
<tr>
<td>ACF</td>
<td>Autocorrelation function</td>
</tr>
<tr>
<td>APF</td>
<td>Auxiliary particle filter</td>
</tr>
<tr>
<td>CPU</td>
<td>Central processing unit</td>
</tr>
<tr>
<td>ESS</td>
<td>Effective sample size</td>
</tr>
<tr>
<td>FFBS</td>
<td>Forward filtering backward smoothing</td>
</tr>
<tr>
<td>FFBSi</td>
<td>Forward filtering backward simulation</td>
</tr>
<tr>
<td>GIMH</td>
<td>Grouped independence Metropolis-Hastings</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphical processing unit</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov model</td>
</tr>
<tr>
<td>IS</td>
<td>Importance sampling</td>
</tr>
<tr>
<td>M-H</td>
<td>Metropolis-Hastings</td>
</tr>
<tr>
<td>MC</td>
<td>Monte Carlo</td>
</tr>
<tr>
<td>MCMC</td>
<td>Markov chain Monte Carlo</td>
</tr>
<tr>
<td>ML</td>
<td>Multi-level</td>
</tr>
<tr>
<td>MRCA</td>
<td>Most recent common ancestor</td>
</tr>
<tr>
<td>MS</td>
<td>Multi-level splitting</td>
</tr>
<tr>
<td>PG</td>
<td>Particle Gibbs</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>SIS</td>
<td>Sequential importance sampling</td>
</tr>
<tr>
<td>SMC</td>
<td>Sequential Monte Carlo</td>
</tr>
<tr>
<td>STR</td>
<td>Stopped time resampling</td>
</tr>
</tbody>
</table>

[13], and it can be formally written as

\[
X_n \mid (X_{1:n-1} = x_{1:n-1}, Y_{1:n-1} = y_{1:n-1}) \sim f_\theta(\cdot \mid x_{n-1})
\]

\[
Y_n \mid (X_{1:n} = x_{1:n}, Y_{1:n-1} = y_{1:n-1}) \sim g_\theta(\cdot \mid x_{n}),
\]

for \( n \geq 1 \) where \( X_1 \sim f_\theta(\cdot \mid x_0) = \mu_\theta(\cdot). \) \( \theta \in \Theta \subseteq \mathbb{R}^d \) is a static parameter which may or may not be known. HMMs are very flexible, and so they are used in a wide array of real world applications. Some examples include stochastic volatility models [62], real-time hand writing recognition [47], and DNA segmentation [75].
When $\theta$ is known, inference on the hidden process at time $n$ relies on the joint density

$$
\pi_\theta (x_{1:n} \mid y_{1:n}) = \frac{\gamma_\theta (x_{1:n}, y_{1:n})}{Z_{\theta,1:n}} = \frac{\prod_{k=1}^{n} \gamma_\theta (y_k \mid x_k) f_\theta (x_k \mid x_{k-1})}{\int \prod_{k=1}^{n} \gamma_\theta (y_k \mid x_k) f_\theta (x_k \mid x_{k-1}) \, dx_{1:n}}. \tag{2.2.1}
$$

The normalising constant $Z_{\theta,1:n}$ is the probability density of the observations given $\theta$ (i.e., $Z_{\theta,1:n} = p_\theta (Y_{1:n} = y_{1:n}) = p_\theta (y_{1:n})$). It is often referred to as the marginal likelihood. Two densities of interest which are related to (2.2.1) are the filtering density and the smoothing density [29] (see below). If $\theta$ is unknown, then we would be interested in inferring not only the hidden process but also values for $\theta$. In this case, we assign a prior density $\pi (\theta)$, and Bayesian inference at time $n$ relies on the joint density

$$
\pi (\theta, x_{1:n} \mid y_{1:n}) \propto \pi (\theta) \gamma_\theta (x_{1:n}, y_{1:n}). \tag{2.2.2}
$$

### 2.2.1 Filtering

In the filtering problem, one infers the state of the hidden process each time a new observation is made available, and so at time $n$, the density of interest is

$$
\pi_\theta (x_n \mid y_{1:n}) = \int \pi_\theta (x_{1:n} \mid y_{1:n}) \, dx_{1:n-1}. \tag{2.2.3}
$$

Computing (2.2.3) can be difficult, especially when $n$ is large and one is presented with a complex version of (2.2.1). To make the integration potentially less cumbersome, there exists a recursive relationship that links $\pi_\theta (x_n \mid y_{1:n})$ to $\pi_\theta (x_{n-1} \mid y_{1:n-1})$:

$$
\pi_\theta (x_n \mid y_{1:n}) \propto g_\theta (y_n \mid x_n) \int f_\theta (x_n \mid x_{n-1}) \pi_\theta (x_{n-1} \mid y_{1:n-1}) \, dx_{n-1}. \tag{2.2.4}
$$

Thus, the filtering density computed at time $(n-1)$ can be used to calculate the filtering density at time $n$.

Even the recursion (2.2.4) will be analytically intractable in most applications, in which case one will have to resort to numerical approximation methods. However, when the
Algorithm 1 Kalman filter

Let $\mu_n$ and $\sigma^2_n$ be the mean and covariance, respectively, of $\pi_\theta (x_n \mid y_{1:n-1})$. Let $\mu_n$ and $\sigma^2_n$ be the mean and covariance, respectively, of $\pi_\theta (x_n \mid y_{1:n})$.

At time step $n > 1$,

- Prediction step: calculate $\mu_n = B_n + A_n \mu_{n-1}$ and $\sigma^2_n = A_n \sigma^2_{n-1} A^T_n + \Sigma_n$.

- Update step: calculate the Kalman gain $K_n = \sigma^2_n C^T_n \left( C_n \sigma^2_n C^T_n + \tau^2_n \right)^{-1}$ for

$$
\mu_n = \mu_n + K_n (y_n - C_n \mu_n)
$$

$$
\sigma^2_n = (I - K_n C_n) \sigma^2_n
$$

HMM of interest is linear Gaussian,

$$
X_1 \sim f_\theta (\cdot \mid x_0) = \mathcal{N} (\mu_0, \Sigma_0)
$$

$$
X_n \mid X_{1:n-1} \sim f_\theta (\cdot \mid x_{n-1}) = \mathcal{N} (B_n + A_n x_{n-1}, \Sigma_n)
$$

$$
Y_n \mid Y_{1:n-1}, X_{1:n} \sim g_\theta (\cdot \mid x_n) = \mathcal{N} (C_n x_n, \tau^2_n)
$$

(with $A_n$, $B_n$, and $C_n$ being known matrices of appropriate dimension), then the Kalman filter [56] (see Algorithm 1) can be used to compute the recursion (2.2.4) exactly.

We briefly note that there are extensions to Algorithm 1 specifically designed for deterministically approximating the hidden states of non-linear Gaussian HMMs. The extended Kalman filter and the unscented Kalman filter [55] are two examples. When $f_\theta$ and $g_\theta$ are non-linear, they can only be used to calculate $\mu_n$ and $\mu_n$. Exact deterministic equations of $\sigma^2_n$ and $\sigma^2_n$ are unavailable. However, provided that the state and observation equations are differentiable, the extended Kalman filter can use Jacobians of those equations to linearise them so that at least approximations of $\sigma^2_n$ and $\sigma^2_n$ can be computed. This technique is difficult to tune and really only reliable when the HMM in question is nearly linear. The unscented Kalman filter offers a more accurate and robust solution when a HMM is highly non-linear. At a time point $n$, the unscented Kalman filter deterministically chooses a set of points around $\mu_n$. The points are then propagated through the state and observation equations, and $\mu_n$ and $\sigma^2_n$ are recovered through these simulations.
2.2.2 Smoothing

In the smoothing problem, all of the observations up to time $T > 1$ are presented, and one infers the hidden state at a time point $t < T$:

$$
\pi_\theta (x_t \mid y_{1:T}) = \int \pi_\theta (x_{t:1:T} \mid y_{1:T}) \, dx_{1:t-1,t+1:T}.
$$ (2.2.5)

Actually, to be even more general, one might infer the state throughout a block of time:

$$
\pi_\theta (x_{t_1:t_2} \mid y_{1:T}) = \int \pi_\theta (x_{1:1:T} \mid y_{1:T}) \, dx_{1:t_1-1,t_2+1:T},
$$

for $1 \leq t_1 \leq t_2 \leq T$.

If one has access to the sequence of filtering densities, then, similar to the filtering problem, there exist two well-known recursive methods that can ease the burden of computing the smoothing density. The forward filtering backward smoothing (FFBS) recursion,

$$
\pi_\theta (x_n \mid y_{1:T}) = \pi_\theta (x_n \mid y_{1:n}) \int \frac{\pi_\theta (x_{n+1} \mid y_{1:T}) f_\theta (x_{n+1} \mid x_n)}{\pi_\theta (x_{n+1} \mid y_{1:n})} \, dx_{n+1},
$$ (2.2.6)

relies on $\{\pi_\theta (x_k \mid y_{1:k})\}_{k \in \{1,\ldots, T\}}$ to compute $\pi_\theta (x_n \mid y_{1:T})$ at each time $n$ using the density $\pi_\theta (x_{n+1} \mid y_{1:T})$ obtained at time $(n+1)$. The two-filter formula of [9] is an alternative to the FFBS recursion. In this algorithm, one uses a “backward information filter”,

$$
\pi_\theta (y_T \mid x_T) = g_\theta (y_T \mid x_T)
$$

$$
\pi_\theta (y_n,T \mid x_n) = g_\theta (y_n \mid x_n) \int f_\theta (x_{n+1} \mid x_n) \pi_\theta (y_{n+1:T} \mid x_{n+1}) \, dx_{n+1},
$$

to compute $\pi_\theta (y_{t:T} \mid x_t)$ for $t < T$; the backward information filter is a recursive formula that is not a probability density in the argument $x_n$. The two-filter formula then gives

$$
\pi_\theta (x_t \mid y_{1:T}) = \frac{\pi_\theta (x_t \mid y_{1:t-1}) \pi_\theta (y_{t:T} \mid x_t)}{p_\theta (y_{t:T} \mid y_{1:t-1})} \propto \pi_\theta (x_t \mid y_{1:t-1}) \pi_\theta (y_{t:T} \mid x_t).
$$ (2.2.7)
2.2 Hidden Markov models

2.2.3 Towards the approximation of hidden Markov models

In the later sections of this thesis, we will present simulation techniques that can be used to perform Bayesian inference for HMMs, and some of those simulation techniques will require repetitious calculation of the likelihood function \( g_\theta \). For instances where this calculation is expensive or impossible (such as in some stochastic volatility models [51] and portfolio allocation problems [52]), we will also review other simulation techniques that bypass the calculation of \( g_\theta \) at the expense of introducing a bias. More specific details will follow in subsequent sections of the thesis (namely, Section 2.10), but we introduce here some mathematical framework that will be essential in developing these alternative, biased methods.

Allow the sequence \( \{X_n\}_{n \geq 1} \) to be an \((E, \mathcal{E})\)—valued discrete-time Markov process with transition densities \( \{M_\theta (x_n \mid x_{n-1})\}_{n \geq 2} \) and \( M_\theta (x_1 \mid x_0) = \mu (x_1) \).

\[ (2.2.8) \]

for some static parameter \( \theta \in \Theta \subseteq \mathbb{R}^{d_\theta} \). Considering the sequence of measurable spaces \( \{(E_n, \mathcal{E}_n)\}_{n \geq 1} \) and the sequence of points \( \{a_n \in E_n\}_{n \geq 1} \), we define a sequence of indicator potentials for each \( X_n = x_n \):

\[ \{W_n (x_n) = \mathbb{I}_{B_n, \epsilon (a_n)} (x_n)\}_{n \geq 1}, \]

\[ (2.2.9) \]

where \((x_n, B_{n, \epsilon} (x_n)) \in E_n \times \mathcal{E}_n\). Thus, as the sequence \( \{X_n\}_{n \geq 1} \) propagates, the values taken by the Markov process are assigned weights (2.2.9) according to their proximity to a sequence of points \( \{a_n \in E_n\}_{n \geq 1} \).

This process’s connection to HMMs is established with a more specific definition of \( \{X_n\}_{n \geq 1} \). Allow \( \{Y_n\}_{n \geq 1} \) to be the \((H, \mathcal{H})\)—valued discrete-time observations of a hidden \((R, \mathcal{R})\)—valued Markov process \( \{K_n\}_{n \geq 1} \), and let \( E = R \times H \). By [52], an approximation of the joint density (2.2.1) is

\[ \pi_\theta (k_{1:n} \mid y_{1:n}) = \frac{\prod_{t=1}^{n} g_\theta (y_t \mid k_t) f_\theta (k_t \mid k_{t-1})}{\int \prod_{t=1}^{n} g_\theta (y_t \mid k_t) f_\theta (k_t \mid k_{t-1}) dk_{1:n}}, \]

\[ (2.2.10) \]
where
\[ g_{\theta}^{(t)}(y_{t} \mid k_{t}) = \frac{\int_{B_{\epsilon}(y_{t})} g_{\theta}(u \mid k_{t}) \, du}{\int_{B_{\epsilon}(y_{t})} \, du}. \]

We set (2.2.8) as
\[ \{ M_{\theta}(x_{n-1}, x_{n}) = f_{\theta}(k_{n} \mid k_{n-1}) \, g_{\theta}(u_{n} \mid k_{n}) \}_{n \geq 2} \tag{2.2.11} \]
and (2.2.9) as
\[ \{ W_{n}(x_{n}) = \mathbb{I}_{\mathbb{R} \times B_{n,\epsilon}(y_{n})}(x_{n}) \}_{n \geq 1}, \tag{2.2.12} \]
where \( x_{n} = (k_{n}, u_{n}) \) and \( B_{n,\epsilon}(y_{n}) \in \mathcal{H} \). Now, the sequence \( \{ X_{n} \}_{n \geq 1} \) propagates with \( k_{n} \sim f_{\theta}(\cdot \mid k_{n-1}) \) and \( u_{n} \sim g_{\theta}(\cdot \mid k_{n}) \). The values taken by this propagating sequence are assigned weights (2.2.12) according to whether or not \( u_{n} \in B_{n,\epsilon}(y_{n}) \). Under strong assumptions, \([52, \text{Theorem 1}]\) and \([52, \text{Theorem 2}]\) show that (2.2.10) is a consistent approximation of (2.2.1) as \( \epsilon \) tends to zero.

### 2.3 Stopped Markov processes

A second type of Markov process which is of interest in this thesis is the partially observed, discrete time stopped Markov process \([23, \text{Section 2.2.3}]\). In this model, a Markov process evolves in discrete time until it reaches a target set \( A \) and terminates. The Markov chain is only observed once it has reached \( A \). This process is used to model phenomena in neuroscience \([8]\), finance \([15]\), and population genetics \((\,[22],[58]\))

In more detail, let \( \theta \in \Theta \subseteq \mathbb{R}^{d_{\theta}} \) be a parameter with an associated prior \( \pi(\theta) \). Now let the sequence \( \{ X_{n} \}_{n \geq 1} \) be an \( (E, \mathcal{E}) \)-valued discrete-time Markov process defined on a probability space \( (\Omega, \mathcal{F}, \mathbb{P}_{\theta}) \). The sequence \( \{ X_{n} \}_{n \geq 1} \) begins its evolution in a non-empty set \( B_{0} \in \mathcal{F} \) with some initial distribution and some appropriate Markov transition kernel \( f_{\theta}(x_{n} \mid x_{n-1}) \) (with \( f_{\theta}(x_{1} \mid x_{0}) = \mu_{\theta}(x_{1}) \)). The process is killed once it reaches the target
set $A \in \mathcal{F}$ (note that $\mathbb{P}_\theta(X_1 \in A) = 0$). The stopping time is defined as

$$\tau = \inf \{ n \geq 2 : X_n \in A \}, \quad \tau \in \mathcal{I},$$

(2.3.1)

where it is assumed that $\mathbb{P}_\theta(\tau < \infty) = 1$ (as in [50]) and $\mathcal{I}$ is a collection of positive integer values related to possible stopping times. With the final state of the process (i.e., the data) being written as $y = x_\tau$, Bayesian inference on $\theta$ relies on the posterior density

$$\pi(\theta, x_{1:\tau}, \tau \mid y) \propto \pi(\theta) \gamma( x_{1:\tau}, \tau, y),$$

(2.3.2)

where

$$\gamma( x_{1:\tau}, \tau, y) = \mathbb{I}_{\{a \in (A \times (A^c)^{\tau-1})\}}( x_{1:\tau}) \pi(\theta)( y \mid x_{1:\tau}) \left[ \prod_{n=1}^\tau f( x_n \mid x_{n-1}) \right]$$

(2.3.3)

is the complete-data likelihood with normalising constant

$$Z_{\theta,1:\tau} = \sum_{\tau \in \mathcal{I}} \int_{E_{\tau}} \gamma( x_{1:\tau}, \tau, y) \, dx_{1:\tau}.$$  

(2.3.4)

The density $\pi(\theta \mid x_{1:\tau})$ is the conditional density of the data given the trajectory of the process.

Section 2.11 will introduce a numerical technique which one can employ to sample the stopped Markov process from $\pi(\theta \mid x_{1:\tau} \mid y)$. We will see that simulation of $\{X_n\}_{1 \leq n \leq \tau}$ becomes much easier when one re-defines the stopped Markov process as another stopped Markov process, $\{\mathcal{X}_l\}_{1 \leq l \leq p}$, that is guided to propagate into the terminal set $A$. We do not yet explain how the simulation methods will work, but we do feel that this is an appropriate place to introduce the mathematical groundwork for Section 2.11. Thus, we commence by defining $\{\mathcal{X}_l\}_{1 \leq l \leq p}$ here.

Consider an arbitrary sequence of $\mathcal{F}$-sets $B_0, B_1, \ldots, B_p$ (with $B_p = A$) that, in some way, must interpolate between $B_0$ and $A$ (see [17] and [23, Section 12.2]). As a stopped Markov process propagates, it will reach each set $B_l$ at a stopping time

$$T_l = \inf\{ n \geq 2 : X_n \in B_l \}.$$  

(2.3.5)
To ensure that the process passes through each of the $\mathcal{F}$-sets $B_0, B_1, \ldots, B_p$ in order and to ensure that the terminal set $A$ is not reached before some $B_l$ is reached, we impose the restriction that the $\mathcal{F}$-sets must be defined such that

$$0 \leq T_1 \leq \cdots \leq T_p = \tau. \tag{2.3.6}$$

With the condition (2.3.6), knowledge of $\{x_{1:\tau}, T_{1:p-1}\}$ implies knowledge of the event $T_p = \tau$. Furthermore, each $T_l$ must be finite and integer-valued, as we assumed above that $\mathbb{P}(\tau < \infty) = 1$ (see just below (2.3.1)).

For clarity on how $\mathcal{F}$-sets may be defined, let us consider an example. If $B_0 \supset A$, we could define the sets as

$$B_0 \supset B_1 \supset \cdots \supset B_p = A, \quad p \geq 2$$

with the corresponding stopping times defined as (2.3.5). The condition (2.3.6) is implied.

The $\mathcal{F}$-sets can be used to redefine $\{X_n\}_{1 \leq n \leq \tau}$ as $\{X_l\}_{1 \leq l \leq p}$:

- $X_1 = (x_{1:\tau_1}, \tau_1)$
- $X_l = (x_{\tau_{l-1}+1:\tau_l}, \tau_l)$ for $2 \leq l \leq p$
- $X_1 \in \{x_{1:\tau_1}, \tau_1 : x_{1:1} \in B_0, x_{1:\tau_1-1} \notin B_1, x_{\tau_1} \in B_1\}$
- $X_l \in \{x_{\tau_{l-1}+1:\tau_l}, \tau_l : x_{\tau_{l-1}+1:\tau_l-1} \in B_{l-1}, x_{\tau_{l-1}+1:1} \notin B_l, x_{\tau_l} \in B_l\}$,

where $\tau_1$ is a realisation of the stopping time $T_1$. It follows from [23, Propositions 12.2.2 and 12.2.4] that the sequence $\{X_l\}_{1 \leq l \leq p}$ forms a Markov chain taking values in

$$E_l = \bigcup_{\tau_1 \in I_l} \{\tau_1\} \times E^{\tau_1},$$

Furthermore, there exists a natural sequence of probability densities

$$\left\{ \frac{\pi_{\theta}(X_{1:l})}{Z_{\theta,1:l}} \right\}_{1 \leq l \leq p} \tag{2.3.7}$$
such that $\gamma_\theta(X_{1:p})$ is equivalent to (2.3.3):

\[
\gamma_\theta(X_1) = \prod_{n=1}^{\tau_1} f_\theta(x_n|x_{n-1}) I_{\{a \in (B_0 \cap B_1^c)\}}(x_{\tau_1-1}) I_{\{a \in B_1\}}(x_{\tau_1}),
\]

\[
\gamma_\theta(X_{1:l}) = \gamma_\theta(X_{1:l-1}) \prod_{n=\tau_{l-1}+1}^{\tau_l} f_\theta(x_n|x_{n-1}) I_{\{a \in (B_{l-1} \cap B_l^c)\}}(x_{\tau_{l-1}+1:\tau_l-1}) I_{\{a \in B_l\}}(x_{\tau_l}),
\]

\[
\gamma_\theta(X_{1:p}) = \gamma_\theta(X_{1:p-1}) \prod_{n=\tau_{p-1}+1}^{\tau_p} f_\theta(x_n|x_{n-1}) \times \pi_\theta(y \mid x_{1:p}) I_{\{a \in (B_{p-1} \cap \bar{A})\}}(x_{\tau_{p-1}+1:p-1}) I_{\{a \in \bar{A}\}}(x_{\tau_p}).
\]

(2.3.8)

Due to the relationship $\gamma_\theta \equiv \gamma_\theta$ (and hence $Z_\theta \equiv Z_\theta$), we have an identity for moving between the sequences $\{X_n\}_{1 \leq n \leq \tau}$ and $\{X_l\}_{1 \leq l \leq p}$.

### 2.3.1 The Wright-Fisher model and Kingman’s coalescent

In this thesis, several numerical examples will concentrate on the stopped Markov process’s applications in population genetics, and so we now turn to descriptions of the Wright-Fisher model and Kingman’s coalescent [58].

**Wright-Fisher model**

We start with the Wright-Fisher model. Consider a very large population of $N$ individuals. The population will evolve over time, but $N$ is held constant. By evolve, we mean that the individuals from generation $G_{n-1}$ die off immediately after giving birth to the individuals of generation $G_n$. The generations $G_1$, $G_2$, $G_3$, … are assumed to be non-overlapping, and we assume that the time index has no minimum and can be negative. Each child of generation $G_n$ has exactly one parent from generation $G_{n-1}$, but the number of children born to the $j^{\text{th}}$ member of $G_{n-1}$ is a random variable $v_j$, with $\sum_{i=1}^N v_i = N$; we assume here that the $v_j$’s collectively follow a symmetric multinomial distribution [58]. Given these assumptions, one can show – as on [58, page 29] – that there must exist some generation $G_{n-1}$ back in time that contains a most recent common ancestor (MRCA) from which all members of generation $G_n$ descended. The value $l$ is the realisation of some random
variable $L$. Thus, to get better insight into the structure of the Wright-Fisher model and the
time at which the MRCA appeared, we would like to determine a density for the random
variable $L$. Kingman’s coalescent [58] allows us to approximate this density.

**Kingman’s coalescent**

In the coalescent model, one starts with a sample of $m \leq N$ individuals. The sample now
evolves backward in time: two randomly chosen individuals at time $(n + 1)$ coalesce, and
this event yields a new sample at time $n$ whose size is reduced by one. [58] proves that this
coalescent model converges to the Wright-Fisher model as $N \to \infty$ (i.e., the coalescent
describes the ancestral process of the original sample of size $m$ as $N \to \infty$). This result
is powerful, as the coalescent model has many useful properties that help to determine
the nature of $L$. Namely, consider the sequence $\{T_n\}_{2 \leq n \leq m}$, where each $T_n$ is a random
variable corresponding to the time required for the sample to evolve from $n$ individuals to
$(n - 1)$ individuals. [58] shows that, as $N \to \infty$, the times $\{T_{2:m}\}$ are independent of one
another, and each $T_n$ is exponentially distributed with rate parameter $\frac{n(n-1)}{2}$. Thus,
if one starts with a sample of $m$ individuals, the density of $L$ can be obtained by

$$f(t_2, \ldots, t_m) = \prod_{n=2}^{m} \frac{n(n-1)}{2} \exp \left[ - \frac{n(n-1)}{2} t_n \right],$$

and the approximation becomes exact as $N \to \infty$.

There are extensions to the coalescent of [58]. Two numerical examples in Chapter
5 are concerned with the coalescent with mutation and the coalescent with mutation and
migration:

- The coalescent with mutation assigns each member of the sample $m$ a genetic type
  from an exhaustive and mutually exclusive classification scheme of $d$ different types.
  As the sample evolves and coalesces, any randomly chosen member could mutate
  into another of the $d - 1$ other types.

- The coalescent with mutation and migration is a collection of coalescent trees that
  allows for the migration of individuals from one tree to another. It is only within each
  of the sub-trees that mutation and coalescent activity occurs [22].
2.3 Stopped Markov processes

In both models, the random time is independent of the mutation and migration rates. Thus, the random time can be removed and the stopped Markov process of Section 2.3 offers a framework through which a practitioner could infer the mutation and migration rates when given a sample of size \( m \). Following the ideas of [22] and [44], we now formulate these models in the language of discrete time stopped Markov processes \( \{X_n\}_{n \geq 1} \), where time steps \( n \) increase in the direction of the MRCA evolving down to a sample of size \( m \).

Coalescent with mutation

Starting with the coalescent with mutation, each random variable within the sequence can be defined as

\[
X_n = (X_n^1, X_n^2, \ldots, X_n^d)
\]

where each \( X_n^i \) is the number of individuals of type \( i \). The chain begins as a vector of zeros with a one in the position of the MRCA, and the sequence is initialised by the density

\[
f_\theta(x_1 \mid x_0) = \begin{cases} 
\nu_i & \text{if } x_1 = e_i \\
0 & \text{otherwise},
\end{cases}
\]

where \( \nu_i \) is the \( i \)th element of the stationary density of the Markov chain; this holds for any \( i \in \{1, \ldots, d\} \). The sequence propagates from time \((n-1)\) to time \( n \) by first selecting an individual at random with probability \( \frac{x_i}{|x_{n-1}|} \). According to [44], mutations occur according to a Poisson process with rate \( \frac{\mu}{2} \) and splits follow an exponential distribution of rate \( \frac{1}{|x_{n-1}|^2} \). Thus, after selecting an individual, a mutation occurs with probability \( \frac{\mu}{|x_{n-1}|+\mu} \) and a split occurs with probability \( \frac{|x_{n-1}|-1}{|x_{n-1}|-1+\mu} \). Additionally, if a mutation occurs, one assigns a probability \( r_{il} \) to each mutation of type \( i \) to type \( l \). The one-step transition density of the sequence is

\[
f_\theta(x_n \mid x_{1:n-1}) = f_\theta(x_n \mid x_{n-1}) = \begin{cases} 
\frac{x_i}{|x_{n-1}|} \frac{\mu}{|x_{n-1}|+\mu} r_{il} & \text{if } x_n = x_{n-1} - e_i + e_l \text{ (mutation)} \\
\frac{x_i}{|x_{n-1}|} \frac{|x_{n-1}|-1}{|x_{n-1}|-1+\mu} & \text{if } x_n = x_{n-1} + e_i \text{ (split)} \\
0 & \text{otherwise},
\end{cases}
\]
where $\theta = \{\mu, R\}$ is a static parameter; note that $r_{ii} \neq 0$ is permitted. The sequence is stopped at time $n = \tau > 1$ when $|x_\tau| = m$, and we set $y = y^{1:d} = x_\tau$ with $\pi_\theta(y \mid x_{1:\tau}) = \prod_{i=1}^{d} y_i! / m!$. An example of the coalescent with mutation is presented in Figure 2.3.1.

Figure 2.3.1: In this example of the coalescent model with mutation, $d = 2$. The tree propagates forward in time from the MRCA (labeled “2”) downwards by a sequence of genetic events (i.e., splits and mutations). Arrows denote a mutation of one type of individual to another. This figure is only a particular realisation of the ancestry and is not the only possible tree that could link a MRCA to the dataset $y$. For example, the bottom of the tree could have been obtained by setting “1” to be the MRCA. The sequence of genetic events could then be split, mutation, split, split, mutation, and split.

The chain just described starts at $x_1 \in B_0 \in \mathcal{F}$ and terminates when the set $A = \{x_\tau \in (\mathbb{Z}^+ \cup \{0\})^d : |x_\tau| = m\}$ is reached. If one wanted to redefine $\{X_n\}_{1 \leq n \leq \tau}$ as $\{X_l\}_{1 \leq l \leq p}$, an arbitrary sequence of sets $B_0, B_1, \ldots, B_p$ (with $B_p = A$) would be required. There are multiple ways to structure the $\mathcal{F}$–sets, but one possible formulation would be

$$B_0 = \{x_1 \in (\mathbb{Z}^+ \cup \{0\})^d : |x_1| = 1\},$$
$$B_l = \{x_n \in (\mathbb{Z}^+ \cup \{0\})^d : |x_n| = h_l\}, \ 1 \leq l \leq p,$$

where $1 < h_1 < h_2 < \cdots < h_p = m$ is a collection of integers. The sets are defined such that the associated stopping times are (2.3.5) and they satisfy (2.3.6). Also, note that these
sets do interpolate between the MRCA and \( y \).

## 2.3 Stopped Markov processes

### Coalescent with mutation and migration

A similar stopped Markov process representation can be structured for the coalescent with mutation and migration as follows. Consider the discrete time process \( \{X_n\}_{n \geq 1} \) moving forward in time (i.e., from MRCA to a sample of size \( m \)). We assign \( g \) to be the number of sub-groups in this coalescent model. The state at time \( n \) is the concatenation of the different genetic types that comprise each of the \( g \) sub-groups:

\[
X_n = (X_{1,n}^1, X_{1,n}^2, \ldots, X_{1,n}^d, \ldots, X_{g,n}^1, X_{g,n}^2, \ldots, X_{g,n}^d)
\]

where \( X_{j,n}^i \) is the number of individuals of type \( i \) in sub-group \( j \). The chain is again initialised by the density (2.3.10), and the process undergoes transitions according to

\[
\begin{align*}
X_n &= X_{n-1} + e_{\alpha,i} \quad \text{(split)} \\
X_n &= X_{n-1} - e_{\alpha,i} + e_{\alpha,l} \quad \text{(mutation)} \\
X_n &= X_{n-1} - e_{\alpha,i} + e_{\beta,i} \quad \text{(migration)},
\end{align*}
\]

where \( \alpha, \beta \in \{1, \ldots, g\}, \alpha \neq \beta \).

The transition probabilities are parametrised by the mutation parameter \( \mu \) and the mutation matrix \( R \), as before. Additionally, the transition probabilities are parametrised by the migration matrix \( G \), which is a symmetric matrix with zero values on the diagonal and positive values on the off-diagonals. Each element \( g_{il} \) gives the rate associated to migrating from sub-group \( i \) to sub-group \( l \). Thus, we allow the sum of the rates of all possible events across all sub-groups at time \( n \) to be

\[
\frac{1}{2} D(x_n) = \sum_{j=1}^{g} \left[ \frac{\left(x_{j,n}\right)}{2} \left(\frac{x_n}{x_{j,n}}\right) + \frac{|x_{j,n}|}{2} + \frac{|x_{j,n}| \sum_{i=1}^{g} g_{ji}}{2} \right]
= \frac{1}{2} \sum_{j=1}^{g} \left[ |x_{j,n}| \left( (|x_{j,n}| - 1) |x_n|/|x_{j,n}| + \mu + \left( \sum_{i=1}^{g} g_{ji} \right) \right) \right].
\]
Within a sub-group $\alpha$, a type $i$ is selected at random with probability $\frac{x^i_{\alpha,n-1}}{|x_{\alpha,n-1}|}$ at time $(n-1)$, and so as splits follow exponential distributions of rate $\left(\frac{|x_{\alpha,n-1}| - 1}{2}\right)$, an individual of type $i$ splits with probability
\[
\frac{x^i_{\alpha,n-1}}{|x_{\alpha,n-1}|} \cdot \frac{|x_{\alpha,n-1}|(|x_{\alpha,n-1}| - 1)}{2} \cdot \frac{2}{D(x_{n-1})} = \frac{x^i_{\alpha,n-1}}{|x_{\alpha,n-1}|} \cdot \frac{|x_{\alpha,n-1}|(|x_{\alpha,n-1}| - 1)}{D(x_{n-1})}.
\]

Mutation from type $i$ to type $l$ occurs with probability
\[
\frac{x^i_{\alpha,n-1}}{|x_{\alpha,n-1}|} \cdot \mu \frac{|x_{\alpha,n-1}|}{D(x_{n-1})} \cdot r_{il} = \frac{x^i_{\alpha,n-1}}{|x_{\alpha,n-1}|} \cdot \mu \frac{r_{il}}{D(x_{n-1})},
\]
and an individual migrates from sub-group $\alpha$ to sub-group $\beta$ with probability
\[
\frac{x^i_{\alpha,n-1}}{|x_{\alpha,n-1}|} \cdot \frac{|x_{\alpha,n-1}| g_{\alpha\beta}}{D(x_{n-1})} = \frac{x^i_{\alpha,n-1}}{|x_{\alpha,n-1}|} \cdot \frac{g_{\alpha\beta}}{D(x_{n-1})}.
\]

The one-step transition density of the sequence is
\[
f_\theta(x_n|x_{1:n-1}) = f_\theta(x_n|x_{n-1}) = \begin{cases} 
\frac{x^i_{\alpha,n-1}}{D(x_{n-1})} g_{\alpha\beta} & \text{if } x_n = x_{n-1} - e_{\alpha,i} + e_{\beta,i} \\
\mu \frac{x^i_{\alpha,n-1}}{D(x_{n-1})} r_{il} & \text{if } x_n = x_{n-1} - e_{\alpha,i} + e_{\alpha,l} \\
\frac{x^i_{\alpha,n-1}}{|x_{\alpha,n-1}|} \cdot |x_{n-1}|(|x_{\alpha,n-1}| - 1) & \text{if } x_n = x_{n-1} + e_{\alpha,i} \\
0 & \text{otherwise.}
\end{cases}
\]

We stop the process at time $\tau$ and set $y = y^{1:gd} = x_\tau$ when the number of individuals in the population reaches $m$, with $\pi_\theta(y \mid x_{1:\tau}) = \prod_{i=1}^{gd} \frac{y_i^{gd}}{m!}$. The $\mathcal{F}$–sets
\[
B_0 = \{ x_1 \in (\mathbb{Z}^+ \cup \{0\})^{gd} : |x_1| = 1 \}, \\
B_l = \{ x_n \in (\mathbb{Z}^+ \cup \{0\})^{gd} : |x_n| = h_l \}, 1 \leq l \leq p,
\]
can be used to redefine $\{X_n\}_{1 \leq n \leq \tau}$ as $\{X_l\}_{1 \leq l \leq p}$.
2.4 Basics of Monte Carlo

The models reviewed so far can be difficult to work with analytically. In challenging scenarios, performing Bayesian inference via (2.2.1), (2.2.2), or (2.3.2) may only be feasible if one resorts to sampling from these posteriors. We review some numerical techniques in the remainder of this chapter, and we focus on advancements in the sequential Monte Carlo and Markov chain Monte Carlo literature. It is important to keep in mind that the utilisation of many of these algorithms is not limited to the models discussed above.

Starting with the basics of MC, consider the random variables $X_{1:n} \in E \subseteq \mathbb{R}^{d_{x_{1:n}}}$ for $n \geq 1$ with joint density $\pi_\theta(x_{1:n})$ depending on the fixed parameter $\theta$. Assume $\pi_\theta(x_{1:n})$ is only known point wise up to a multiplicative constant; it is not possible (or at least desirable) to evaluate the normalising constant $Z_\theta$, $1:n = \int_E \gamma_\theta(x_{1:n}) \, dx_{1:n}$. Therefore, for some function $h : \mathbb{R}^{d_{x_{1:n}}} \to \mathbb{R}$, the expectation $\mu = \mathbb{E}_{\pi_\theta}[h(x_{1:n})] = \int_E h(x_{1:n}) \pi_\theta(x_{1:n}) \, dx_{1:n}$ (2.4.1) is not analytically available. However, suppose it is possible to independently sample $X_{1:n}^i \sim \pi_\theta(\cdot)$ for $i \in \{1, \ldots, N\}$. The strong law of large numbers states that, as $N \to \infty$, $\hat{\mu}_N \xrightarrow{a.s.} \mu$, where $\hat{\mu}_N = \frac{1}{N} \sum_{i=1}^N h(x_{1:n}^i)$. Thus, one can estimate $\mu$ numerically by sampling from $\pi_\theta$ and computing $\hat{\mu}_N$. This process defines a basic Monte Carlo scheme.

The simplicity of basic MC allows for easy calculation of standard results related to the behaviour of $\hat{\mu}_N$. The unbiased estimator $\hat{\mu}_N$ is a random variable itself, and its variance is

$$\nabla_{\pi_\theta} \hat{\mu}_N = \frac{1}{N} \left[ \mathbb{E}_{\pi_\theta}[h^2(x_{1:n})] - \mu^2 \right] = \frac{1}{N} \left[ \int_E h^2(x_{1:n}) \pi_\theta(x_{1:n}) \, dx_{1:n} - \mu^2 \right].$$

This expression tells us that the variance of the estimator is $O\left( N^{-1} \right)$. As $N \to \infty$, the probability distribution of $\hat{\mu}_N$ will converge to a limiting distribution, and the central limit theorem states that $\sqrt{N}(\hat{\mu}_N - \mu) \Rightarrow \mathcal{N}(0, \sigma^2)$, where $\sigma^2 = \nabla_{\pi_\theta}[h(x_{1:n})]$ (assuming $0 < \sigma^2 < \infty$).
Finally, the form of $\hat{\mu}_N$ motivates an unbiased approximation of $\pi_{\theta}(x_{1:n})$:

$$\hat{\pi}_{\theta}(dx_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X^i_{1:n}}(dx_{1:n}).$$

It is easy to compute the variance of this estimator and show that as $N$ increases, its variance will decrease.

2.5 Importance Sampling

Suppose that sampling directly from $\pi_{\theta}(x_{1:n})$ is difficult, or even impossible. As an alternative to the basic MC scheme, one may resort to importance sampling (IS). In IS, one identifies an importance density $q_{\theta}(x_{1:n})$ with which it is easier to draw samples (also, it is required that $q(x_{1:n}) = 0 \implies \pi(x_{1:n}) = 0$). Rather than obtaining $\{x^i_{1:N}\}$ from $\pi_{\theta}(x_{1:n})$, the samples are drawn using $q_{\theta}(x_{1:n})$ and each $x^i_{1:n} \in \{x^i_{1:n}\}$ is assigned an un-normalised weight:

$$W_n^i(x^i_{1:n}) = \frac{\gamma_{\theta}(x^i_{1:n})}{q_{\theta}(x^i_{1:n})}. \quad (2.5.1)$$

The motivation behind defining the weights as such is that each weight is an unbiased estimate of the normalising constant:

$$\mathbb{E}_{q_{\theta}}[W_n(x_{1:n})] = \int_E W_n(x_{1:n}) q_{\theta}(x_{1:n}) dx_{1:n} = \int_E \gamma_{\theta}(x_{1:n}) dx_{1:n} = Z_{\theta,1:n}.$$ 

Averaging the $N$ weights produces an unbiased estimate whose non-asymptotic variance decreases as $N$ increases: the relative variance of the estimate

$$\hat{Z}_{\theta,1:n} = \frac{1}{N} \sum_{i=1}^{N} W^i_n(x^i_{1:n})$$

is

$$\frac{\nabla_{q_{\theta}}[\hat{Z}_{\theta,1:n}]}{Z^2_{\theta,1:n}} = \frac{1}{N} \left[ \mathbb{E}_{q_{\theta}}[W^2_n(x_{1:n})] - 1 \right] = \frac{1}{N} \left[ \int_E \gamma_{\theta}(x_{1:n}) dx_{1:n} - 1 \right]. \quad (2.5.2)$$
2.5 Importance Sampling

A drawback to IS is that the approximation of (2.4.1),

\[ \hat{\mu}_N = \sum_{i=1}^{N} w^i_n h(x^i_{1:n}) \]

where

\[ w^i_n (x^i_{1:n}) = \frac{W^i_n (x^i_{1:n})}{\sum_{i=1}^{N} W^i_n (x^i_{1:n})} \propto W^i_n (x^i_{1:n}) , \]

is biased for finite \( N \). However, by using the delta method and Taylor series expansions, one can derive the asymptotic bias and show that \( \hat{\mu}_N \) is consistent [29]:

\[
\lim_{N \to \infty} N (\hat{\mu}_N - \mu) = - \int_E \frac{\pi^2_\theta (x_{1:n})}{q_\theta (x_{1:n})} (h(x_{1:n}) - \mu) dx_{1:n} .
\]

The estimate also satisfies a central limit theorem with asymptotic variance as follows [29]:

\[
\frac{1}{N} \int_E \frac{\pi^2_\theta (x_{1:n})}{q_\theta (x_{1:n})} (h(x_{1:n}) - \mu)^2 dx_{1:n} .
\]

Once again, the form of \( \hat{\mu}_N \) motives an approximation of the density \( \pi_\theta (x_{1:n}) \):

\[ \hat{\pi}_\theta (dx_{1:n}) = \sum_{i=1}^{N} w^i_n \delta_{X^i_{1:n}} (dx_{1:n}) . \]

This expression is biased, but it will become more precise as \( N \) increases.

2.5.1 Random weight importance sampling

We note here that using random unbiased estimates of (2.5.1) will still yield a valid importance sampling algorithm ([42],[49],[84]). To illustrate this point, consider the following simple example. Suppose that we have a target \( \pi_\theta (x_{1:n}) \) and a proposal \( q_\theta (x_{1:n}) \). For the random variables \( X_{1:n} \in E \) and \( U \in F \), assume we have a function \( \psi_\theta (x_{1:n}, u) \) which takes values in \( \mathbb{R}^+ \). Assume also that \( \mathbb{E}_\gamma [\psi_\theta (x_{1:n}, u)] = W_{\gamma} (x_{1:n}) \), where the expectation
is taken with respect to some density $\gamma$. For some $\pi\theta$—integrable function $h(x_{1:n})$, we have:

$$E_{\pi\theta}[h(x_{1:n})] = \int_E h(x_{1:n}) W_n(x_{1:n}) q_{\theta}(x_{1:n}) dx_{1:n}$$

$$= \int_E h(x_{1:n}) \left[ \int_F \psi_{\theta}(x_{1:n}, u) \gamma(u) du \right] q_{\theta}(x_{1:n}) dx_{1:n}.$$ 

Thus, one could perform importance sampling by sampling from the extended proposal $q_{\theta} \times \gamma$.

### 2.6 Sequential importance sampling

Allow $n$ to now be a discrete (time) index which increases in increments of one. Consider the scenario where, at each time point $n$, we are required to sample from $\pi_{\theta}(x_{1:n})$. It would not be sensible to use IS to sample $X_{1:n}$ for each value of $n$ without making use of the samples $\{x_{1:n-1}^1, x_{1:n-1}^2, \ldots, x_{1:n-1}^N\}$ obtained at time $(n-1)$. It would be more efficient to retain the samples $\{x_{1:n-1}^1, x_{1:n-1}^2, \ldots, x_{1:n-1}^N\}$ and simply draw $X_n \mid x_{1:n-1}$ to build the union $X_{1:n}$. This more efficient scheme is called sequential importance sampling (SIS).

Rewriting the target $\pi_{\theta}(x_{1:n})$ as

$$\pi_{\theta}(x_{1:n}) \propto \gamma_{\theta}(x_{1:n}) = \frac{\gamma_{\theta}(x_{1:n})}{\gamma_{\theta}(x_{1:n-1})} \gamma_{\theta}(x_{1:n-1})$$

$$= \frac{\gamma_{\theta}(x_{1:n})}{\gamma_{\theta}(x_{1:n-1})} \cdots \frac{\gamma_{\theta}(x_{1:2})}{\gamma_{\theta}(x_1)} \gamma_{\theta}(x_1)$$

$$= \gamma_{\theta}(x_1) \prod_{k=2}^n \gamma_{\theta}(x_k \mid x_{1:k-1}),$$

we can construct an importance density as

$$q_{\theta}(x_{1:n}) = q_{\theta}(x_1) \prod_{k=2}^n q_{\theta}(x_k \mid x_{1:k-1}).$$

Essentially, at each time point of the SIS algorithm, a target $\pi_{\theta}(x_n \mid x_{1:n-1})$ is approximated with weighted samples obtained from $q_{\theta}(x_n \mid x_{1:n-1})$. The specific steps of SIS are
2.6 Sequential importance sampling

detailed in Algorithm 2, where

\[ W_n^i = \frac{\gamma_\theta(x_n^i \mid x_{1:n-1}^i)}{q_\theta(x_n^i \mid x_{1:n-1}^i)}. \]  

(2.6.1)

Similar to IS, one can use the weights (2.6.1) obtained at any iteration \( n \) of SIS to calculate an approximation of a target density via

\[ \hat{\pi}_\theta(dx_{1:n}) = \sum_{i=1}^{N} \omega_n^i \delta_{x_{1:n}^i}(dx_{1:n}), \]

where

\[ \omega_n^i(x_{1:n}^i) = \frac{\prod_{k=1}^{n} W_k^i(x_{1:k}^i)}{\sum_{i=1}^{N} \prod_{k=1}^{n} W_k^i(x_{1:k}^i)}. \]

An unbiased estimate of that target’s normalising constant is

\[ \hat{Z}_{\theta,1:n} = \frac{1}{N} \sum_{i=1}^{N} \left[ \prod_{k=1}^{n} W_k^i(x_{1:k}^i) \right]. \]  

(2.6.2)

As SIS is just a special case of standard IS, the relative variance of (2.6.2) will follow (2.5.2). Also, according to [29], SIS yields the unbiased estimate

\[ \frac{\hat{Z}_{\theta,1:n}}{Z_{\theta,1:n-1}} = \sum_{i=1}^{N} \omega_{n-1}^i W_n^i, \]  

(2.6.3)

which can be used to calculate \( \hat{Z}_{\theta,1:n} \) and is originally motivated by the expression

\[
\int_E W_n(x_n) \pi\theta (x_{1:n-1}) q_\theta (x_n \mid x_{1:n-1}) \, dx_{1:n} = \int_E \frac{\gamma_\theta(x_{1:n})}{\pi_\theta(x_{1:n-1})} \pi_\theta (x_{1:n}) \, dx_{1:n} = \int_E \frac{\gamma_\theta(x_{1:n})}{Z_{\theta,1:n-1}} \, dx_{1:n} = \frac{Z_{\theta,1:n}}{Z_{\theta,1:n-1}}.
\]

As SIS is just a special case of standard IS, the variance of SIS estimators behave similarly to their IS counterparts. Thus, the variance of SIS estimators will, in the worst case, increase exponentially with increasing values of \( n \) ([29],[60]). For example, note that the variance of (2.6.2) follows (2.5.2), which includes the term \( \pi^2_\theta(x_{1:n}) \). Also, the variance
Algorithm 2 Sequential importance sampling (SIS) algorithm

- **Step 1:** For \(i \in \{1, \ldots, N\}\), sample \(X^i_1 \sim q_\theta(\cdot)\) and compute the un-normalised weight:

\[
W^i_1 = \frac{\gamma_\theta(x^i_1)}{q_\theta(x^i_1)}.
\]

Set \(n = 2\).

- **Step 2:** For \(i \in \{1, \ldots, N\}\), sample \(X^i_n \mid x^i_1:n-1 \sim q_\theta(\cdot \mid x^i_1:n-1)\) and compute the un-normalised weight, \(W^i_n\). Set \(n = n + 1\) and return to the start of Step 2.

The variance of a sampled path \(i\)'s weight is

\[
\mathbb{V}_{q_\theta} \left[ \prod_{k=1}^{n} W^i_k \right] = \int E \gamma^2_\theta(x_{1:n}) \, dx_{1:n} - Z^2_{\theta,1:n},
\]

which includes \(\gamma^2_\theta(x_{1:n})\). It turns out that as the variance of the weights increases, the simulations suffer from weight degeneracy (i.e., all but one of the sampled paths will have a normalised importance weight that is effectively equal to zero). A resampling technique can be introduced to the SIS algorithm to combat weight degeneracy, thereby turning SIS into sequential Monte Carlo.

### 2.7 Basics of sequential Monte Carlo

In an SMC algorithm, one performs the same sampling and weighting steps as in SIS, but after computing the weights (2.6.1) at time \(n\), the samples \(\{x^i_{1:n}\}\) are resampled proportional to their normalised weights (see Algorithm 3). This resampling step has the effect of removing those samples with a smaller weight and multiplying those with a larger weight. Also, by resetting the normalised weights to equal \(1/N\) at each time step \(n\), we avoid multiplying \(n\) incremental weights together to determine a potential for each sample. Resetting the system as such removes the weight degeneracy problem. It also has a positive effect on \(\hat{Z}_{\theta,1:n}/Z_{\theta,1:n}\), whose asymptotic variance does not include exponential terms that depend
on $n$:

$$
\frac{1}{N} \left[ \int_{E} \pi_{\theta}^{2}(x_{1}) \, dx_{1} - 1 + \sum_{k=2}^{n} \int_{E} \pi_{\theta}(x_{1:k}) \pi_{\theta}(x_{k} \mid x_{1:k-1}) q_{\theta}(x_{k} \mid x_{1:k-1}) \, dx_{k-1:k} - 1 \right],
$$

as $N \to \infty$ [29].

After a resampling step is applied at an iteration of Algorithm 3, the output can be used to approximate a density $\pi_{\theta}(x_{1:n})$ via

$$
\hat{\pi}_{\theta}(x_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} I_{X_{1:n}^{i}}(x_{1:n}).
$$

The estimate (2.6.2) is not unbiased when using Algorithm 3, but (2.6.3) is still unbiased. Thus, by [23, page 112], an unbiased estimate of the normalising constant is

$$
\hat{Z}_{\theta,1:n} = \prod_{k=1}^{n} \left[ \frac{1}{N} \sum_{i=1}^{N} W_{k}^{i} I_{X_{1:k}^{i}}(X_{1:k}) \right]. \tag{2.7.1}
$$

We also mention that sometimes the order of the sampling and re-sampling steps can be reversed so that more unique samples are present at time $n$.

### 2.7.1 A note on resampling

There are several ways of performing the resampling step in an SMC algorithm (e.g., see Algorithms 4 and 5, which also appear in [29]). For an algorithm to constitute valid resampling, it must adhere to the following condition: if $N_{i}^{n}$ is the number of times that $x_{1:n}^{i}$ is resampled, then $\mathbb{E}[N_{i}^{n} \mid w_{i}^{n}] = N w_{i}^{n}$. In other words, we require that the resampled particles be distributed according to some distribution which is an unbiased approximation of the distribution related to $\hat{\pi}_{\theta}(x_{1:n})$.

In practice, resampling at every iteration of Algorithm 3 may not be desirable. A drawback to resampling is that, as particles with high importance weights are selected more frequently than those with low importance weights, it is likely that there will be fewer than $N$ unique paths after a resampling step. After many successive resampling steps through a time $n$, there may be very few unique paths through time $(n-k)$. This loss of path diversity
Algorithm 3 Generic sequential Monte Carlo (SMC) algorithm

- Step 1: For \( i \in \{1, \ldots, N\} \), sample \( X^i_1 \sim q_\theta (\cdot) \) and compute the un-normalised weight:

\[
W^i_1 = \frac{\gamma_\theta (x^i_1)}{q_\theta (x^i_1)}.
\]

For \( i \in \{1, \ldots, N\} \), sample \( A^i_1 \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j^{th} \) probability \( w^j_1 \propto W^j_1 \). The sample \( \{a^i_1 : N\} \) are the indices of the resampled particles. Set all normalised weights equal to \( 1/N \), and set \( n = 2 \).

- Step 2: For \( i \in \{1, \ldots, N\} \), sample \( X^{i|a_i(1:n-1)}_n \sim q_\theta (\cdot | x^{a(i)}_{1:n-1}) \) and compute the un-normalised weight:

\[
W^n_i = \frac{\gamma_\theta (x^{i|a_i(1:n-1)}_n)}{q_\theta (x^{i|a_i(1:n-1)}_n)}.
\]

For \( i \in \{1, \ldots, N\} \), sample \( A^{i|a_i(1:n)}_{1:n} \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j^{th} \) probability \( w^n_j \propto W^n_j \). Set all normalised weights equal to \( 1/N \), and set \( n = n+1 \). Return to the start of Step 2.

Algorithm 4 Multinomial resampling

At time step \( n \) of an SMC algorithm, and for \( i \in \{1, \ldots, N\} \),

- Step 1: Sample the random variable \( U \sim U (0, 1) \). Set \( g = 1 \).

- Step 2: If \( \sum_{k=1}^{g-1} w^n_k \leq U \leq \sum_{k=1}^{g} w^n_k \), then set \( a^n_i = g \) (thereby sampling the entire set of indices \( A^{i|a_i(1:n)}_{1:n} \)). Otherwise, set \( g = g + 1 \) and return to the beginning of Step 2.

Algorithm 5 Systematic resampling

At time step \( n \) of an SMC algorithm,

- Step 1: Sample the random variable \( U_1 \sim U (0, \frac{1}{N}) \). For \( i \in \{2, \ldots, N\} \), set \( U_i = U_{i-1} + \frac{1}{N} \). Set \( g = 1 \).

- Step 2: For \( i \in \{1, \ldots, N\} \), if \( \sum_{k=1}^{g-1} w^n_k \leq U_i \leq \sum_{k=1}^{g} w^n_k \), then set \( a^n_i = g \) (thereby sampling the entire set of indices \( A^{i|a_i(1:n)}_{1:n} \)). Otherwise, set \( g = g + 1 \).
is called path degeneracy. Heuristic solutions are available to mitigate the effects of path degeneracy. For example, one could choose to resample only when the effective sample size ([60],[64]) falls below a certain threshold. In such an adaptive SMC scheme, effective sample size (ESS) is defined as

\[
\text{ESS} = \frac{N}{1 + \nabla_q^\theta \left[ \prod_{k=1}^N W_k \right]} \approx \frac{1}{\sum_{i=1}^N (w_i^q)^2}.
\]

Note that every SMC algorithm implemented in this thesis and most of the algorithms discussed in this literature review either implicitly or explicitly make use of Algorithm 4.

### 2.7.2 Block-sampling within sequential Monte Carlo

Path degeneracy issues can also be mitigated by further modifying Algorithm 3, such as in the block-sampling approach of [27]. Consider propagating the samples \( \{x_{1:n}^{1:N}\} \) at time \( n \), where the samples \( \{x_{n-B+1:n}^{1:N}\} \) for \( B > 1 \) are sampled again and the originals are discarded. This scheme requires extending the target \( \pi^\theta (x_{1:n}) \) for \( n > B \). Thus, denoting \( \bar{x}_n^i \) as a new sample at time \( n \), the new extended target at time \( n > B \) will be

\[
\pi^\theta (x_{1:n-1}, x_{n-B+1:n}^i) = \pi^\theta (x_{1:n-B}, x_{n-B+1:n}) \lambda^\theta (x_{n-B+1:n-1} \mid x_{1:n-B}, x_{n-B+1:n})
\]

with some constructed density \( \lambda^\theta \) and associated proposal density

\[
q^\theta (x_{1:n-1}, x_{n-B+1:n}^i) = \pi^\theta (x_{1:n-1}) q^\theta (x_{n-B+1:n} \mid x_{1:n-1})
\]

(see Algorithm 6). It can be shown that larger values for \( B \) typically reduce the variance of the importance weights (2.7.2) and reduce path degeneracy ([29], [27]).

### 2.7.3 High dimensionality

While a strong effort has been devoted to reducing the variance of SMC algorithms as \( n \) in the sequence \( X_{1:n} \) increases, less work has focused on the dimensionality of each individual \( X_k \). Standard SMC algorithms’ errors grow as the dimension of each \( X_k \) increases, and so
Algorithm 6 SMC with fixed lag block-sampling

- **Step 1:** For $i \in \{1, \ldots, N\}$, sample $X^i_1 \sim q_\theta(\cdot)$ and compute the un-normalised weight:

$$W^i_1 = \frac{\gamma_\theta(x^i_1)}{q_\theta(x^i_1)}.$$ 

For $i \in \{1, \ldots, N\}$, sample $A^i_1 \in \{1, \ldots, N\}$ from a discrete distribution on \{1, \ldots, N\} with $j^{th}$ probability $w^i_j \propto W^i_j$. The sample $\{a^i_1:N\}$ are the indices of the resampled particles. Set all normalised weights equal to $1/N$, and set $n = 2$.

- **Step 2:** If $n \geq B$, then go to Step 3. Otherwise, for $i \in \{1, \ldots, N\}$, sample $X^{a(i)}_1:n-1 \sim q_\theta(\cdot \mid x^{a(i)}_1:n-1)$ and compute the un-normalised weight:

$$W^i_n = \frac{\gamma_\theta(x^{a(i)}_1:n-1 \mid \pi^i_1:n)}{\gamma_\theta(x^{a(i)}_1:n-1) q_\theta(x^{a(i)}_1:n \mid \pi^i_1:n)}.$$ 

Set $\{x^{1:N}_1:n \} = \{\pi^{1:N}_1:n \}$. For $i \in \{1, \ldots, N\}$, sample $A^{i}_1:n \in \{1, \ldots, N\}$ from a discrete distribution on \{1, \ldots, N\} with $j^{th}$ probability $w^i_j \propto W^i_j$. Set all normalised weights equal to $1/N$, and set $n = n + 1$. Return to the start of Step 2.

- **Step 3:** For $i \in \{1, \ldots, N\}$, sample $X^{a(i)}_{n-B+1:n} \mid x^{a(i)}_{1:n-1} \sim q_\theta(\cdot \mid x^{a(i)}_{1:n-1})$ and compute the un-normalised weight:

$$W^i_n = \frac{\gamma_\theta(x^{a(i)}_{1:n-B} \mid \pi^{i}_{n-B+1:n}) \lambda_\theta(x^{a(i)}_{n-B+1:n-1} \mid x^{a(i)}_{1:n-B}, \pi^{i}_{n-B+1:n})}{\gamma_\theta(x^{a(i)}_{1:n-1}) q_\theta(\pi^{i}_{n-B+1:n} \mid x^{a(i)}_{1:n-1})}. \tag{2.7.2}$$

Set $\{x^{1:N}_{n-B+1:n} \} = \{\pi^{1:N}_{n-B+1:n} \}$. For $i \in \{1, \ldots, N\}$, sample $A^{i}_{1:n} \in \{1, \ldots, N\}$ from a discrete distribution on \{1, \ldots, N\} with $j^{th}$ probability $w^i_j \propto W^i_j$. Set all normalised weights equal to $1/N$, and set $n = n + 1$. Return to the start of Step 3.
as the size of $X_k$ increases, prohibitively higher numbers of samples must be drawn to keep errors at an acceptable level. The problems related to dimensionality collectively form an open area of research. Work has been done recently to get a better understanding of the issue and begin moving forward towards a solution ([6],[7],[80]).

### 2.8 Sequential Monte Carlo filtering techniques for hidden Markov models

Up to this point, the SMC discussion has focused on general algorithms where the target at time $n$ takes the form $\pi_\theta (x_{1:n})$. We now switch to the specific case where $\pi_\theta (x_{1:n})$ is the joint density (2.2.1) of an HMM. The next three sections review how SMC can be used to sample from the filtering and smoothing densities introduced in Sections 2.2.1 and 2.2.2.

Recall that in the filtering problem, one has the recursion

$$
\pi_\theta (x_n \mid y_{1:n}) \propto g_\theta (y_n \mid x_n) \int f_\theta (x_n \mid x_{n-1}) \pi_\theta (x_{n-1} \mid y_{1:n-1}) \, dx_{n-1}.
$$

(2.8.1)

If $\pi_\theta (x_n \mid y_{1:n})$ is not analytically available, then one may use (2.8.1) to build samples from it recursively (i.e., if samples from $\pi_\theta (x_{n-1} \mid y_{1:n-1})$ are obtained at time $(n-1)$, then the expression (2.8.1) gives a framework for propagating those samples into draws from $\pi_\theta (x_n \mid y_{1:n})$). Any of the numerical algorithms discussed so far can be used to sample from this recursion. However, the SIS and SMC algorithms are particularly attractive because they offer online solutions; an online algorithm is one that does not increase in complexity as $n$ increases and only requires a fixed allocation of memory as it iterates. To illustrate these ideas, we re-write Algorithm 3 in terms of the HMM filtering problem (see Algorithm 7).

Algorithm 7 is sometimes called a particle filter, where the term “particle” refers to a sample $x_{1:n}^i$. Throughout the literature, different choices of proposal density $q_\theta$ have led to the development of different particle filters. The simplest choice of $q_\theta = f_\theta$ yields the bootstrap particle filter of [43]. The choice of $q_\theta \left( x_n^i \mid x_{n-1}^{a(i)} \right) = \pi_\theta \left( x_n^i \mid y_n, x_{n-1}^{a(i)} \right)$ minimises the conditional variance of the importance weights ([28, Proposition 2], [94]). Coupling the optimal importance density with a slight modification of the resampling procedure yields
Chapter 2. Literature Review

Algorithm 7 SMC filtering algorithm for HMMs

- Step 1: For \( i \in \{1, \ldots, N\} \), sample \( X^i_1 \sim q_\theta(\cdot) \) and compute the un-normalised weight:

\[
W^i_1 = \frac{\mu_\theta(x^i_1) g_\theta(y_1 | x^i_1)}{q_\theta(x^i_1)}.
\]

For \( i \in \{1, \ldots, N\} \), sample \( A^i_1 \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j^{th} \) probability \( w^j_1 \propto W^j_1 \). The sample \( \{a^i_1 N\} \) are the indices of the resampled particles. Set all normalised weights equal to \( 1/N \), and set \( n = 2 \).

- Step 2: For \( i \in \{1, \ldots, N\} \), sample \( X^i_n | x^{a(i)}_{n-1} \sim q_\theta(\cdot | x^{a(i)}_{n-1}) \) and compute the un-normalised weight:

\[
W^i_n = \frac{f_\theta(x^i_n | x^{a(i)}_{n-1}) g_\theta(y_n | x^i_n)}{q_\theta(x^i_n | x^{a(i)}_{n-1})}.
\]

(2.8.2)

For \( i \in \{1, \ldots, N\} \), sample \( A^i_1:n \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j^{th} \) probability \( w^j_n \propto W^j_n \). Set all normalised weights equal to \( 1/N \), and set \( n = n + 1 \). Return to the start of Step 2.

the auxiliary particle filter (APF) of [77]; see also [14], [78], and [53] for extensions and improvements over the original APF that show reduced variance.

The APF of [14] and [53] is reprinted here as Algorithm 8. It assumes that one has access to “optimal” proposals which are not ordinarily available; these proposals should be approximated when they are not analytically available. We note that the incremental weights are equal to one, but [53] show that the asymptotic variances of estimates obtained via Algorithm 8 are not always less than those of estimates obtained via Algorithm 7.

2.8.1 Comparison to Kalman filters

As mentioned in Section 2.2.1, the extended and unscented Kalman filters can be used to obtain biased approximations of the hidden state of non-linear Gaussian HMMs. SMC offers a more flexible solution to HMM filtering, as it is not limited to Gaussian models. The SMC algorithms are also easier to implement in high dimensional problems. Especially in the case of the extended Kalman filter, the deterministic solutions require tedious
Algorithm 8 Auxiliary particle filter for HMMs

- Step 1: For $i \in \{1, \ldots, N\}$, sample $X^i_1 \mid y_1 \sim q_\theta (\cdot \mid y_1)$ and compute the un-normalised weight:

$$W^i_1 = \frac{\mu_\theta (x^i_1) g_\theta (y_1 \mid x^i_1)}{q_\theta (x^i_1 \mid y_1)}.$$  

For $i \in \{1, \ldots, N\}$, sample $A^i_1 \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $j$th probability $w^i_1 \propto \pi_\theta (y_2 \mid x^i_1) W^i_1$. The sample $\{a^1_1; N\}$ are the indices of the resampled particles. Set all normalised weights equal to $1/N$, and set $n = 2$.

- Step 2: For $i \in \{1, \ldots, N\}$, sample $X^n_i \mid x^{a(i)}_{n-1}, y_n \sim q_\theta (\cdot \mid x^{a(i)}_{n-1}, y_n)$ and compute the un-normalised weight:

$$W^n_i = \frac{f_\theta (x^n_i \mid x^{a(i)}_{n-1}) g_\theta (y_n \mid x^n_i)}{\pi_\theta (y_n \mid x^{a(i)}_{n-1}) q_\theta (x^n_i \mid x^{a(i)}_{n-1}, y_n)} = 1.$$  

For $i \in \{1, \ldots, N\}$, sample $A^i_{1:n} \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $j$th probability $w^n_j \propto \pi_\theta (y_{n+1} \mid x^n_j) W^n_j$. Set all normalised weights equal to $1/N$, and set $n = n + 1$. Return to the start of Step 2.
calculations in the high dimensional cases.

As \( n \) grows to be large, the Kalman filters encounter additional problems. The estimated covariances will continue to increase as the filters iterate, and this issue becomes a significant problem when one has limited memory for storing a covariance matrix obtained via computer simulation. The ensemble Kalman filter is another extension of Algorithm 1 that solves this problem by using Monte Carlo techniques [31]. This filter samples the state of the system at time \( n \) and calculates a sample covariance of the draws (also called ensembles). The sample covariance is then used in place of the deterministically estimated covariance. The ensemble Kalman filter assumes that the HMM is Gaussian, and so SMC still offers a class of more general solutions.

2.9 Sequential Monte Carlo smoothing techniques for hidden Markov models

Running a particle filter (e.g., Algorithm 7) through time \( T \) and marginalising the output to obtain \( \hat{\pi}_\theta (x_t \mid y_{1:T}) \) is a valid smoothing procedure. However, as a result of path degeneracy (see Section 2.7.1), there will be fewer unique samples of \( x_t \) as \( t \) becomes smaller relative to \( T \). In fact, when \( t \ll T \), it is possible that \( x_t \) will be approximated by a single unique particle.

To mitigate the path degeneracy issue, SMC approximations to the FFBS (2.2.6) and the two-filter formula (2.2.7) have been developed, and we review those methods in the following two subsections.

2.9.1 Approximation of FFBS

[59] originally proposed using a particle filter to obtain the sequence of filtering approximations \( \{ \hat{\pi}_\theta (x_n \mid y_{1:n}) \}_{n \in \{1, \ldots, T\}} \), with corresponding normalised weights \( \{ w_{1:T}^{1:N} \} \), to compute
2.9 Sequential Monte Carlo smoothing techniques for hidden Markov models

an approximation to (2.2.6):

\[ \hat{\pi}_\theta (dx_n \mid y_{1:T}) = \sum_{i=1}^{N} w_{n+1}^{i} \delta_{X_{n+1}^{i}} (dx_n) = \sum_{i=1}^{N} w_{n}^{i} \left[ \sum_{j=1}^{N} w_{n+1}^{j} f_{\theta} (x_{n+1}^{j} \mid x_{n}^{i}) \right] \delta_{X_{n}^{i}} (dx_n), \]

(2.9.1)

where \( \hat{\pi}_\theta (dx_{n+1} \mid y_{1:T}) = \sum_{i=1}^{N} w_{n+1}^{i} \delta_{X_{n+1}^{i}} (dx_{n+1}) \). At each time step \( n \) of the particle filter, the approximation \( \hat{\pi}_\theta (x_n \mid y_{1:n}) \) and the weights \( w_{n}^{1:N} \) are saved before the resampling step is carried out. Thus, there are many unique elements used to compute (2.9.1), leading to a diverse set of samples from \( \pi_\theta (x_n \mid y_{1:T}) \). However, note that computing (2.9.1) is an \( \mathcal{O} (N^2) \) procedure, and calculating \( \hat{\pi}_\theta (x_1 \mid y_{1:T}) \) requires a forward and a backward pass across the time index \( n \). There are two modifications of this procedure that simplify it by either reducing its complexity or eliminating its need for a backward pass.

To bypass the \( \mathcal{O} (N^2) \) complexity, [26] consider a Markov chain \( \{J_n\}_{t \leq n \leq T} \) which moves backward in time where the value \( j_{n+1} \) corresponds to the index of the smoothed particle \( x_{n+1}^{j} \) sampled from \( \hat{\pi} (x_{n+1} \mid y_{1:T}) \). In other words, \( w_{n+1}^{j} \) is the probability that \( J_{n+1} = j_{n+1} \) to \( J_n = i_n \). This idea can be exploited to formalise the \( \mathcal{O} (N) \) forward filtering backward simulation (FFBSi) algorithm (see Algorithm 9), and the estimator (2.9.1) is replaced with

\[ \hat{\pi}_\theta (dx_t \mid y_{1:T}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_{t}^{i}} (dx_t). \]

for \( t \in \{1, \ldots, T-1\} \).

In the specific application where one is computing a smoothed additive functional of the form

\[ S_{\theta,T} = \mathbb{E}_{\pi_\theta (x_{1:T} \mid y_{1:T})} [S_T (X_{1:T}) \mid y_{1:T}] \]

(2.9.2)
Chapter 2. Literature Review

Algorithm 9 Forward filtering backward simulation (FFBSi) algorithm

- Step 0: Run Algorithm 7 (or a suitable SMC filtering alternative) to target the sequence \( \{\pi(x_{1:n} \mid y_{1:n})\}_{n \in \{1, \ldots, T\}} \), and save the normalised incremental weights \( \{w_{1:T}^i\} \).

- Step 1: For \( i \in \{1, \ldots, N\} \), sample \( J_T^i \propto w_T^i \). Set \( n = T - 1 \).

- Step 2: If \( n = t - 1 \), stop. Otherwise, for \( i \in \{1, \ldots, N\} \), sample

  \[
  J_n^i \propto w_n^i f_{\theta} \left( x_{n+1} \mid x_n^i \right)
  \]

  and set \( n = n - 1 \). Return to the start of Step 2.

where

\[
S_T(x_{1:T}) = s_1(x_1) + \sum_{n=2}^T s_n(x_{n-1}, x_n)
\]

and each \( s_n \) is a real-valued function, one could eliminate the backward pass of FFBS to yield an online SMC smoothing algorithm [25]. One notable example of (2.9.2) would be the score for an HMM:

\[
\nabla \log p_{\theta}(y_{1:T}) = \sum_{n=1}^T \mathbb{E}_{\pi_{\theta}(x_{1:T} \mid y_{1:T})} \left[ \nabla \log f_{\theta}(X_n \mid x_{n-1}) \right] + \sum_{n=1}^T \mathbb{E}_{\pi_{\theta}(x_{1:T} \mid y_{1:T})} \left[ \nabla \log g_{\theta}(y_n \mid x_n) \right].
\]

To name just a few of its applications, the score can be used to calculate Jeffreys prior (a non-informative prior that is invariant under re-parametrisations of \( \theta \)), and it is used in the classical Fisher scoring algorithm for finding the maximum likelihood estimate of \( \theta \). To eliminate the backward pass of FFBS, [25, Proposition 2.1] defines the recursion

\[
T_{\theta,n}(x_n) = \int S_n(x_{1:n}) \pi_{\theta}(x_{1:n-1} \mid y_{1:n-1}, x_n) \, dx_{1:n-1}
\]

\[
= \int \left[ T_{\theta,n-1}(x_{n-1}) + s_n(x_{n-1}, x_n) \right] \pi_{\theta}(x_{n-1} \mid y_{1:n-1}, x_n) \, dx_{n-1},
\]

where \( T_{\theta,1}(x_1) = 0 \) and \( S_{\theta,n} = \int T_{\theta,n}(x_n) \pi_{\theta}(x_n \mid y_{1:n}) \, dx_n \). The SMC approximation
to this recursion (as given in Algorithm 10) yields an $O(N^2)$ version of FFBS that only requires a forward-in-time pass.

### 2.9.2 Generalised two-filter smoothing

Before we review SMC approximations to the two-filter formula (2.2.7), notice that the term $\pi_\theta(y_{t:T} | x_t)$ is not necessarily a probability density in the argument $x_t$ in the general case. Monte Carlo methods are only valid when their targets have finite integrals. As there is no guarantee that $\int \pi_\theta(y_{t:T} | x_t) \, dx_t < \infty$, we cannot use SMC to target $\pi_\theta(y_{t:T} | x_t)$. [10] propose a slight alteration to the two-filter formula that, using pseudo-priors, removes the term $\pi_\theta(y_{t:T} | x_t)$ and replaces it with a term that can be targeted using SMC. The way this is achieved is as follows. First, consider a sequence of pseudo-priors $\{\xi_{n,\theta}\}_{n \in \{1, \ldots, T\}}$ such that if $\pi_\theta(y_{n:T} | x_n) > 0$, then $\xi_{n,\theta} > 0$. [10, Proposition 1] defines a sequence of
artificial probability densities

\[
\tilde{\pi}_\theta (x_{n:T} \mid y_{n:T}) = \frac{\xi_{n,\theta} (x_n) g_\theta (y_n \mid x_n) \left[ \prod_{k=n+1}^{T} g_\theta (y_k \mid x_k) f_\theta (x_k \mid x_{k-1}) \right]}{\int \xi_{n,\theta} (x_n) g_\theta (y_n \mid x_n) \left[ \prod_{k=n+1}^{T} g_\theta (y_k \mid x_k) f_\theta (x_k \mid x_{k-1}) \right] \, dx_{n:T}} \tag{2.9.3}
\]

for \( n \in \{1, \ldots, T - 1\} \) and

\[
\tilde{\pi}_\theta (x_T \mid y_T) = \frac{\xi_{T,\theta} (x_T) g_\theta (y_T \mid x_T) \left[ \prod_{k=T}^{T} g_\theta (y_k \mid x_k) f_\theta (x_k \mid x_{k-1}) \right]}{Z_{\theta,n:T}} \tag{2.9.4}
\]

such that

\[
\pi_\theta (y_{n:T} \mid x_n) = \tilde{Z}_{\theta,n:T} \frac{\tilde{\pi}_\theta (x_n \mid y_{n:T})}{\xi_{n,\theta} (x_n)} \quad \forall n \in \{1, \ldots, T\}. \tag{2.9.5}
\]

Combining (2.2.7) and (2.9.5) yields the generalised two-filter smoothing formula,

\[
\pi_\theta (x_t \mid y_{1:T}) \propto \frac{\pi_\theta (x_t \mid y_{1:t-1}) \tilde{\pi}_\theta (x_t \mid y_{t:T})}{\xi_{t,\theta} (x_t)}. \tag{2.9.6}
\]

Now, the term \( \pi_\theta (y_{1:T} \mid x_t) \) is replaced with \( \tilde{\pi}_\theta (x_t \mid y_{1:T}) \), which can be sampled from using SMC as follows.

We have already discussed how to use SMC to approximate \( \tilde{\pi}_\theta (x_t \mid y_{1:t-1}) \). Approximation of \( \tilde{\pi}_\theta (x_t \mid y_{1:T}) \) relies on the backward-in-time SMC filter (see Algorithm 11). Algorithm 11 uses the same principles as Algorithm 7, but it targets the sequence of densities (2.9.3),(2.9.4) and propagates from time \( T \) to time \( t < T \). It can also be made more efficient in the same ways that Algorithm 7 can be optimised \[10\] (e.g. \( q_\theta (x^i_n \mid x_{n+1}^{a(i)}) = \pi_\theta (x^i_n \mid y_n, x_{n+1}^{a(i)}) \) minimises the variance of the importance weights (2.9.7), and a backward version of Algorithm 8 is straightforward). With a forward filter and a backward filter defined, the generalised two-filter SMC smoother follows (see Algorithm 12).

The third step of Algorithm 12 is \( \mathcal{O}(N^2) \) in complexity, which can be undesirable when one needs to calculate \( \tilde{\pi}_\theta (x_t \mid y_{1:T}) \) many times. [36] develop an \( \mathcal{O}(N) \) version of
2.9 Sequential Monte Carlo smoothing techniques for hidden Markov models

Algorithm 11 Backward SMC filtering algorithm for HMMs

- Step 1: For \( i \in \{1, \ldots, N\} \), sample \( X^i_T \sim q_\theta (\cdot) \) and compute the un-normalised weight:

\[
W^i_T = \frac{\xi_{T,\theta} (x^i_T) g_\theta (y_T | x^i_T)}{q_\theta (x^i_T)}.
\]

For \( i \in \{1, \ldots, N\} \), sample \( A^i_T \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j \)th probability \( w^j_T \propto W^j_T \). The sample \( \{a^i_{1:N}\} \) are the indices of the resampled particles. Set all normalised weights equal to \( 1/N \), and set \( n = T - 1 \).

- Step 2: If \( n = t - 1 \), stop. Otherwise, for \( i \in \{1, \ldots, N\} \), sample \( X^i_n \mid x^{a(i)}_{n+1} \sim q_\theta (\cdot \mid x^{a(i)}_{n+1}) \) and compute the un-normalised weight:

\[
W^i_n = \frac{\xi_{n,\theta} (x^i_n) f_\theta (x^{a(i)}_{n+1} \mid x^i_n) g_\theta (y_n \mid x^i_n)}{\xi_{n+1,\theta} (x^{a(i)}_{n+1}) q_\theta (x^i_n \mid x^{a(i)}_{n+1})}.
\]

(2.9.7)

For \( i \in \{1, \ldots, N\} \), sample \( A^i_{n:T} \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j \)th probability \( w^j_n \propto W^j_n \). Set all normalised weights equal to \( 1/N \), and set \( n = n - 1 \). Return to the start of Step 2.

Algorithm 12 \( O(N^2) \) generalised two-filter SMC smoother for HMMs

- Step 1: Run Algorithm 7 (or a suitable SMC forward filtering alternative) to target \( \pi (x^i_{1:t-1} \mid y_{1:t-1}) \), and save the normalised incremental weights \( \{w^i_{1:N}\} \).

- Step 2: Run Algorithm 11 (or a suitable SMC backward filtering alternative) to target \( \tilde{\pi} (x_T \mid y_{t:T}) \), and save the normalised incremental weights \( \{\tilde{w}^i_{1:N}\} \).

- Step 3: Calculate

\[
\hat{\pi}_\theta (dx_t \mid y_{1:T}) \propto \sum_{i=1}^{N} \tilde{w}^i_t \left[ \sum_{j=1}^{N} \frac{\tilde{w}^j_{t-1} f_\theta (x_t \mid \tilde{x}^{j}_{t-1})}{\xi_{t,\theta} (x_t)} \right] \delta_{x^i_t} (dx_t).
\]
Algorithm 13 $O(N)$ generalised two-filter SMC smoother for HMMs

- Step 1: Run Algorithm 7 (or a suitable SMC forward filtering alternative) to target $\pi (\mathcal{X}_{1:t-1} | y_{1:t-1})$, where the final resampling step samples $\mathcal{A}^{1}_{t-1} \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $k$th probability $\beta_{t-1}^k \propto \pi_\theta (y_{t-1} | \mathcal{X}_{t-1}^k) \tilde{W}^k_{t-1} \approx \pi_\theta (y_t | \mathcal{X}_{t-1}^k) \tilde{W}^k_{t-1}$. Assign the indices of the resampled forward particles to be the values of the vector $\mathcal{X}_{1:t-1}$.

- Step 2: Run Algorithm 11 (or a suitable SMC backward filtering alternative) to target $\tilde{\pi} (\mathcal{X}_{t+1:T} | y_{t+1:T})$, where the final resampling step samples $\mathcal{A}^{1}_{t+1:T} \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $k$th probability $\beta_{t+1}^k \propto \pi_\theta (y_{t+1} | \mathcal{X}_{t+1}^k) \tilde{W}^k_{t+1} \approx \pi_\theta (y_t | \mathcal{X}_{t+1}^k) \tilde{W}^k_{t+1}$. Assign the indices of the resampled backward particles to be the values of the vector $\mathcal{X}_{1:t+1}$.

- Step 3: For $l \in \{1, \ldots, N\}$, sample $\mathcal{X}_l^i \mid \mathcal{X}_{t-1}^l, \mathcal{X}_{t+1}^l \sim q_\theta (\cdot | \mathcal{X}_{t-1}^l, \mathcal{X}_{t+1}^l)$ and compute the un-normalised incremental weight:

$$W_l^i \propto f_\theta (x^i_l \mid \mathcal{X}_{t-1}^l) g_\theta (y_t \mid x^i_l) f_\theta (\mathcal{X}^j_{t+1} \mid x^i_l) \beta_{t-1}^j \beta_{t+1}^j \xi_{t+1, \theta} (\mathcal{X}^j_{t+1}) q_\theta (x^j_{t+1} \mid \mathcal{X}^j_{t-1}, \mathcal{X}^j_{t+1})$$

the generalised two-filter SMC smoother by re-writing (2.9.6) as

$$\pi_\theta (x_t \mid y_{1:T}) = \int \pi_\theta (x_{t-1} \mid y_{1:t-1}) f_\theta (x_t \mid x_{t-1}) \mathcal{W}_{t-1} \tilde{\pi}_\theta (x_t \mid y_{1:T}) \xi_{t, \theta} (x_t) \, dx_{t-1} \tilde{\pi}_\theta (x_t | y_{1:T})$$

$$= \int \pi_\theta (x_{t-1} \mid y_{1:t-1}) f_\theta (x_t \mid x_{t-1}) f_\theta (x_{t+1} \mid x_t) \tilde{\pi}_\theta (x_{t+1} \mid y_{1:T+1}) \xi_{t+1, \theta} (x_{t+1}) \, dx_{t-1}, x_{t+1}.$$  

(2.9.8)

The target (2.9.8) can be sampled from using the algorithm of [36], which is reproduced here as Algorithm 13.

2.10 HMM inference using approximate Bayesian computation and SMC

In the filtering and smoothing algorithms just discussed, it is necessary to calculate the likelihood density $g_\theta$. The likelihood density can actually be quite difficult to evaluate for complex models. As practitioners are developing ever more intricate ways of modelling
real world phenomena, likelihood-free inference methods, such as approximate Bayesian computation [87], constitute an active area of research. In a basic approximate Bayesian computation (ABC) algorithm, one samples an unknown parameter from its prior distribution and then, based on the value of that sample, draws from the likelihood distribution. When a sample from the likelihood is sufficiently close to the actual observed data (with the samples and observed data usually being compared via summary statistics; see [34] for an advanced example), then the corresponding parameter value is accepted as a sample from a biased approximation of its posterior. In this scheme, the likelihood density is never actually calculated, although it is assumed that one can draw from the likelihood distribution.

SMC and ABC have been combined throughout the literature; see [51], [52], and [67] for examples of algorithms that can perform filtering and smoothing for HMMs. We focus here on two particular combinations of SMC and ABC: the particle filter of [52] and the alive particle filter of [51].

The authors of [52] use the second model of Section 2.2.3 to obtain a biased approximation of an SMC algorithm targeting an HMM with transitions \( f_\theta \) and likelihoods \( g_\theta \) when \( g_\theta \) is either impossible or undesirable to compute (assuming it is still possible to simulate from the likelihood distribution). Basically, the particle filter of [52] commences by simulating \( k_n \sim f_\theta (\cdot \mid k_{n-1}) \), simulating \( u_n \sim g_\theta (\cdot \mid k_n) \), and then finally considering \( k_n \) to be a draw from the latent process of the HMM only if \( u_n \in B_{n,\epsilon} (y_n) \) (see Algorithm 14).

As some incremental weights will be intentionally set to equal zero, it is possible that Algorithm 14 will die out in practice (i.e., all chains after a sampling step at time \( n \) might have a weight of zero). The alive particle filter of [51] is a special version of Algorithm 14 that cannot die out. At time \( n \), [51] propose obtaining the samples \( \{ x_1^{1:T_n} \} \), where

\[
T_n = \inf \left\{ h \geq N : \sum_{i=1}^{h} W_i^N \geq N \right\}.
\]  

(2.10.1)

In other words, sampling continues until \( N \) samples of non-zero weight are obtained (see Algorithm 15, which has a random running time). The alive particle filter has an upper bound on its error that does not depend on \( n \) [51, Theorem 3.1], and its associated unbiased
Algorithm 14 ABC filtering algorithm for HMMs

- Step 1: For $i \in \{1, \ldots, N\}$, sample $X_1^i \sim \mu(\cdot)$ and compute the un-normalised weight:
  $$W_1^i = \mathbb{I}_{R \times B_1, \epsilon(y_1)}(x_1^i).$$
  For $i \in \{1, \ldots, N\}$, sample $A_1^i \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $j$th probability $w_1^j \propto W_1^j$. The sample $\{a_1^i, \ldots, a_N^i\}$ are the indices of the resampled particles. Set $n = 2$.

- Step 2: For $i \in \{1, \ldots, N\}$, sample $X_n^i | k_{n-1}^{a(i)} \sim M_\theta(\cdot | x_{n-1}^{a(i)})$ and compute the un-normalised weight:
  $$W_n^i = \mathbb{I}_{R \times B_n, \epsilon(y_n)}(x_n^i).$$
  For $i \in \{1, \ldots, N\}$, sample $A_1^{i:n} \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $j$th probability $w_n^j \propto W_n^j$. Set $n = n + 1$. Return to the start of Step 2.

An estimate of the normalising constant is given by [51, Proposition 3.1]:

$$\hat{Z}_{\theta,1:n} = \left[ \prod_{k=1}^{n-1} \frac{N - 1}{T_k - 1} \right] \frac{1}{T_n - 1} \sum_{i=1}^{T_n-1} W_n^i = \frac{n}{\prod_{k=1}^{n-1} T_k - 1}$$

(2.10.2)

Note that in proving these results, the authors of [51] make use of a nuance in their algorithm: the last sampled particle is deleted at every time step.

The applications of the particle filters of [51] and [52] are not actually limited to inference on HMMs. Both algorithms can be used in any case where one wishes to assign some forms of indicator potentials to be the incremental weights of samples of a Markov chain $X_{1:n}$ obtained via the transitions $M_\theta$. The HMM case was simply presented first for ease of exposition. A general version of the alive particle filter is given as Algorithm 16, where we assume that one aims for each sample of $X_n$ to be arbitrarily close to some value $b_n$.

Finally, we note that the conditional density of the stopping time $T_n = t_n$ and the particles $x_{n:n}^{1:t_n}$ generated by either Algorithm 15 or Algorithm 16 at time $n$ is

$$\pi(x_{n:n}^{1:t_n}, t_n | \mathcal{F}_{n-1}) = \left( \frac{t_n}{N - 1} \right) \left( \prod_{i=1}^{t_n} \frac{1}{W_{n-1}^{a(i)} x_{n-1}^{a(i)}} M_\theta(x_n^i | x_{n-1}^{a(i)}) \right),$$

(2.10.3)
Algorithm 15 Alive particle filter for HMMs

- Step 1: Set \( h = 1 \) and \( S = 0 \).
- Step 2: Sample \( X_1^h \sim \mu(\cdot) \) and compute the un-normalised weight:
  \[
  W_1^h = \mathbb{I}_{B_1}(y_1)(x_1^h) .
  \]
  Compute \( S = \sum_{i=1}^{h} W_i^1 \). If \( S < N \), then set \( h = h + 1 \) and return to the beginning of Step 2. Otherwise, set \( T_1 = h \) and \( n = 2 \).
- Step 3: Set \( h = 1 \) and \( S = 0 \).
- Step 4: Sample \( A^h_{1:n-1} \in \{1, \ldots, T_{n-1} - 1\} \) from a discrete distribution on \( \{1, \ldots, T_{n-1} - 1\} \) with \( j \)-th probability \( w^j_{n-1} \propto W^j_{n-1} \). Sample \( X_n^h | a^{(h)}_{n-1} \sim M_\theta \left( \cdot | x^{a^{(h)}_{n-1}}_{n-1} \right) \) and compute the un-normalised weight:
  \[
  W_n^h = \mathbb{I}_{B_n}(y_n)(x_n^h) .
  \]
  Compute \( S = \sum_{i=1}^{h} W_i^n \). If \( S < N \), then set \( h = h + 1 \) and return to the beginning of Step 4. Otherwise, set \( T_n = h \) and \( n = n + 1 \) and return to the start of Step 3.

Algorithm 16 General alive particle filter

- Step 1: Set \( h = 1 \) and \( S = 0 \).
- Step 2: Sample \( X_1^h \sim \mu(\cdot) \) and compute the un-normalised weight:
  \[
  W_1^h = \mathbb{I}_{B_1}(b_1)(x_1^h) .
  \]
  Compute \( S = \sum_{i=1}^{h} W_i^1 \). If \( S < N \), then set \( h = h + 1 \) and return to the beginning of Step 2. Otherwise, set \( T_1 = h \) and \( n = 2 \).
- Step 3: Set \( h = 1 \) and \( S = 0 \).
- Step 4: Sample \( A^h_{1:n-1} \in \{1, \ldots, T_{n-1} - 1\} \) from a discrete distribution on \( \{1, \ldots, T_{n-1} - 1\} \) with \( j \)-th probability \( w^j_{n-1} \propto W^j_{n-1} \). Sample \( X_n^h | a^{(h)}_{n-1} \sim M_\theta \left( \cdot | x^{a^{(h)}_{n-1}}_{n-1} \right) \) and compute the un-normalised weight:
  \[
  W_n^h = \mathbb{I}_{B_n}(b_n)(x_n^h) .
  \]
  Compute \( S = \sum_{i=1}^{h} W_i^n \). If \( S < N \), then set \( h = h + 1 \) and return to the beginning of Step 4. Otherwise, set \( T_n = h \) and \( n = n + 1 \) and return to the start of Step 3.
where $\mathcal{F}_{n-1}$ is the filtration generated by the particle system through time $(n - 1)$ and we require that $\sum_{i=1}^{n} W_n^i (x_n^i) = N$ and $W_n^i (x_n^i) = 1$. This is a result which is used in the original work appearing in Chapter 4.

### 2.11 Sequential Monte Carlo techniques for stopped Markov processes

There is also a type of SMC algorithm particularly suited for sampling from the stopped Markov process of Section 2.3. Recall that the stopped Markov process $\{X_n\}_{1 \leq n \leq \tau}$ propagates from $B_0$ to $A$, and one can introduce the $\mathcal{F}$-sets $B_1, \ldots, B_p$ (where $B_p = A$) to interpolate between $B_0$ and $A$ and redefine $\{X_n\}_{1 \leq n \leq \tau}$ as $\{X_l\}_{1 \leq l \leq p}$. To introduce an SMC solution for sampling from the stopped Markov process $\{X_n\}_{1 \leq n \leq \tau}$, we first review another sampling technique called multi-level splitting (see [41] for an example).

Multi-level splitting (MS) uses the construction $\{X_l\}_{1 \leq l \leq p}$ to obtain samples of $\{X_n\}_{1 \leq n \leq \tau}$. The algorithm is initialised with $s_0$ particles in $B_0$. Using some appropriate importance distribution, a trajectory is simulated for each particle until it either lands in $B_1$ or it lands in some absorbing set $K$ ($K$ is defined by the practitioner, as are the interpolating sets). The particles that land in $K$ are removed from the simulation. For each of the particles that lands in $B_1$, $s_1$ independent replicas are created and their trajectories are propagated until $B_2$ or $K$ is reached. The process is iterated until all trajectories lead to $A$ or $K$. Thus, MS uses the $\mathcal{F}$-sets to focus computational effort on simulating paths that actually reach the target set. MS can be rather difficult to implement efficiently, though, when $A$ is a set consisting only of a few rare events. Hitting a rare target set $A$ from an arbitrary starting point in $B_0$ may require a careful calibration of the importance distribution and the factors $\{s_j\}_{0 \leq j \leq p-1}$.

[17] and [23, Section 12.2.6] introduce an SMC resampling procedure inspired by MS called stopped time resampling (STR). Their work is essentially equivalent to re-engineering generic SMC so that one may propagate $N$ parallel trajectories of $X_{1:p}$ with the true intention of targeting (2.3.3). In SMC with STR (henceforth called multi-level SMC), samples of $X_{1:\tau}$ are generated via SIS. As the simulations propagate, a sample $X_{1:\tau}^i$ is frozen when it reaches a set $B_l$. After all samples have reached the set $B_l$, a resampling step is performed and SIS commences towards $B_{l+1}$ (see Algorithm 17). As $Z_\theta \equiv Z_\theta (re-
call Section 2.3), the multi-level SMC approximation of the normalising constant (2.3.4) is

\[
\hat{Z}_\theta = \hat{Z}_\theta = \prod_{n=1}^{p} \left[ \frac{1}{N} \sum_{l=1}^{N} W_n^l \right].
\]  

(2.11.1)

Similar to MS, Algorithm 17 focuses computational effort on simulating those paths that will reach the target set most quickly. Notice by the weight expression (2.11.2) that a path which reaches a set \( B_l \) more quickly will have a larger incremental weight and will have a greater chance of being resampled. Thus, Algorithm 17 prunes away those paths which require too many steps to reach \( A \).

### 2.11.1 Backward importance sampling for stopped Markov processes

Suppose \( A \) is a target set that consists only of a few rare events and \( B_0 \) is a set that consists of many common events. As presented so far, Algorithm 17 would not necessarily be better suited than MS for hitting a rare target set \( A \) from \( B_0 \). It is more sensible to follow the methods of [44] and adopt IS with time reversed. We can initialise Algorithm 17 in the rare event set and then propagate towards the set of common events. More specifically, we can initialise the sampling scheme with the data \( y \) and introduce an importance density \( q_\theta (x_{n-1} | x_n) \) through which we sample \( \{X_n\}_{1 \leq n \leq \tau} \) backward in time. This procedure ensures that the data is hit when the process is considered forward in time.

To properly structure backward SIS for stopped Markov processes, the definitions of the initial set \( B_0 \) and the terminal set \( A \) must be swapped (i.e., \( B_0 \) now contains the data \( y \)). We denote the reverse of the event sequence \( \{X_n\}_{1 \leq n \leq \tau} \) as \( \{X_j\}_{1 \leq j \leq \tau} \), where we are replacing the forward time index \( n \) with the reverse time index \( j \). The un-normalised likelihood (2.3.3) is re-written

\[
\gamma_\theta (x_{1:\tau}, \tau, y) = \mathbb{I}_{\{a \in (A \times (A^c)^{\tau-1})\}} (x_{1:\tau}) \pi_\theta (y | x_{1:\tau}) \left[ \prod_{j=1}^{\tau} f_\theta (x_{j-1} | x_{j}) \right],
\]  

(2.11.3)

and the normalising constant (2.3.4) is unchanged. A proposal density for the full path
Algorithm 17 Multi-level SMC

- Step 0: For \( i \in \{1, \ldots, N\} \), initialise \( X^i_0 \in B_0 \).
- Step 1: For \( i \in \{1, \ldots, N\} \), start from \( n = 1 \) and sample \( X^i_n \mid x^i_{n-1} \sim q_\theta (\cdot \mid x^i_{n-1}) \) until \( x^i_n \in B_1 \) (i.e., increase \( n \) and iterate until \( x^i_n \in B_1 \)). If \( x^i_n \in B_1 \), set \( \tau_i^1 = n \) and \( \mathcal{X}_1^i = (x_{1,\tau_i^1}, \tau_i^1) \), and compute the un-normalized weight:

\[
W_1^i = \frac{\tau_1 \left( \mathcal{X}_1^i \right)}{\tau_1 \left( \mathcal{X}_1^i \right)},
\]

where \( \tau_1 \left( \mathcal{X}_1^i \right) = \prod_{n=1}^{\tau_i^1} q_\theta (x^i_n \mid x^i_{n-1}) \). Once \( x^i_{\tau_i^1} \in B_1 \) for all \( i \), then for \( i \in \{1, \ldots, N\} \), sample \( A_1^i \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j \)th probability \( w_1^j \propto W_1^j \). The sample \( \{a_1^1, N\} \) are the indices of the resampled particles. Set all normalised weights equal to \( 1/N \), and set \( l = 2 \).

- Step 2: If \( l = p + 1 \), stop. Otherwise, for \( i \in \{1, \ldots, N\} \), start from \( n = \tau_{l-1}^{a(i)} + 1 \) and sample \( X^i_n \mid x^i_{n-1} \sim q_\theta (\cdot \mid x^i_{n-1}) \). Continue to increase \( n \) and sample from the appropriate importance distribution until \( x^i_n \in B_l \). If \( x^i_n \in B_l \), set \( \tau_i^l = n \), \( \mathcal{X}_l^i = (x_{l,\tau_i^l}, \tau_i^l) \), and \( \mathcal{X}_{l-1}^i = (\mathcal{X}_{l-1}^{a(i)} \mid \mathcal{X}_{l-1}^{a(i)}) \). Compute the un-normalized weight:

\[
W_l^i = \frac{\tau_l \left( \mathcal{X}_{l-1}^i \right)}{\tau_{l-1} \left( \mathcal{X}_{l-1}^{a(i)} \right) \tau_l \left( \mathcal{X}_{l-1}^i \right) \tau_l \left( \mathcal{X}_{l-1}^{a(i)} \right)},
\]

where \( \tau_0 \left( \mathcal{X}_1^i \right) = \prod_{n=1}^{\tau_i^1} q_\theta (x^i_n \mid x^i_{n-1}) \). Once \( x^i_{\tau_i^l} \in B_l \) for all \( i \), then for \( i \in \{1, \ldots, N\} \), sample \( A_l^i \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j \)th probability \( w_l^j \propto W_l^j \). Set all normalised weights equal to \( 1/N \), and set \( l = l + 1 \). Return to the start of Step 2.
2.11 Sequential Monte Carlo techniques for stopped Markov processes

starting from the dataset \( y \) can now be written as

\[
q_\theta (x_{1:\tau}) = \mathbb{I}_{\{a \in (B_0 \cap y)\}} (x_1) \prod_{j=1}^\tau q_\theta (x_j \mid x_{j-1}) \prod_{a \in A} (x_\tau),
\]

which means the likelihood (2.11.3) can again be re-written as

\[
\gamma_\theta (x_{1:\tau}, \tau, y) = \pi_\theta (y \mid x_{1:\tau}) \prod_{j=1}^\tau \left( f_\theta (x_{j-1} \mid x_j) \prod_{a \in (B_0 \cap B_\tau)} \right) q_\theta (x_{1:\tau}).
\]

with

\[
Z_{\theta,1:\tau} = \sum_{\tau \in \mathbb{Z}} \int_{B^{\tau}} \pi_\theta (y \mid x_{1:\tau}) \prod_{j=1}^\tau \left( f_\theta (x_{j-1} \mid x_j) \prod_{a \in (B_0 \cap B_\tau)} \right) q_\theta (x_{1:\tau}) \, dx_{1:\tau}.
\]

Redefining a backward-in-time version of \( \{X_l\}_{1 \leq l \leq p} \) from \( \{X_j\}_{1 \leq j \leq \tau} \) requires that we appropriately modify all interpolating sets \( B_1, \ldots, B_{\tau-1} \) in such a way that the relationship \( 0 \leq T_1 \leq \cdots \leq T_p \) still holds. We also modify (2.3.8) as

\[
\overline{\pi}_\theta (X_1) = \pi_\theta (y \mid x_{1:\tau}) \prod_{j=1}^{\tau_1} f_\theta (x_{j-1} \mid x_j) \prod_{a \in (B_0 \cap B_1)} (x_{1:\tau_1}) \prod_{a \in B_1} (x_{\tau_1}),
\]

\[
\overline{\pi}_\theta (X_{1:n}) = \overline{\pi}_\theta (X_{1:n-1}) \prod_{j=\tau_{n-1}+1}^{\tau_n} f_\theta (x_{j-1} \mid x_j) \prod_{a \in (B_{n-1} \cap B_0)} (x_{\tau_{n-1}+1:\tau_n}) \prod_{a \in B_n} (x_{\tau_n})
\]

for \( n = 2, \ldots, p \) (note that we are now denoting \( \{X_n\}_{1 \leq n \leq p} \) as the backward version of \( \{X_l\}_{1 \leq l \leq p} \)). Due to the relationship \( \overline{\pi}_\theta \equiv \gamma_\theta \), we still have an identity for moving between the sequences \( \{X_j\}_{1 \leq j \leq \tau} \) and \( \{X_n\}_{1 \leq n \leq p} \).

Going forward, the backward SIS scheme will take the place of the forward sampling scheme in Algorithm 17.

Backward importance sampling for coalescent models

The above framework is somewhat abstract, and so to be clear, we illustrate some concepts on the coalescent models of Section 2.3.1. Essentially, the backward SIS procedure would begin in \( B_0 \) with the sample of size \( m \) and propagate until the MRCA is reached in set \( A \).
The likelihoods would be
\[
\gamma_\theta(x_{1:\tau}, \tau, y) = \mathbb{I}_{\{a \in (A \times (A^c)^{\tau-1})\}} (x_{1:\tau}) \frac{m - 1}{m - 1 + \mu} \cdot \frac{\prod_{i=1}^{\tau} y_i!}{m!} \left[ \prod_{j=1}^{\tau} f_\theta(x_{j-1}|x_j) \right]
\]
for the coalescent with mutation and
\[
\gamma_\theta(x_{1:\tau}, \tau, y) = \mathbb{I}_{\{a \in (A \times (A^c)^{\tau-1})\}} (x_{1:\tau}) \frac{m - 1}{m - 1 + \mu} \cdot \frac{\prod_{i=1}^{\tau} y_i!}{m!} \left[ \prod_{j=1}^{\tau} f_\theta(x_{j-1}|x_j) \right]
\]
for the coalescent with mutation and migration. Note the \(\frac{m-1}{m-1+\mu}\) terms. When simulating the coalescent models forward in time, one would normally propagate until a sample of size \((m + 1)\) is obtained and then delete the last additional individual. This process ensures that, once all \(m\) individuals are present, all possible mutations and migrations at that point are considered before the evolution process is stopped. Finally, the interpolating sets could now be structured as
\[
B_0 = \{ x_1 \in (\mathbb{Z}^+ \cup \{0\})^d : x = y \}, \quad (2.11.6)
\]
\[
B_n = \{ x_j \in (\mathbb{Z}^+ \cup \{0\})^d : |x_j| = h_n \}, \quad 1 \leq n \leq p,
\]
for the coalescent with mutation and
\[
B_0 = \{ x_1 \in (\mathbb{Z}^+ \cup \{0\})^{gd} : x = y \}, \quad (2.11.7)
\]
\[
B_n = \{ x_j \in (\mathbb{Z}^+ \cup \{0\})^{gd} : |x_j| = h_n \}, \quad 1 \leq n \leq p,
\]
for the coalescent with mutation and migration, where \(m > h_1 > h_2 > \cdots > h_p = 1\) is a collection of integers.

### 2.12 Twisted particle filters

To complete a thorough review of the methodology, we close the discussion that is specific to SMC with an examination of two more major extensions.

A recent study in [91] has led to a better understanding of how introducing a change
2.12 Twisted particle filters

of measure on the particle system of a traditional SMC scheme might yield an ideal SMC
algorithm in the following sense:

\[
\frac{1}{n} \log \left( \frac{\mathbb{E}[\tilde{Z}_{\theta,1:n}^2]}{Z_{\theta,1:n}^2} \right) \xrightarrow{a.s.} 0 \quad \text{as} \quad n \to \infty,
\]

(2.12.1)

where the expectation is taken with respect to the joint density of the samples obtained by
the algorithm. One will notice that (2.12.1) implies that the variance of \( \tilde{Z}_{\theta,1:n} \) will be low,
which is important when one is using SMC to run particle marginal Metropolis-Hastings
algorithms (see Section 2.18 below).

To explain this notion in more detail, recall the generic SMC algorithm: Algorithm
3. For an \((E, \mathcal{E})\)-valued Markov process \( \{X_n\}_{n \geq 1} \), one can write the associated SMC
algorithm’s transition density \( M_\theta : E^N \times \mathcal{E}^\otimes N \to [0, 1] \) as

\[
M_\theta (x_{k-1}, x_k) = \prod_{i=1}^{N} \Phi_\theta (\eta^N) (x_{k}^i),
\]

\[
\Phi_\theta (\eta^N) (x_{k}^i) = \frac{1}{N} \sum_{j=1}^{N} W_{k-1}^j (x_{k-1}^j) q_\theta (x_k^i | x_{k-1}^j),
\]

\[
\eta^N (W) = \frac{1}{N} \sum_{l=1}^{N} W^l (x_l^i).
\]

The authors of [91] define the additive, non-negative functional

\[
h(x^{1:N}) = \frac{1}{N} \sum_{i=1}^{N} h(x^i),
\]

where each \( h : \mathbb{R}^{dx} \to \mathbb{R}^+ \) is a particular eigenfunction such that, for the operator

\[
Q_\theta (\varphi) (x_k) = \int Q_\theta (u | x_k) \varphi (u) du
\]

\[
Q_\theta (x_{k+1} | x_k) \propto q_\theta (x_{k+1} | x_k) W_k (x_k),
\]

one has

\[
Q_\theta (h) = \lambda h
\]
for $\lambda \in \mathbb{R}^+$; the eigenfunctions and their eigenvalues are defined in [91, Proposition 2], and they are also stated later in Chapter 4. By using the additive functional $h$ to change the measure of the entire particle system (i.e., by replacing $M_\theta(x_{k-1}, x_k)$ with $\tilde{M}_\theta(x_{k-1}, x_k) \propto M_\theta(x_{k-1}, x_k) h(x_k)$), one obtains the ideal SMC algorithm of [91] that achieves (2.12.1). This algorithm uses a new estimate of the normalising constant,

$$
\hat{Z}_{\theta,1:n} = \prod_{k=1}^{n} \left[ \frac{1}{N} \sum_{i=1}^{N} Q_{\theta}(h)(x_{k-1}^i) \right]^{1/N} dM_\theta(x_{k-1}, \cdot) \left( x_k \right),
$$

which is clearly unbiased when the expectation is taken with respect to $\tilde{M}_\theta$. As re-weighting Markov transitions using eigenfunctions is typically referred to as “twisting”, the authors call their algorithm the twisted particle filter (see Algorithm 18).

It is worth mentioning that there are difficulties in using this strategy. Deriving the optimal $h$ and $Q_{\theta}(h)(x_k)$ is impossible (or at least extremely difficult) in most cases, and so they usually need to be approximated. However, as the numerical examples in [91] show (and as Sections 4.4 and 4.6 of Chapter 4 of this thesis will also show), even good approximations of the optimal $h$ and $Q_{\theta}(h)(x_k)$ can lead to lower variance estimates of $\hat{Z}_{\theta,1:n}$.

### 2.13 Sequential Monte Carlo samplers

Finally, it is important to note that the SMC methodology is not limited to approximating target densities that are defined on state spaces of increasing dimension. Allow $m$ to be a discrete time index which increases in increments of one, and consider the scenario where, at each time point $m$, we are required to target the time-inhomogeneous density $\pi_{m, \theta}(x_m)$. For all values of $m$, each $\pi_{m, \theta}$ is defined on some common measurable space $(E, \mathcal{E})$ (e.g., one could have $\pi_{m, \theta}(x_m) = \pi_{m, \theta}(x \mid y_{1:m})$ where $y_{1:m}$ are realisations of some random variable $Y$). The SMC samplers of [24] are designed specifically for sampling from the sequence $\{\pi_{m, \theta}\}_{m \geq 1}$. [24] use backward Markov kernels of the form $L_{k-1}(x_{k-1} \mid x_k)$ to create a new sequence of targets $\{\tilde{\pi}_{m, \theta}\}_{m \geq 1}$ whose state spaces do increase in dimension.
Algorithm 18 Twisted particle filter

- **Step 0:** For \( i \in \{1, \ldots, N\} \), sample \( X^i_0 \) from some approximate initial distribution and set the un-normalised weight: \( W^i_0 = 1 \). Set \( n = 1 \).

- **Step 1:** Resampling steps:
  - Sample \( K \) from the discrete uniform distribution on \( \{1, \ldots, N\} \).
  - Sample \( A^k_{n-1} \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j^{th} \) probability proportional to \( Q_\theta(h)(x^j_{n-1}) \).
  - For \( i \in \{1, \ldots, N\} \) and \( i \neq k \), sample \( A^i_{n-1} \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j^{th} \) probability proportional to \( W^j_{n-1} \).

- **Step 2:** Sampling steps:
  - Sample \( X^k_n \mid x^{a(k)}_{n-1} \propto q_\theta \left( \cdot \mid x^{a(k)}_{n-1} \right) h(\cdot) \).
  - For \( i \in \{1, \ldots, N\} \) and \( i \neq k \), sample \( X^i_n \mid x^{a(i)}_{n-1} \sim q_\theta \left( \cdot \mid x^{a(i)}_{n-1} \right) \).

- **Step 3:** For \( i \in \{1, \ldots, N\} \), compute the un-normalised weight:

\[
W^i_n = \frac{\gamma_\theta \left( x^i_n \mid x^{a(i)}_{n-1} \right)}{q_\theta \left( x^i_n \mid x^{a(i)}_{n-1} \right)}.
\]

Set \( n = n + 1 \) and return to the start of Step 1.
with increasing values of $m$. Each $\tilde{\pi}_{m,\theta}$ is defined as

$$
\tilde{\pi}_{m,\theta}(x_{1:m}) \propto \tilde{\gamma}_{m,\theta}(x_{1:m}) = \gamma_{m,\theta}(x_m) \prod_{k=2}^{m} L_{k-1}(x_{k-1} | x_k).
$$

Forward Markov kernels of the form $Q_m(x_m | x_{m-1})$ can be used in an SMC algorithm to sample from $\tilde{\pi}_{m,\theta}$, where the incremental weights take the form

$$
\tilde{W}_m = \frac{\tilde{\gamma}_{m,\theta}(x_{1:m})}{\tilde{\gamma}_{m-1,\theta}(x_{1:m-1})} \frac{Q_m(x_m | x_{m-1})}{Q_m(x_{m-1} | x_{m-1})}.
$$

The true targets of interest $\pi_{m,\theta}$ can clearly be marginalised from $\tilde{\pi}_{m,\theta}$ and standard estimates of the normalising constants apply.

### 2.14 Basics of Markov chain Monte Carlo

Suppose now that we would like to sample from

$$
\pi(\theta, x_{1:n}) = \pi(\theta) \pi_{\theta}(x_{1:n})
$$

for $n \geq 1$ and $\theta \sim \pi(\cdot)$ with $\theta \in \Theta \subseteq \mathbb{R}^d$ and $X_{1:n} \in E \subseteq \mathbb{R}^{d_1 \cdot n}$; one example is the case where we want to infer $\theta$ and the hidden process $X_{1:n}$ of an HMM using an analytically intractable density $\pi(\theta, x_{1:n} | y_{1:n})$. The MCMC methodology offers a class of algorithms that can be used to sample from densities of the form $\pi(\theta, x_{1:n})$. To match the style of the introduction to MC methods in Section 2.4, the following introduction to MCMC is based around the goal of computing an approximation to the expectation $\mathbb{E}_\pi[h(\theta, x_{1:n})]$, for some real-valued function $h$. However, please keep in mind that computing the first moment is not the primary interest of this thesis.

#### 2.14.1 Metropolis-Hastings algorithm

To introduce MCMC algorithms, we begin by following the discussion of [83]. We simplify the notation by setting $n = 1$ without loss of generality; thus, we set $X_{1:n} = X_1 = X$. We consider below a Markov chain with kernel $\kappa$ and measurable space $(W, \mathcal{W})$ with $W = \mathbb{R}^d$. 


2.14 Basics of Markov chain Monte Carlo

In an MCMC algorithm, one defines a Markov chain \((\theta_1, x_1), (\theta_2, x_2), \ldots\) such that, if the chain iterates long enough, each \((\theta_i, x_i)\) is eventually an approximate sample from the target of interest \(\pi(\theta, x)\). In other words, when \(i\) is greater than some positive constant \(B\), \((\theta_i, x_i)\) is approximately sampled from the target. The samples are not independent of one another (as they correspond to consecutive realisations of a Markov chain), but when

\[
1 \ll B \ll M \quad \text{and as} \quad M \to \infty, \quad \text{then}
\]

\[
\hat{\mu}_{(M-B)} = \frac{1}{M-B} \sum_{i=B+1}^{M} h(\theta_i, x_i) \to \mathbb{E}_\pi[h(\theta, x)] = \mu
\]

when the Markov chain is ergodic for the target \(\pi(\theta, x)\). Our Markov chain \(\{(\theta^i, x^i)\}_{i \geq 1}\) must satisfy three conditions to be ergodic for \(\pi\):

- the chain must be irreducible, meaning that there exists a probability measure \(\nu\) on \((W, \mathcal{W})\) such that \(\nu(Y > 0) \Rightarrow \int_W f(Y | u) \kappa(du | w) > 0\) for all \(w \in W\), all \(Y \in \mathcal{W}\), and \(f = \sum_{n=0}^\infty \kappa^n\);

- all states of the chain must be aperiodic (i.e., for each state \(w \in W\), the period \(d_w = \gcd\{n \geq 1 : \kappa^n(w | w) > 0\} = 1\), where \(\gcd\) denotes “greatest common divisor”); and

- if the state of the chain is ever distributed according to \(\pi\), then all future states will also always be distributed according to \(\pi\) (i.e., \(\pi\) is the unique stationary density of the chain, and \(\sum_{(\theta, x) \in W} \pi(\theta, x) \kappa(\zeta, y | \theta, x) = \pi(\zeta, y)\), with \((\zeta, y) \in W\)).

The third item above explains why a Markov chain \(\{(\theta^i, x^i)\}_{i \geq 1}\) will have to run for \(B\) iterations (called a burn in period) before one can sample from the target \(\pi\). The reason for the burn in period is that, in practice, a practitioner will likely not know if the chain has been initialised in its stationary distribution. The burn in period allows the chain to reach its steady state.

[45] and [68] developed the Metropolis-Hastings (M-H) algorithm, which runs an ergodic Markov chain \(\{(\theta^i, x^i)\}_{i \geq 1}\) to sample from any \(\pi(\theta, x)\). This general algorithm is formally presented here as Algorithm 19, where \(q\) is a density that proposes a move within
Chapter 2. Literature Review

Algorithm 19 Basic Metropolis-Hastings

- Step 0: Set $\theta^0$ and $x^0$ arbitrarily. Set $k = 1$.
- Step 1: Sample $\theta^*, X^* \mid \theta^{k-1}, x^{k-1} \sim q(\cdot \mid \theta^{k-1}, x^{k-1})$.
- Step 2: With acceptance probability

$$1 \wedge \frac{\pi (\theta^*, x^*) q (\theta^{k-1}, x^{k-1} \mid \theta^*, x^*)}{\pi (\theta^{k-1}, x^{k-1}) q (\theta^*, x^* \mid \theta^{k-1}, x^{k-1})}, \quad (2.14.2)$$

set $\theta^k = \theta^*$ and $x^k = x^*$. Otherwise, discard $\theta^*$ and $x^*$ and set $\theta^k = \theta^{k-1}$ and $x^k = x^{k-1}$. Set $k = k + 1$. Return to the start of Step 1.

the chain to $(\theta^i, x^i)$ from $(\theta^{i-1}, x^{i-1})$ and (2.14.2) is the probability of accepting the proposed move. In Algorithm 19, we could alternatively sample $\theta^*, X^* \sim q(\cdot)$ to obtain the independent M-H, or even $\theta^*, X^* \mid \theta, x \sim q(\cdot \mid \theta, x) = q(\theta, x \mid \theta^*, x^*)$ to obtain the symmetric M-H. In any event, the proposal $q$ must always be chosen in such a way that the Markov chain is irreducible. The acceptance probability (2.14.2) is chosen to ensure that the Markov chain is reversible with respect to $\pi$, which implies that $\pi$ is the stationary density. Finally, aperiodicity is ensured because the probability of eventually rejecting a move is greater than zero.

2.14.2 Gibbs sampler

When possible, it is advantageous to sample the random variables $(\theta, X)$ from their conditional densities given all other random variables in the target $\pi$. Suppose $(\zeta, z)$ has already been accepted as a sample from $\pi$. If we choose the proposal density $q(\theta, x \mid \zeta, z) = \pi (\theta \mid z) \pi (x \mid \theta)$, then the acceptance probability (2.14.2) becomes

$$1 \wedge \frac{\pi (\theta, x) \pi (\zeta \mid x) \pi (z \mid \zeta)}{\pi (\zeta, z) \pi (\theta \mid z) \pi (x \mid \theta)}, \quad (2.14.3)$$

The R.H.S. of (2.14.3) is always equal to one. Thus, the choice $q(\theta, x \mid \zeta, z) = \pi (\theta \mid z) \times \pi (x \mid \theta)$ results in an M-H algorithm that always accepts a proposed move from $(\zeta, z)$ to $(\theta, x)$. This particular algorithm is called the (fixed scan) Gibbs sampler ([38], [39]), and
Algorithm 20 Basic Gibbs sampler

- Step 0: Set $\theta^0$ and $x^0$ arbitrarily. Set $k = 1$.
- Step 1: Sample $\theta^k | \cdots \sim \pi (\cdot | x^{k-1})$.
- Step 2: Sample $X^k | \cdots \sim \pi (\cdot | \theta^k)$. Set $k = k + 1$. Return to the start of Step 1.

It is detailed in Algorithm 20. One could also implement a Gibbs sampler that randomly chooses which component of the state space to update at each iteration, but we do not consider that algorithm here.

2.14.3 Convergence of Markov chain Monte Carlo

Standard convergence results are available for MCMC algorithms that justify their use. As the Markov chain $\{(\theta^i, x^i)\}_{i \geq 1}$ is ergodic, the following holds true:

$$
\mathbb{P} \left( (\theta^M, X^M) \in A \right) = \int_A \pi (\theta, x) \, d\theta \, dx
$$

for $A \subseteq (\Theta \times E)$ as $M \to \infty$, where the chain starts from some arbitrary point $(\theta^1, x^1)$. This result implies that MCMC algorithms can effectively target (2.14.1) after the burn in period. Furthermore, we can be more specific about the type of ergodicity and the rate of convergence. Letting $\kappa^i (\theta, x | \zeta, z)$ be the marginal at time step $i$ conditioned on the starting point $(\zeta, z)$, the Markov chain $\{(\theta^i, x^i)\}_{i \geq 1}$ is said to be geometrically ergodic if the following holds:

$$
\sup_{\theta, x} |\kappa^i (\theta, x | \zeta, z) - \pi (\theta, x)| \leq C(\zeta, z) \rho^i
$$

with $\rho \in [0, 1)$ and $C$ is a non-negative real-valued function that is finite for any $(\zeta, z) \subseteq (\Theta \times E)$. By geometrically ergodic, we mean that the chain converges at a geometric rate to the equilibrium $\pi$. If there exists a finite constant $C \in \mathbb{R}^+$, which is independent of $(\zeta, z)$, such that

$$
\sup_{\theta, x} |\kappa^i (\theta, x | \zeta, z) - \pi (\theta, x)| \leq C \rho^i
$$
then the Markov chain is said to be uniformly ergodic.

We have already mentioned in Section 2.14.1 that \( \hat{\mu}_{(M-B)} \overset{a.s.}{\rightarrow} \mu \) as \( M \rightarrow \infty \). A central limit theorem for ergodic Markov chains with unique stationary distributions additionally states that

\[
\sqrt{(M - B)}(\hat{\mu}_{(M-B)} - \mu) \Rightarrow \mathcal{N}(0, \sigma^2),
\]

where

\[
\sigma^2 = \sum_{i=1}^{\infty} \text{Cov}_x[h(\theta^{B+1}, x^{B+1}), h(\theta^{B+1+i}, x^{B+1+i})] < \infty \ (2.14.4)
\]

for a bounded \( h \) (assuming \( 0 < \sigma^2 \)) [54]. The expression (2.14.4) tells us that as samples from the Markov chain move closer to being independent, the approximation \( \hat{\mu}_{(M-B)} \) becomes more precise.

A practitioner likely does not know the ideal values of \( B \) and \( M \). Thus, MCMC simulations must be monitored to determine when the equilibrium might have been reached and when estimators can start being calculated. There are several ways of tracking convergence, but in this thesis, we will focus on the following methods:

- In some numerical simulations, we will monitor trace plots and autocorrelation function (ACF) plots of the sampled random variables. Trace plots show the trajectory that a single random variable follows as it is sampled across iterations. These plots give a sense of how a random variable’s state space is being explored and whether or not an algorithm appears to be sampling from a random variable’s entire state space. The ACF plots show the autocorrelation between the samples of a single random variable across iterations. Due to the expression (2.14.4), we would like to see autocorrelation approach zero.

- In other numerical simulations, we will monitor convergence through the ideas of [11]. Essentially, [11] state that we should run a new MCMC algorithm multiple times, and if the variance measured within copies of the algorithm matches the variance measured across the iterations of all copies of the algorithm, then one does not have evidence that the MCMC scheme is not ergodic.
2.15 Block updating for Markov chain Monte Carlo

The applicability of the original MCMC algorithms is limited in the real world, and many extensions have been proposed in the literature to sample from more complex forms of the target $\pi$. For example, when $\pi$ has a large state space, Algorithm 19 requires many iterations to fully explore the full state space. Consider the target $\pi(\theta, x_{1:n})$ for $n \gg 1$. Algorithm 19 may find difficulty in proposing a high dimensional sample that also has a high probability of being accepted. If one wishes to obtain samples as quickly as possible, then it is necessary to design more efficient MCMC algorithms that can quickly explore many different sections of the state space. Block updating is one extension of the basic MCMC methodology that can be of aid. In a block updating MCMC algorithm, one designs a series of proposal densities similar to

$$q(\theta, x_{1:K} \mid x_{K+1:n}),$$
$$q(x_{k+1:k+K} \mid \theta, x_{1:k}, x_{k+K+1:n}) \quad \text{for } k \in \{K, 2K, \ldots \},$$

for some $1 \leq K \leq n$. The adjacent blocks of $K$ random variables are updated in turn in an M-H algorithm or a Gibbs sampler. Thus, a block updating version of Algorithm 20 uses fewer sampling iterations to obtain a new sample. A block updating version of Algorithm 19 proposes lower dimensional moves, which might have a higher probability of being accepted. In practice, block updating must be carefully crafted. If $K$ is too small, then the resulting MCMC algorithm will still require many iterations to yield a new sample. If $K$ is too large, then it may become difficult to find a suitable sequence (2.15.1).

2.16 Parallel tempering for Markov chain Monte Carlo

When the target $\pi$ is highly multi-modal with large regions of low probability density in between modes, the basic M-H and Gibbs algorithms can again have long mixing times. Algorithms 19 and 20 transition by conditioning on their current state, and so the proposals will likely assign high probability to local moves within a current mode. This makes it difficult to jump to new and far away modes. Even the independent M-H has a tendency to
remain in a single mode for many iterations, as it is challenging to determine a proposal $q$ that can facilitate jumps between modes when one does not know where the modes of the target actually reside.

Parallel tempering is a scheme that is more adapted for sampling from highly multimodal targets ([30], [66]). In a parallel tempering MCMC algorithm, one runs $N > 1$ Metropolis-Hastings or Gibbs algorithms that each target $\pi$. We label each chain with the index $i$. For a sequence of real-valued constants $1 = T_1 < T_2 < \cdots < T_i < \cdots < T_N < \infty$, each chain $i$ targets a version of $\pi$ that is scaled by $T_i$ (i.e., multiplied by $T_i$). Note that those chains operating under higher values of $T_i$ jump more easily between modes of the target because the modes are closer together. As the $N$ chains run, a randomly chosen chain $i$ is proposed to swap its samples with either the $(i - 1)^{\text{th}}$ chain or the $(i + 1)^{\text{th}}$ chain, and a swap is accepted according to some Metropolis-Hastings criterion that depends on the ratio $T_{i-1}/T_i$ or $T_i/T_{i+1}$. After many of these swaps, the hope is that chains propagating under lower values of $T_i$ are exploring regions of the state space that otherwise would have only been found under high values of $T_i$. Furthermore, at any time point after the burn in period, the samples that are currently being propagated by the first chain (which, recall, is targeting a version of $\pi$ that is scaled by $T_1 = 1$) are considered to be samples from the true target $\pi$.

Parallel tempering can be very difficult to implement, as it is challenging to find a sequence $1 = T_1 < T_2 < \cdots < T_i < \cdots < T_N < \infty$ that encourages wide exploration of the state space and allows for the high probability of a swap being accepted. Some adaptive schemes have been proposed in the literature to deal with these issues; see [69] for an example.

### 2.17 Exact approximation of Markov chain Monte Carlo

If a target density $\pi$ is difficult or expensive to compute, then calculating the acceptance probability of a Metropolis-Hastings algorithm is also a challenge. Promising work has been done in the direction of constructing exact approximations of MCMC algorithms that allow one to circumvent the calculation of the exact acceptance probability in this scenario. In an exact approximation to an MCMC algorithm, auxiliary random variables are used to
define an extended target $\pi^N$ that admits the true target of interest $\pi$ as a marginal. An MCMC algorithm is then defined to directly draw samples from $\pi^N$.

To illustrate the basic concept, we follow the discussion of [92] and establish the following constraints. We assume that an ideal Metropolis-Hastings algorithm (similar to Algorithm 19) exists, where one can theoretically target $\pi(\theta)$ via the proposal $q(\theta | \zeta)$ for some previously accepted draw $\zeta \sim \pi$. Defining a Monte Carlo estimate of $\pi(\theta)$ to be $r(\theta)$, assume $\mathbb{E}[r(\theta) | \theta] = \pi(\theta)$, where the expectation is taken with respect to the estimate’s noise. We define $Z = r(\theta) / \pi(\theta)$ to be the noise of the estimate and let $Z \sim q(\cdot | \theta)$. The final condition that we will impose is that $\mathbb{E}[Z | \theta] = 1$.

Given these constraints, one can define an extended target $\pi^N(\theta, z) = \pi(\theta) q(z | \theta) z = r(\theta) q(z | \theta)$. It is clear that this target admits $\pi(\theta)$ as a marginal, as $\mathbb{E}[Z | \theta] = 1$. It is also clear that one can define a Metropolis-Hastings algorithm to target $\pi^N(\theta, z)$ via the proposal $q(z | \theta) q(\theta | \zeta)$. Hence, one has an exact approximation of a Metropolis-Hastings algorithm that targets $\pi(\theta)$.

This idea was first introduced in [5] in the form of the grouped independence Metropolis-Hastings (GIMH) algorithm (see also [3] for a good introduction). To run GIMH, consider the target $\pi(\theta)$ as above, from which we assume it is difficult to sample even though an MCMC algorithm does exist in theory. Also, consider the real-valued auxiliary random variable $Z$, and the artificial target

$$ r(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{\pi(\theta, z_i)}{q(z_i | \theta)}, \quad (2.17.1) $$

where $q(z_i | \theta)$ is a proposal density with which one can sample the $i$th component of the auxiliary random variable $Z$. Assuming it is easier to target $r(\theta)$ than it is to target $\pi(\theta)$, samples of $\pi(\theta)$ can be obtained marginally by drawing from $r(\theta)$ via GIMH (see Algorithm 21). To see this more clearly, [5] re-writes the acceptance probability (2.17.2)
Algorithm 21 Grouped independence Metropolis-Hastings (GIMH)

- Step 0: Set $\theta^0$ arbitrarily. Set $k = 1$.
- Step 1: Sample $\theta^* | \theta^{k-1} \sim q(\cdot | \theta^{k-1})$.
- Step 2: For $i \in \{1, \ldots, N\}$, sample $Z^*_i | \theta^* \sim q(\cdot | \theta^*)$.
- Step 3: With acceptance probability

$$
1 \wedge \frac{r(\theta^*) q(\theta^{k-1} | \theta^*)}{r(\theta^{k-1}) q(\theta^* | \theta^{k-1})},
$$

(2.17.2)

set $\theta^k = \theta^*$ and $Z^k = Z^*$. Otherwise, discard $\theta^*$ and $Z^*$ and set $\theta^k = \theta^{k-1}$ and $Z^k = Z^{k-1}$. Set $k = k + 1$. Return to the start of Step 1.

2.18 Particle Markov chain Monte Carlo

Particle Markov chain Monte Carlo [2] algorithms are similar to GIMH. In a PMCMC algorithm, we still aim to target an extended density $\pi^N(\theta, z)$ that yields the true target of interest $\pi$ as a marginal, only now $\pi$ is extended in such a way that SMC algorithms can be used to sample some of the variables of $\pi^N$. More specifically, consider the true target of the form (2.14.1). We have already seen in Section 2.7 that $\pi_{\theta}(x_{1:n})$ can be targeted using forms of Algorithm 3. In [2], the authors propose using Algorithm 3 within an MCMC proposal
density to target an extension of \( \pi (\theta, x_{1:n}) \).

To develop the PMCMC methodology, we start by building the extended target \( \pi^N \). First, consider that when one excludes the very final resampling step, the joint density of the simulated variables through time \( n \) of Algorithm 3 will be

\[
\psi_0(x_{1:n}, a_{1:n-1}) = \prod_{i=1}^{N} q_0(x_1^i) \prod_{j=2}^{n} r(a_{j-1} \mid w_{j-1}) \prod_{i=1}^{N} q_0(x_j^i \mid x_{1:j-1}^i), \tag{2.18.1}
\]

where \( r(a_{j-1} \mid w_{j-1}) = \prod_{i=1}^{N} w_{j-1}^{a(i)} \). Given a value of \( \theta \), the density of \( \{x_{1:n}^{1:N}, a_{1:n-1}^{1:N}\} \) conditioned on \( X_{1:n}^k = x_{1:n}^k, A_{1:n-1}^k = a_{1:n-1}^k \) will be

\[
\frac{\psi_0(x_{1:n}, a_{1:n-1})}{q_0(x_{1:n}^k) \left( \prod_{j=2}^{n} w_{j-1}^{a(k)} q_0(x_j^k \mid x_{1:j-1}^k) \right)}. \tag{2.18.2}
\]

One can use (2.18.1) and (2.18.2) to establish an extended target as

\[
\pi^N \left( k, \theta, x_{1:n}, a_{1:n-1} \right) = \frac{\pi \left( \theta, x_{1:n}^k \right)}{N^n} \cdot \frac{\psi_0(x_{1:n}, a_{1:n-1})}{q_0(x_{1:n}^k) \left( \prod_{j=2}^{n} w_{j-1}^{a(k)} q_0(x_j^k \mid x_{1:j-1}^k) \right)}, \tag{2.18.3}
\]

where

\[
\frac{\pi \left( \theta, x_{1:n}^k \right)}{N^n} = \pi^N \left( \theta, x_{1:n}^k, a_{1:n-1}^k \mid \cdots \right).
\]

The random variable \( K \) corresponds to the index of a single sample \( (x_{1:n}^k, a_{1:n-1}^k) \) that is resampled from \( \{x_{1:n}^{1:N}, a_{1:n-1}^{1:N}\} \). Thus, if one has a sample \( \{k, \theta, x_{1:n}, a_{1:n-1}\} \) from (2.18.3), the path \( (x_{1:n}^k, a_{1:n-1}^k) \) and the parameter \( \theta \) can be selected as a sample from the marginal \( \pi (\theta, x_{1:n}) \). To obtain the sample \( \{k, \theta, x_{1:n}, a_{1:n-1}\} \), one can target (2.18.3) via the particle marginal Metropolis-Hastings (PMMH) algorithm (see Algorithm 22), whose proposal density takes the form

\[
q^N \left( k, \theta, x_{1:n}, a_{1:n-1} \right) = q (\theta \mid \zeta) \psi_0(x_{1:n}, a_{1:n-1}) w_n^k.
\]

\( q (\theta \mid \zeta) \) is the density that proposes a new value \( \theta \in \Theta \) conditional on a current accepted value \( \zeta \in \Theta \) and \( w_n^k \) is the probability of resampling the path \( (x_{1:n}^k, a_{1:n-1}^k) \).

Recall that if \( \pi (\theta, x_{1:n}) = \pi (\theta, x_{1:n} \mid y_{1:n}) \) and \( \gamma (\theta, x_{1:n}) = \pi (\theta, x_{1:n} \mid y_{1:n}) \), then the
Algorithm 22 Particle marginal Metropolis-Hastings (PMMH)

- Step 0: Set $l = 0$. Set $\theta^l$ arbitrarily. All remaining random variables can be sampled from their full conditionals defined by the target (2.18.3):
  - Sample $x_{1:n}^l, a_{1:n-1}^l | \cdots \sim \psi_{\theta^l} (\cdot)$ via Algorithm 3, excluding the final resampling step.
  - Choose $k^l \propto W_n^{l,k^l}$.
  
  Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta^l,1:n}^l$, via (2.7.1).

- Step 1: Set $l = l + 1$. Sample $\theta^* \sim q (\cdot | \theta^{l-1})$. All remaining random variables can be sampled from their full conditionals defined by the target (2.18.3):
  - Sample $x_{1:n}^*, a_{1:n-1}^* | \cdots \sim \psi_{\theta^*} (\cdot)$ via Algorithm 3, excluding the final resampling step.
  - Choose $k^* \propto W_n^{*,k^*}$.
  
  Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta^*,1:n}^*$, via (2.7.1).

- Step 2: With acceptance probability

$$1 \wedge \frac{\pi^N (k^*, \theta^*, x_{1:n}^*, a_{1:n-1}^*)}{\pi^N (k^{l-1}, \theta^{l-1}, x_{1:n}^{l-1}, a_{1:n-1}^{l-1})} \cdot \frac{q^N (k^{l-1}, \theta^{l-1}, x_{1:n}^{l-1}, a_{1:n-1}^{l-1})}{q^N (k^*, \theta^*, x_{1:n}^*, a_{1:n-1}^*)} = 1 \wedge \frac{\pi (\theta^*)}{\pi (\theta^{l-1})} \frac{q (\theta^{l-1} | \theta^*)}{q (\theta^* | \theta^{l-1})} \frac{\hat{Z}_{\theta^*,1:n}^*}{\hat{Z}_{\theta^{l-1},1:n}^{l-1}},$$

set $k^l = k^*$, $\theta^l = \theta^*$, $x_{1:n}^l = x_{1:n}^*$, and $a_{1:n-1}^l = a_{1:n-1}^*$. Otherwise, set $k^l = k^{l-1}$, $\theta^l = \theta^{l-1}$, $x_{1:n}^l = x_{1:n}^{l-1}$, and $a_{1:n-1}^l = a_{1:n-1}^{l-1}$.

Return to the beginning of Step 1.

normalising constant of $\pi_\theta (x_{1:n}) = \pi_\theta (x_{1:n} | y_{1:n})$ is $Z_{\theta,1:n} = p_\theta (y_{1:n})$ (see the review of HMMs in Section 2.2). In this scenario, the acceptance probability of Algorithm 22 suggests that PMMH targets $\pi (\theta | y_{1:n}) \propto \pi (\theta) p_\theta (y_{1:n})$, which is a marginal density of $\pi (\theta, x_{1:n} | y_{1:n})$. Hence the name particle marginal Metropolis-Hastings.

A PMCMC version of the Gibbs sampler also targets (2.18.3). Similar to Algorithm 20,
the particle Gibbs (PG) sampler draws from the following conditional densities, in turn:

\[
\pi^N \left( \theta \mid \mathcal{X}_{1:n}, a_{1:n-1}, k \right) = \pi \left( \theta \mid x^k_{1:n} \right)
\]

\[
\pi^N \left( \mathcal{X}_{1:n}, a_{1:n-1} \mid k, \theta \right) = \frac{\psi_\theta \left( \mathcal{X}_{1:n}, a_{1:n-1} \right)}{q_\theta \left( x^k_1 \right) \prod_{j=2}^{n} w^{a(k)}_{j-1} q_\theta \left( x^k_j \mid x^{a(k)}_{1:j-1} \right)}
\]

\[
\pi^N \left( k \mid \theta, \mathcal{X}_{1:n}, a_{1:n-1} \right) = w^k_n,
\]

where we assume it is possible to sample from \( \pi \left( \theta \mid x_{1:n} \right) \). In order to sample from \( \pi^N \left( \mathcal{X}_{1:n}, a_{1:n-1} \mid k, \theta \right) \), the authors of [2] develop a conditional version of Algorithm 3 where the path \( x^k_{1:n}, a^k_{1:n-1} \) is chosen to survive all resampling steps (see Algorithm 23). The PG sampler is formalised as Algorithm 24.

### 2.18.1 Extensions of particle Markov chain Monte Carlo

PMCMC is a new, active area of research, and its success is helping to popularise the use of extended state spaces in simulation methods. It is likely that computational statistics will see more work using extended state spaces in the years to come. Some extensions/applications of the original PMMH and PG algorithms have already been developed.

The SMC\(^2\) algorithm [19] is one such application in Bayesian parameter inference for HMMs where the aim is to sample from \( \pi \left( \theta, x_{1:n} \mid y_{1:n} \right) \). Essentially, the SMC\(^2\) algorithm initialises by drawing \( M \) samples of a parameter \( \theta \) from its prior distribution. For each \( \theta^i \), one runs Algorithm 7 (placing the resampling step before the sampling step) and computes the marginal likelihood estimates \( \hat{Z}_{\theta^i, 1:n} \) at each time step \( n \) of Algorithm 7. The marginal likelihood estimates act as weights for each sample \( \left( \theta^i, x^i_{1:n}, a^i_{1:n-1} \right) \). As the particle filters are running, some degeneracy criterion is monitored. When that criterion is satisfied, a weighted mixture of PMCMC kernels is used to rejuvenate the sample \( \left( \theta^i, x^i_{1:n}, a^i_{1:n-1} \right) \) and its corresponding weight is reset to one. The SMC\(^2\) algorithm has shown to be effective in numerical simulations where \( \pi \left( \theta, x_{1:n} \mid y_{1:n} \right) \) is highly multi-modal.

Specific PMCMC algorithms have also been designed to sample from multiple change point models [90]. In a multiple change-point problem, one tries to segment a sequence of time-series observations \( y_1, y_2, \ldots, y_T \) by choosing change-point locations \( 0 < \tau_1 < \cdots < \)
Algorithm 23 Conditional SMC algorithm

- **Step 0:** Set $k, a_{1:T-1}^k$ to be the ancestry of a path which is to survive all resampling steps.

- **Step 1:** For $i \in \{1, \ldots, N\}$ and $i \neq a_1^k$, sample $X_i^1 \sim q_0(\cdot)$ and compute the un-normalised weight:
  
  $$W_i^1 = \frac{\gamma_1(x_i^1)}{q_0(x_i^1)}.$$ 

  For $i \in \{1, \ldots, N\}$ and $i \neq a_1^k$, sample $A_i^1 \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $j^{th}$ probability $w_{ij}^1 \propto W_i^1$. The sample $\{a_i^1:N\}$ are the indices of the resampled particles. Set all normalised weights equal to $1/N$, and set $n = 2$.

- **Step 2:** If $n = T$, go to Step 3. Otherwise, for $i \in \{1, \ldots, N\}$ and $i \neq a_n^k$, sample $X_i^n \mid x_{1:n-1}^{a(i)} \sim q_0(\cdot \mid x_{1:n-1}^{a(i)})$ and compute the un-normalised weight:
  
  $$W_i^n = \frac{\gamma_n(x_i^n \mid x_{1:n-1}^{a(i)})}{q_0(x_i^n \mid x_{1:n-1}^{a(i)})}.$$ 

  For $i \in \{1, \ldots, N\}$ and $i \neq a_n^k$, sample $A_i^n \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $j^{th}$ probability $w_{ij}^n \propto W_i^n$. Set all normalised weights equal to $1/N$, and set $n = n + 1$. Return to the start of Step 2.

- **Step 3:** For $i \in \{1, \ldots, N\}$ and $i \neq k$, sample $X_T^i \mid x_{1:T-1}^{a(i)} \sim q_0(\cdot \mid x_{1:T-1}^{a(i)})$ and compute the un-normalised weight:
  
  $$W_T^i = \frac{\gamma_{T}(x_T^i \mid x_{1:T-1}^{a(i)})}{q_0(x_T^i \mid x_{1:T-1}^{a(i)})}.$$
2.18 Particle Markov chain Monte Carlo

Algorithm 24 Particle Gibbs (PG) sampler

- Step 0: Set $l = 0$. Set a path $x_{1:n}$ arbitrarily. Set its genealogy to $s_{1:n} = 1$.
- Step 1: Set $l = l + 1$. Sample $\theta^l | \cdots \sim \pi (\cdot | x_{1:n})$.
- Step 2: Sample

$$x^l_{1:n} | \theta^l, x_{1:n}, s_{1:n} \propto \frac{\psi^l(x^l_{1:n}, a^l_{1:n-1})}{q_{\theta^l}(x_1)} \left( \prod_{j=2}^n w^l_{j-1} q^l(x_j | x_{1:j-1}) \right)$$

via Algorithm 23, where $x_{1:n}$ will survive all resampling steps.
- Step 3: Sample $k^l \sim W_n^{l,k^l}$, and set $s_{1:n} = k^l, a^l_{1:n-1}$. Set $x_{1:n} = x^k_{1:n}$.

Return to the beginning of Step 1.

$T$ that identify when different trends are present in the data. MCMC algorithms applied to sample from the posteriors of these models have been slow mixing in the past, due to the strong correlations between change-points. The specially designed PMCMC algorithms of [90] offer a more efficient solution for sampling from the posteriors by using a type of SMC (see [33]) that is suited for change point models. Essentially, the SMC algorithm of [33] exploits the fact that the change point locations take values in a finite space to avoid the random importance sampling step of standard SMC. It also employs a resampling algorithm that avoids particle duplication.

Finally, the original PMMH algorithm of [2] performs inference on models where it is assumed that the normalising constant is not a function of $\theta$. [32] proposes a new PMMH algorithm that can be used for inference in models where the normalising constant is an intractable function of $\theta$ (partially observed Markov random fields are one such type of model, and they can be used to study social networks, pixelated images, and the structure of the world wide web). For a noisy observation $Y \in \mathbb{R}^d_Y$ of a hidden random variable $X \in \mathbb{R}^d_X$ and model parameter $\theta \in \Theta \subseteq \mathbb{R}^d_{\theta}$, first suppose we want to sample from a posterior that factorises as

$$\pi(\theta | x, y) \propto \frac{\pi(\theta) \gamma(x, y | \theta)}{Z(\theta) \pi(y)},$$

(2.18.4)
where

- $\pi(\theta)$ is a prior density of $\theta$,
- $\gamma(x,y | \theta)$ is the likelihood of $(x,y)$,
- $\pi(y)$ is the probability density of the observed random variable, and
- $Z(\theta)$ is an intractable normalising constant that is a function of $\theta$.

Allowing $\zeta \in \Theta$ to be an accepted parameter sample of (2.18.4), a PMMH algorithm could be used to target $\pi(\theta | x,y)$: it would use an SMC sampler (see Section 2.13) in place of Algorithm 3 and it would rely on the proposal $q(\theta | \zeta)$. However, this PMMH algorithm will have the term $Z(\zeta)/Z(\theta)$ appear in its acceptance ratio, therefore making it impossible to run. [32] uses an idea from [71] to eliminate the ratio $Z(\zeta)/Z(\theta)$ by further extending the target and defining a new PMMH to sample from that extended target.

### 2.19 Markov chain Monte Carlo moves within sequential Monte Carlo

PMCMC is not the only example of SMC and MCMC being combined in the literature. In order to combat the path degeneracy problem faced by SMC algorithms (see Section 2.7.1), some have proposed embedding MCMC within SMC. Following [40], we introduce the sequence of Markov kernels $\{\phi_\theta(z_{1:k} | x_{1:k})\}_{k \in \{1,...,n\}}$ with respective stationary densities $\{\pi_\theta(z_{1:k})\}_{k \in \{1,...,n\}} = \{\pi_\theta(x_{1:k})\}_{k \in \{1,...,n\}}$. One could construct an SMC algorithm with MCMC moves as in Algorithm 25. The MCMC moves serve the purpose of randomising the resampled paths, thereby creating more unique trajectories after the resampling step. This algorithm constitutes valid IS, as it simply corresponds to sampling from the enlarged spaces $\{q_\theta(x_{1:k})\phi_\theta(z_{1:k} | x_{1:k})\}_{k \in \{1,...,n\}}$ and extending the targets to $\{\pi_\theta(z_{1:k}) K_\theta(x_{1:k} | z_{1:k})\}_{k \in \{1,...,n\}}$, where $K_\theta$ is some Markov kernel:

$$
\int \pi_\theta(x_{1:n}) \, dx_{1:n} = \int \int q_\theta(x_{1:n}) \phi_\theta(z_{1:n} | x_{1:n}) \frac{\pi_\theta(z_{1:n}) K_\theta(x_{1:n} | z_{1:n})}{q_\theta(x_{1:n}) \phi_\theta(z_{1:n} | x_{1:n})} \, dx_{1:n} \, dz_{1:n}.
$$

Note that, if one so desires, it is straightforward to modify Algorithm 25 so that the MCMC move only updates a path of fixed length $B$ at each time step $n > B$. 
Algorithm 25 SMC with MCMC moves

- Step 1: For $i \in \{1, \ldots, N\}$, sample $X_i^{1} \sim q_{\theta} (\cdot)$ and compute the un-normalised weight:
  
  $$W_i^1 = \frac{\gamma_{\theta}(x_i^1)}{q_{\theta}(x_i^1)}.$$  

  For $i \in \{1, \ldots, N\}$, sample $A_i^1 \in \{1, \ldots, N\}$ from a discrete distribution on \{1, \ldots, N\} with $j$th probability $w_i^1 \propto W_i^1$. The sample $\{a_i^{1:N}\}$ are the indices of the resampled particles. Set all normalised weights equal to $1/N$, and set $n = 2$.

- Step 2: For $i \in \{1, \ldots, N\}$, sample $Z_{1:n-1}^i \sim \phi_{\theta}(\cdot | x_{1:n-1}^{a(i)})$ and set $x_{1:n-1}^{a(i)} = z_{1:n-1}^i$.

- Step 3: For $i \in \{1, \ldots, N\}$, sample $X_{n}^i | x_{1:n-1}^{a(i)} \sim q_{\theta} (\cdot | x_{1:n-1}^{a(i)})$ and compute the un-normalised weight:
  
  $$W_n^i = \frac{\gamma_{\theta}(x_n^i | x_{1:n-1}^{a(i)})}{q_{\theta}(x_n^i | x_{1:n-1}^{a(i)})}.$$  

  For $i \in \{1, \ldots, N\}$, sample $A_{1:n}^i \in \{1, \ldots, N\}$ from a discrete distribution on \{1, \ldots, N\} with $j$th probability $w_n^j \propto W_n^j$. Set all normalised weights equal to $1/N$, and set $n = n + 1$. Return to the start of Step 2.
2.20 Parallel computing

As the amount of data being produced by researchers continuously grows, and as these massive datasets are accompanied by models whose dimensionality is also increasing, the classical computational techniques are reaching their limits. Practitioners are finding that Monte Carlo methods are requiring an excessively long time to run because the algorithms need to draw many samples and run for very long burn in periods to effectively target the densities of interest. This problem is driving the research trends in the wider Monte Carlo community, and the current chapter has already shown some examples of how research is developing to answer these demands of the real world. Additionally, we have yet to mention that new adaptive methods are also being developed so that tuning parameters can be automatically optimised as an algorithm iterates (some relevant examples include adaptive MCMC [4], adaptive SMC samplers [35], and Chapter 5 of this thesis). Adaptive PMCMC methods have also been developed for non-linear hidden Markov model selection [76].

In the literature, one can see examples of increased interest in hardware as well. The use of parallel computing and graphical processing units ([63], [86]) to shorten the runtime of Monte Carlo algorithms (including SMC and PMCMC) is gaining popularity. Monte Carlo techniques must obtain samples consecutively when implemented on a single central processing unit (CPU), but graphical processing units (which consist of many independent processors that are linked together on a single electronic circuit) offer an inexpensive avenue through which parallel sampling can be performed. As graphical processing units (GPUs) are so easily available to practitioners, more research is being conducted on how the technology can best be used. Some GPU implementations of Monte Carlo algorithms have been shown to run 250 times faster than their counterpart CPU implementations [63]. In the case of SMC, the resampling step still prevents the chains from running completely independently of one another. However, new resampling schemes are being tested to address this (see [72] and [73] for examples).
2.21 Final remarks

This literature review has presented SMC and PMCMC in the context of importance sampling. In a more theoretical discussion, the algorithms could have been presented in an even more abstract sense, where, for example, an incremental weight would be any strictly positive, bounded, real-valued function that assigns a potential to a sampled path. As the original work in this thesis mostly pertains to advances in methodology, such an advanced introduction would have been unnecessary and off topic. The reader who is interested in a more theoretical introduction to particle methods should refer to [23].
Chapter 3

Generalised Two-filter Smoothing and Particle Markov chain Monte Carlo

Recall that the acceptance probability of a PMMH algorithm depends upon the unbiased estimate of the normalising constant, \( (2.7.1) \), obtained via SMC. Therefore, it is reasonable to assume that reducing the variance of the estimate of the normalising constant may improve the convergence of a PMMH algorithm. With this intuition, we set out to develop a new, more precise estimate \( \hat{Z}_\theta \) with the ultimate goal of developing faster converging PMCMC methods for HMMs.

Section 3.1 presents our new estimate, which uses generalised two-filter smoothing to approximate \( Z_{\theta,1:T} = p_\theta (y_{1:T}) \) for an HMM as the product of three marginals: \( p_\theta (y_{1:t-2}), p_\theta (y_{t+2:T}) \), and

\[
\int f_\theta (x_t \mid x_{t-1}) g_\theta (y_t \mid x_t) f_\theta (x_{t+1} \mid x_t) \times \\
\pi_\theta (x_{t-1} \mid y_{1:t-1}) \pi_\theta (x_{t+1} \mid y_{t+1:T}) \, dx_{t-1:t+1},
\]

for a fixed time \( T > 4 \) and some \( t \in \{3, \ldots, T - 2\} \). Essentially, we use the output of Algorithm 13 (which is an \( \mathcal{O} (N) \) method) to calculate an approximation of each of these marginals; the exact formula is presented in Section 3.1.2 as equation (3.1.4). Calculation of our estimate is \( \mathcal{O} (N) \) in complexity, which can be beneficial when one wants to quickly run SMC repeatedly (as in PMCMC). However, a major potential issue for our strategy is
that the samples which are used to calculate $\hat{p}_\theta (y_{1:t-2})$ are drawn independently from the samples which are used to calculate $\hat{p}_\theta (y_{t+2:T})$. Thus, our estimate can have a high variability under the wrong algorithmic settings, thereby nullifying the benefits of $\hat{Z}_{\theta,1:T}$ being $O(N)$ in complexity. We do identify algorithmic settings that will significantly reduce the variance (see below), but these settings will often times not be available analytically and they will need to be approximated.

Section 3.1 also presents a central limit theorem for our estimate. The asymptotic variance term of that central limit theorem is very difficult to evaluate analytically, and so we explore the estimate’s variability via two numerical examples (a linear Gaussian model and a stochastic volatility model) in Section 3.2. The numerical analysis leads to two insights:

• within Algorithm 13, when one sets the sequence of pseudo-priors $\{\xi_{n,\theta}\}_{n \geq 1}$ to be $\{\pi (x_n \mid y_{1:n-1})\}_{n \geq 1}$, the incremental weights of Algorithm 11 are equal to one; and

• our $O(N)$ unbiased estimate of the marginal likelihood has a much lower variance than that obtained by Algorithm 7 (the SMC filter) when the sequence $\{\xi_{n,\theta}\}_{n \geq 1} = \{\pi (x_n \mid y_{1:n-1})\}_{n \geq 1}$ is employed.

These findings motivate the application of our unbiased estimate in developing a new PMMH algorithm which uses generalised two-filter smoothing to sample from the latent process of an HMM in lieu of the SMC filter; we develop the algorithm in Section 3.4 and compare it in Section 3.5 to single-filter PMCMC. In those same sections, we also formulate and test a new PG sampler. The new PMCMC algorithms only converge faster than their single-filter counterparts when $\xi_{n,\theta} (x_n) = \pi_{\theta} (x_n \mid y_{1:n-1})$, meaning that using two filters as opposed to one is not actually all that beneficial when running PMCMC. Rather, the optimal sequence $\{\xi_{n,\theta}\}_{n \geq 1} = \{\pi (x_n \mid y_{1:n-1})\}_{n \geq 1}$ is the real driver of any boost in performance, and perhaps one should instead aim to employ Algorithm 11 (with $\{\xi_{n,\theta}\}_{n \geq 1} \approx \{\pi (x_n \mid y_{1:n-1})\}_{n \geq 1}$) when possible.

Finally, the discovery of the optimal sequence $\{\xi_{n,\theta}\}_{n \geq 1} = \{\pi (x_n \mid y_{1:n-1})\}_{n \geq 1}$ is discussed further in Section 3.3, where we identify an ideal $O(N)$ smoothing algorithm that only requires a single backward pass. The ideal algorithm is analytically unavailable for most models, but we do formulate and test algorithms that approximate it in Sections 3.6
Chapter 3. Generalised Two-filter Smoothing and Particle Markov chain Monte Carlo

Unfortunately, these alternative algorithms prove to have high variance. Section 3.8 concludes the chapter with a summary and discussion.

3.1 Unbiased estimates of the marginal likelihood for HMMs

Our effort to expedite the convergence of a PMMH algorithm targeting an HMM begins with a study of how one might reduce the variance of $\hat{Z}_\theta$, which is used to calculate the acceptance probability of a PMMH algorithm. Recall the HMM of Section 2.2. If one were to run Algorithm 7 through time $T$, an unbiased estimate of the marginal likelihood could be calculated via (2.7.1). Using two filters as opposed to one could possibly reduce the variance of an estimate of $Z_{\theta,1:T}$. Thus, we now show that it is possible to use Algorithm 13 to calculate an $O(N)$ unbiased estimate of $Z_{\theta,1:T}$. We first review the derivation of an $O(N^2)$ unbiased estimate, from which a derivation of the $O(N)$ unbiased estimate follows.

3.1.1 Estimate of complexity $O(N^2)$

We begin with the two-filter smoothing formula (2.2.7),

$$
\pi_\theta(x_t | y_{1:T}) = \frac{\pi_\theta(x_t | y_{1:t-1}) \pi_\theta(y_t | x_t) \rho_{t,\theta}(x_t)}{\tilde{\pi}_\theta(x_t | y_{1:T})},
$$

where $\tilde{\pi}_\theta$ is the artificial density constructed to be the target of Algorithm 11. This implies

$$
p_{\theta}(y_{1:T} | y_{1:t-1}) \pi_\theta(x_t | y_{1:T}) = \int \frac{\pi_\theta(x_{t-1} | y_{1:t-1}) f_\theta(x_t | x_{t-1}) d x_{t-1}}{\rho_{t,\theta}(x_t)} \tilde{\pi}_\theta(x_t | y_{1:T}) \tilde{p}_\theta(y_{1:T}),
$$
which subsequently implies

\[
p_{\theta}(y_{1:T}) = p_{\theta}(y_{1:t-1}) \tilde{p}_{\theta}(y_{t:T}) \int \pi_{\theta}(x_{t-1} \mid y_{1:t-1}) \frac{f_{\theta}(x_{t} \mid x_{t-1})}{\xi_{t,\theta}(x_{t})} \tilde{\pi}_{\theta}(x_{t} \mid y_{t:T}) \, dx_{t:t-1}
\]

\[
= p_{\theta}(y_{1:t-1}) \tilde{p}_{\theta}(y_{t:T}) \int \frac{f_{\theta}(x_{t} \mid x_{t-1})}{\xi_{t,\theta}(x_{t})} \times \frac{\pi_{\theta}(x_{t-1} \mid y_{1:t-1}) \tilde{\pi}_{\theta}(x_{t} \mid y_{t:T})}{\pi_{\theta}(x_{t-1} \mid y_{1:t-1})} \times \frac{\tilde{\pi}_{\theta}(x_{t} \mid y_{t:T}) q_{\theta}(x_{t-1} \mid x_{t-2}) q_{\theta}(x_{t} \mid x_{t+1})}{\tilde{\pi}_{\theta}(x_{t} \mid y_{t:T}) q_{\theta}(x_{t-1} \mid x_{t-2}) q_{\theta}(x_{t} \mid x_{t+1})} \, dx_{1:T}.
\]

Recalling the definitions of the weights (2.8.2) and (2.9.7) yields

\[
p_{\theta}(y_{1:T}) = p_{\theta}(y_{1:t-1}) \tilde{p}_{\theta}(y_{t:T}) \int \frac{f_{\theta}(x_{t} \mid x_{t-1})}{\xi_{t,\theta}(x_{t})} \hat{W}_{t-1} \hat{W}_{t} \times \frac{\pi_{\theta}(x_{1:t-2} \mid y_{1:t-2}) \tilde{\pi}_{\theta}(x_{t+1} \mid y_{t+1:T}) q_{\theta}(x_{t-1} \mid x_{t-2}) q_{\theta}(x_{t} \mid x_{t+1})}{\pi_{\theta}(x_{1:t-2} \mid y_{1:t-2}) \tilde{\pi}_{\theta}(x_{t+1} \mid y_{t+1:T}) q_{\theta}(x_{t-1} \mid x_{t-2}) q_{\theta}(x_{t} \mid x_{t+1})} \, dx_{1:T}.
\]

Recall that we are using rightward and leftward arrows to distinguish between forward and backward incremental weights, respectively. The final form of the marginal likelihood is

\[
Z_{\theta,1:T} = p_{\theta}(y_{1:t-2}) \tilde{p}_{\theta}(y_{t+1:T}) \int \frac{f_{\theta}(x_{t} \mid x_{t-1})}{\xi_{t,\theta}(x_{t})} \hat{W}_{t-1} \hat{W}_{t} \times \pi_{\theta}(x_{t-2} \mid y_{1:t-2}) \tilde{\pi}_{\theta}(x_{t+1} \mid y_{t+1:T}) q_{\theta}(x_{t-1} \mid x_{t-2}) q_{\theta}(x_{t} \mid x_{t+1}) \, dx_{1:t-2:t+1}.
\] (3.1.1)

Algorithms 7 and 11 can be used to approximate (3.1.1) via the formula

\[
\hat{Z}_{\theta,1:T} = \prod_{n=1}^{t-2} \left[ \frac{1}{N} \sum_{j=1}^{N} \hat{W}_{jn}^{T} \right] T-1 \prod_{n=0}^{T-2} \left[ \frac{1}{N} \sum_{j=1}^{N} \hat{W}_{jn}^{T-n} \right] \left[ \frac{1}{N^{2}} \sum_{i=1}^{N} \sum_{j=1}^{N} \hat{W}_{ji}^{T} \hat{W}_{lj}^{T-1} \frac{f_{\theta}(x_{i} \mid x_{j})}{\xi_{t,\theta}(x_{i})} \right].
\] (3.1.2)
3.1.2 Estimate of complexity $\mathcal{O}(N)$

The estimate (3.1.2) is perhaps slightly undesirable, as it has a computational cost of $\mathcal{O}(N^2)$ due to the term

$$
\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \tilde{W}_t^i \tilde{W}_t^{j-1} f_\theta \left( \frac{\tilde{x}_t^i}{\tilde{x}_t^{j-1}} | \tilde{x}_1^i \right) \xi_{t,\theta} \left( \tilde{x}_t^i \right).
$$

This term dictates how the forward and backward filters are combined at the meeting time $t$. Thus, we can reduce the computational cost to $\mathcal{O}(N)$ if we change how the forward and backward filters meet. To that end, we turn to Algorithm 13.

In order to use Algorithm 13, the expression (3.1.1) must be adjusted. Recalling the definitions of the weights (2.8.2) and (2.9.7) yields

$$
p_\theta(y_{1:T}) = p_\theta(y_{1:t-1}) \tilde{p}_\theta(y_{t+1:T}) \int \pi_\theta(x_{t-1} | y_{1:t-1}) \tilde{\pi}_\theta(x_{t+1} | y_{t+1:T}) \times
$$

$$
\frac{f_\theta(x_t | x_{t-1}) f_\theta(x_{t+1} | x_t)}{\xi_{t+1,\theta}(x_{t+1})} g_\theta(y_t | x_t) \, dx_{t-1:t+1}.
$$

We can again apply some standard manipulations to obtain

$$
p_\theta(y_{1:T}) = p_\theta(y_{1:t-1}) \tilde{p}_\theta(y_{t+1:T}) \int \pi_\theta(x_{t+1} | y_{1:t-1}) \frac{f_\theta(x_t | x_{t-1}) f_\theta(x_{t+1} | x_t)}{\xi_{t+1,\theta}(x_{t+1})} g_\theta(y_t | x_t) \times
$$

$$
\frac{\pi_\theta(x_{t+2} | y_{1:t-2}) q_\theta(x_{t+1} | x_{t+2})}{\pi_\theta(x_{t+2} | y_{1:t-2}) q_\theta(x_{t+1} | x_{t+2})} \frac{\tilde{\pi}_\theta(x_{t+1} | y_{t+1:T})}{\pi_\theta(x_{t+1} | y_{t+1:T})} q_\theta(x_{t+1} | x_{t+2}) \
\times
\pi_\theta(y_{1:t-2} | y_{1:t-2}) \tilde{\pi}_\theta(x_{t+2:T} | y_{t+2:T}) q_\theta(x_{t+1} | x_{t+2}) q_\theta(x_{t+1} | x_{t+2}) \, dx_{1:T},
$$

and using the definitions of the weights (2.8.2) and (2.9.7) leads to

$$
p_\theta(y_{1:T}) = p_\theta(y_{1:t-1}) \tilde{p}_\theta(y_{t+1:T}) p_\theta(y_{1:t-2}) \tilde{p}_\theta(y_{t+2:T}) \times
$$

$$
\int \frac{f_\theta(x_t | x_{t-1}) f_\theta(x_{t+1} | x_t)}{\xi_{t+1,\theta}(x_{t+1})} g_\theta(y_t | x_t) \tilde{W}_t^{-1} \tilde{W}_{t+1} \times
$$

$$
\pi_\theta(x_{t+2} | y_{1:t-2}) \tilde{\pi}_\theta(x_{t+2:T} | y_{t+2:T}) \times
\pi_\theta(y_{1:t-2} | y_{1:t-2}) \tilde{\pi}_\theta(x_{t+2:T} | y_{t+2:T}) q_\theta(x_{t+1} | x_{t+2}) \
q_\theta(x_{t+1} | x_{t+2}) q_\theta(x_{t+1} | x_{t+2}) \, dx_{1:T}.
$$
The final form of the marginal likelihood will be

\[
Z_{\theta,1:T} = p_\theta(y_{1:t-2}) \prod_{i=1}^{T-t+1} \int f_\theta(x_t \mid x_{t-1}) f_\theta(x_{t+1} \mid x_t) \times g_\theta(y_t \mid x_t) \tilde{W}_{t-1} \tilde{W}_{t+1} \pi_\theta(x_{t-2} \mid y_{1:t-2}) \tilde{\pi}_\theta(x_{t+2} \mid y_{t+2:T}) \times q_\theta(x_{t-1} \mid x_{t-2}) q_\theta(x_{t+1} \mid x_{t+2}) \, dx_{t-2:t+2}.
\]

We can run Algorithm 13 and approximate (3.1.3) via the formula

\[
\hat{Z}_{\theta,1:T} = \prod_{n=1}^{t-2} \left[ \frac{1}{N} \sum_{j=1}^{N} \tilde{W}_{j,n} \right] \prod_{n=0}^{T-t-2} \left[ \frac{1}{N} \sum_{j=1}^{N} \tilde{W}_{j,T-n} \right] \left[ \frac{1}{N^3} \sum_{l=1}^{N} \tilde{W}_{l-1} \tilde{W}_{l+1} \right].
\]

Under assumption (A1) (which is detailed in the chapter appendix in Section A), the following theorem states that the estimate (3.1.4) is unbiased. It provides a central limit theorem for the expression as well; the notations for the expression of the asymptotic variance are defined in the chapter appendix in Section A.

**Theorem 3.1.1** We have

\[
\mathbb{E}[\hat{Z}_{\theta,1:T}] = Z_{\theta,1:T} \quad \forall \theta \in \Theta.
\]

In addition, assume (A1). Then for fixed \(T > 4\), \(t \in \{3, \ldots, T-2\}\) and any \(\theta \in \Theta\) we have

\[
\sqrt{N}(\hat{Z}_{\theta,1:T} - Z_{\theta,1:T}) \Rightarrow A_\theta
\]

where \(A_\theta \sim \mathcal{N}(0, \sigma^2_{t,T}(\theta))\) with

\[
\sigma^2_{t,T}(\theta) = \sum_{q=1}^{T-t-1} (t_q \eta_{q,\theta}(1) \tilde{Q}_{q,t+1,\theta}[\tilde{W}_{t+1} I_{gf}(\cdot, \cdot)]) + \sum_{q=0}^{T-t-1} (T_q \eta_{T-q,\theta}(1) \tilde{Q}_{T-q,t+1,\theta}[\tilde{W}_{t+1} I_{gf}(\cdot, \cdot)])
\]

where, for \(\varphi \in B_b(\mathbb{R}^{2d_\epsilon})\),

\[
\sigma^2_{q,\theta}(\varphi) = \sum_{q=1}^{t-1} (q \eta_{q,\theta}(1) \tilde{Q}_{q,t-1,\theta}[\tilde{W}_{t-1} I_{gf}(\cdot, \cdot)]) + \sum_{q=0}^{T-t-1} (T_q \eta_{T-q,\theta}(1) \tilde{Q}_{T-q,t+1,\theta}[\tilde{W}_{t+1} I_{gf}(\cdot, \cdot)])
\]

and

\[
\sigma^2_{q,\theta}(\varphi) = \sum_{q=1}^{t-1} (q \eta_{q,\theta}(1) \tilde{Q}_{q,t-1,\theta}[\tilde{W}_{t-1} I_{gf}(\cdot, \cdot)]) + \sum_{q=0}^{T-t-1} (T_q \eta_{T-q,\theta}(1) \tilde{Q}_{T-q,t+1,\theta}[\tilde{W}_{t+1} I_{gf}(\cdot, \cdot)])
\]
Chapter 3. Generalised Two-filter Smoothing and Particle Markov chain Monte Carlo

In the expression for $\sigma^2_{t-1, \theta} (\phi)$, each summand gives the variance of a forward filter starting at time point $q$ and ending at time point $(t - 1)$. The variance is taken with respect to the sampling law of the forward filter. The expression for $\sigma^2_{t+1, \theta} (\phi)$ has an analogous interpretation for the backward filter.

Remark 3.1.1 The delta method can be used to obtain

\[
\sqrt{N} \left( \log \left( \hat{Z}_{\theta,1:T} \right) - \log \left( Z_{\theta,1:T} \right) \right) \Rightarrow A^*_\theta
\]

where $A^*_\theta \sim \mathcal{N} \left( 0, \frac{\sigma^2_{t, \theta} (\phi)}{Z^2_{\theta,1:T}} \right)$. Under some additional mixing conditions, one can establish that the asymptotic variance obeys

\[
\frac{\sigma^2_{t, \theta} (\phi)}{Z^2_{\theta,1:T}} \leq C_{1, \theta} (t - 1) + C_{2, \theta} (T - t)
\]

where $C_{1, \theta} (t - 1)$ is Algorithm 7’s $L_2$-relative error and $C_{2, \theta} (T - t)$ is Algorithm 11’s $L_2$-relative error. Thus, the central limit theorem provides little intuition on how to select $t$. It simply implies that if the forward filter performs better, one should choose a large $t$, and vice versa.

Remark 3.1.2 For some function $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}$ with $\varphi \in \mathcal{B}_b (\mathbb{R}^d)$, consider the expectation $\mathbb{E}[\varphi (X_t) \mid y_{1:T}]$, which is taken with respect to the joint smoothing distribution, with $3 \leq t \leq T - 2$. One can show that

\[
\frac{1}{\hat{Z}_{\theta,1:T}} \prod_{t=1}^{T-2} \left[ \frac{1}{N} \sum_{j=1}^{N} W^j W^j_{T-n} \right] \prod_{t=0}^{T-2} \left[ \frac{1}{N} \sum_{j=1}^{N} \varphi (x^j_t) W^j_{T-t} \right] \left[ \frac{1}{N^2} \sum_{l=1}^{N} \varphi (x^l_t) W^l_{T-t} \right] \left[ \frac{1}{N^3} \sum_{l=1}^{N} \varphi (x^l_t) W^l_{T-t} \right] \quad (3.1.5)
\]

is an unbiased estimate of $\mathbb{E}[\varphi (X_t) \mid y_{1:T}]$. Furthermore, we have been using $\pi_\theta (x_t, y_{1:t})$ to denote the joint density of $(x_t, y_{1:t})$. If we momentarily abuse our notation and use $\bar{\pi}_\theta (x_t, y_{1:t})$ to denote the probability measure as well, we can denote (3.1.5) as $\hat{Z}_{\theta,1:T}$ and $\mathbb{E}[\varphi (X_t) \mid y_{1:T}]$ as $\bar{\pi}_\theta (\varphi (x_t), y_{1:t}) \mid Z_{\theta,1:T}$. Standard calculations (for example, as on [23, page 301]) then yield

\[
\frac{\hat{Z}_{\theta,1:T}}{\bar{\pi}_\theta (\varphi (x_t), y_{1:t})} - \frac{\pi_\theta (\varphi (x_t), y_{1:t})}{Z_{\theta,1:T}} = \frac{Z_{\theta,1:T}}{\pi_\theta (\varphi (x_t), y_{1:t})} \left( \frac{1}{Z_{\theta,1:T}} \left[ \varphi (x_t) - \frac{\pi_\theta (\varphi (x_t), y_{1:t})}{\bar{\pi}_\theta (\varphi (x_t), y_{1:t})} \right] \right).
\]
3.2 Numerical comparison of $O(N)$ estimates

From the proof in Section A, one can deduce that $Z_{\theta,1:T}/\hat{Z}_{\theta,1:T}$ converges in probability to one and, for $\tilde{\varphi} = \frac{1}{Z_{\theta,1:T}} \left[ \varphi(x_t) - \pi_{\theta}(\varphi(x_t), y_{1:t})/Z_{\theta,1:T} \right]$, 

$$\sqrt{N} \tilde{\pi}_{\theta} \left( \frac{1}{Z_{\theta,1:T}} \left[ \varphi(x_t) - \pi_{\theta}(\varphi(x_t), y_{1:t})/Z_{\theta,1:T} \right] \right) \Rightarrow A_{\theta}(\tilde{\varphi}).$$

We have $A_{\theta}(\tilde{\varphi}) \sim N(0, \sigma^2_{t,T}(\tilde{\varphi}))$ with 

$$\sigma^2_{t,T}(\tilde{\varphi}) = \sigma^2_{t-1,T}(\tilde{\varphi}_{t-1,T}) + \sigma^2_{t+1,T}(\tilde{\varphi}_{t+1,T})$$

and 

$$I_{g\tilde{\varphi}}(\tilde{x}_{t-1}, \tilde{x}_{t+1}) = \int_{R^d_x} g_{\theta}(y_t|x_t)\tilde{\varphi}(x_t)f_{\theta}(\tilde{x}_{t+1} | x_t)f_{\theta}(x_t | \tilde{x}_{t-1})dx_t.$$

3.2 Numerical comparison of $O(N)$ estimates

We now compare the variability of (3.1.4) to that of (2.7.1) and show that under certain settings and scenarios, (3.1.4) is preferred to (2.7.1); we do this via numerical illustration because the asymptotic variance expression in Theorem 3.1.1 is too difficult to evaluate analytically. We consider two example HMMs. The first is a simple linear Gaussian model [36, Section 4]:

$$X_0 \sim N(\mu_0, \Sigma_0)$$

$$X_{n+1} \mid (X_{1:n} = x_{1:n}, Y_{1:n} = y_{1:n}) \sim N(F x_n, Q) = f_{\theta}(x_{n+1} \mid x_n)$$

$$Y_n \mid (X_{1:n} = x_{1:n}, Y_{1:n-1} = y_{1:n-1}) \sim N(G x_n, R) = g_{\theta}(y_n \mid x_n)$$

$$G = (1, 0), \quad F = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad R = \tau^2, \quad Q = \nu^2 \begin{pmatrix} \frac{1}{3} & \frac{1}{2} \\ \frac{1}{2} & 1 \end{pmatrix}. \tag{3.2.1}$$
The second is a more challenging non-linear, non-Gaussian stochastic volatility model [88]:

\[ X_0 \sim \mathcal{N}(\mu_0, \Sigma_0) \]  

(3.2.2)

\[ X_{n+1} \mid (X_{1:n} = x_{1:n}, Y_{1:n} = y_{1:n}) \sim \mathcal{N}(x_n, \nu^2) = f_\theta(x_{n+1} \mid x_n) \]

\[ Y_n \mid (X_{1:n} = x_{1:n}, Y_{1:n-1} = y_{1:n-1}) \sim \mathcal{G}(1, \tau \cdot \exp(x_n)) = g_\theta(y_n \mid x_n). \]

For (3.2.1), Algorithm 7 is used to calculate \( \hat{Z}_{\theta, 1:T} \) via (2.7.1) and Algorithm 13 is used to calculate \( \hat{Z}_{\theta, 1:T} \) via (3.1.4). Two versions of Algorithm 13 are implemented:

- Type A: Only the predict equations of Algorithm 1 are used to determine the mean and covariance for \( \{\xi_{1:T, \theta}\} \). In other words, the observations do not influence the calculation of the pseudo-priors. This technique yields a sequence of pseudo-priors that are not ideal but are reflective of the information that might be available in an application of interest.

- Type B: The predict and the update equations of Algorithm 1 are used to determine the mean and covariance for the sequence of pseudo-priors (i.e., the observations do influence \( \{\xi_{1:T, \theta}\} \)). This technique yields a sequence of pseudo-priors that each take the form \( \pi_\theta(x_n \mid y_{1:n-1}) \). The pseudo-priors are optimal in the sense that they lead to incremental weights in Algorithm 11 that are of minimum variability (see Section 3.3). Note that these priors would generally not be available for many models.

For (3.2.2), we again compare (2.7.1) and (3.1.4) by running Algorithms 7 and 13. However, as the optimal pseudo-priors are not analytically available for this model, we only run the Type A version of Algorithm 13.

In each example, we calculate

\[
\log \left[ \mathbb{V} \left[ \hat{Z}_{\theta, 1:T} \right] \right] - \log \left[ \mathbb{V} \left[ \hat{Z}_{\theta, 1:T} \right] \right]
\]

for different pairs of values of the state noise \( \nu \) and the observation noise \( \tau \), where the variance is taken with respect to the appropriate algorithm. As Theorem 3.1.1 does not specify an optimal \( t \), we run two batches of experiments. In the first batch, \( t = T/2 \). In the second batch, \( t \) varies from one simulation to the next; it is sampled from a discrete
uniform on \((\frac{T}{4}, \frac{3T}{4})\) centred at \(T/2\).

### 3.2 Numerical comparison of \(\mathcal{O}(N)\) estimates

#### 3.2.1 Linear Gaussian model

For all simulations, the filters use the optimal importance densities and \(\beta\) resampling weights as defined in [36, Appendix A]:

\[
q_\theta \left( x_n \mid \bar{x}_{n-1}^{\beta}, y_n \right) \propto \mathcal{N} \left( \bar{x}_n \mid \mu_{n|n-1}, \Sigma_{n|n-1} \right)
\]
\[
q_\theta \left( x_n \mid \bar{x}_{n+1}^{\beta}, y_n \right) \propto \mathcal{N} \left( \bar{x}_n \mid \mu_{n|n+1}, \Sigma_{n|n+1} \right)
\]
\[
q_\theta \left( x_t \mid \bar{x}_{t-1}^{\beta}, \bar{x}_{t+1}^{\beta}, y_t \right) \propto \mathcal{N} \left( x_t \mid \mu_{t|T}, \Sigma_{t|T} \right)
\]

\[
\beta_{t+1}^{\beta} \propto \tilde{W}_{t+1}^{\beta} \mathcal{N} \left( y_t \mid GF \bar{x}_{t-1}^{\beta}, R + GG' \right)
\]
\[
\beta_t^{\beta} \propto \tilde{W}_{t+1}^{\beta} \mathcal{N} \left( y_t \mid G \left( \tilde{F}_t \bar{x}_{t+1} + \tilde{Q}_t \Sigma_{t}^{-1} \mu_t \right), R + G\tilde{Q}_t G' \right),
\]

where

\[
\tilde{F}_t = \Sigma_t F' \Sigma_{t+1}^{-1}
\]
\[
\tilde{Q}_t = \Sigma_t F' \Sigma_{t+1} Q F'^{-1}
\]
\[
\Sigma_{n|n-1} = \left( Q^{-1} + G' R^{-1} G \right)^{-1}
\]
\[
\mu_{n|n-1} = \Sigma_{n|n-1} \left( Q^{-1} F' \bar{x}_{n-1}^{\beta} + G' R^{-1} y_n \right)
\]
\[
\Sigma_{n|n+1} = \left( \Sigma_1^{-1} + G' R^{-1} G + F' Q^{-1} F \right)^{-1}
\]
\[
\mu_{n|n+1} = \Sigma_{n|n+1} \left( \Sigma_1^{-1} \mu_n + G' R^{-1} y_n + F' Q^{-1} \bar{x}_{n+1} \right)
\]
\[
\Sigma_{t|T} = \left( Q^{-1} + G' R^{-1} G + F' Q^{-1} F \right)^{-1}
\]
\[
\mu_{t|T} = \Sigma_{t|T} \left( Q^{-1} F' \bar{x}_{t-1}^{\beta} + G' R^{-1} y_t + F' Q^{-1} \bar{x}_{t+1}^{\beta} \right)
\]

and \(\Sigma_n\) and \(\mu_n\) are the variance and mean of the pseudo-prior, respectively, at time \(n\).

When \(t = T/2\), we consider values for \(\nu^2\) and \(\tau^2\) that range from one to 98 in steps of seven, and 50 simulations per pair \((\nu^2, \tau^2)\) are run. When \(t\) is allowed to vary, \(\nu^2\) and \(\tau^2\) range from one to 99 in steps of 33, and we run 300 simulations per pair of values. The experiment is repeated under different values of \(N\) and \(T\). Algorithms 7 and 13 always
require approximately the same computational time (approximately two seconds per run for the case where \( T = 100 \) and \( N = 200 \), using a Linux workstation with an Intel Core 2 Quad Q9550 CPU at 2.83 GHz).

We acknowledge that this experiment does not follow a full factorial design. As with other numerical analyses in this thesis, the simulations have quite a long running time (especially when one wants to run the same algorithm 50 or 300 times per algorithmic setting). In order to test a more broad range of scenarios in a timely fashion (with the computational power that is available to us), we skip some of the scenarios that one would have chosen in a full factorial design.

Results

- \( T = 100, N = 200, \) fixed \( t \): see Figure 3.2.1
- \( T = 200, N = 200, \) fixed \( t \): see Figure 3.2.1
- \( T = 300, N = 300, \) fixed \( t \): see Figure 3.2.1
- \( T = 100, N = 2000, \) fixed \( t \): see Figure 3.2.2
- \( T = 200, N = 2000, \) fixed \( t \): see Figure 3.2.2
- \( T = 300, N = 2000, \) fixed \( t \): see Figure 3.2.2
- \( T = 100, N = 1000, \) variable \( t \): see Figure 3.2.3
- \( T = 200, N = 1000, \) variable \( t \): see Figure 3.2.3
- \( T = 300, N = 1000, \) variable \( t \): see Figure 3.2.3

When the Type A version of Algorithm 13 runs, the variability of (2.7.1) is much lower than the variability of (3.1.4) in most cases, as the combination step of Algorithm 13 is a major source of variability (this is explained in the next paragraph). When the state noise is large relative to the observation noise, the combination step performs better and the variances of the two algorithms are about the same. The figures also illustrate that the results do not change significantly as \( T \) or \( N \) is increased.
3.2 Numerical comparison of $O(N)$ estimates

With the Type B version of Algorithm 13, we see that the two-filter smoother outperforms the forward filter by several orders of magnitude in most cases. One will also notice that the two-filter algorithm outperforms the forward filter even more as $T$ becomes larger, whereas increases in $N$ do not change the output. However, when the observation noise is large relative to the state noise, the forward filter algorithm provides the more consistent estimates. This phenomenon is likely a result of the combination step of the two-filter algorithm. When the hidden Markov process cannot vary widely relative to the observation noise, the chances of the forward and backward chains meeting naturally are small. We force them to meet when we propagate at the midpoint $t$, which is why the algorithm produces such variable results over several simulations. This variability becomes worse as $\tau^2$ is increased, as the observations provide even less information to guide the hidden process. The significance of these results is explained in Section 3.3.
Figure 3.2.1: Here we focus on small values for \( N \), with Type A simulations at left and Type B at right. We set \( T = 100 \) (top), 200 (middle), and 300 (bottom). Each block represents the average of 50 simulations. In the two-filter SMC algorithm, the filters meet at the fixed time point \( T/2 \).
3.2 Numerical comparison of $O(N)$ estimates

$\log\left[ V(\hat{\omega}_{\theta,1:T}) \right] - \log\left[ V(\hat{\omega}_{\theta,1:T}) \right]$
Figure 3.2.3: In the two-filter SMC algorithm, the filters now meet at a variable time point (with Type A simulations at left and Type B at right). We set $T = 100$ (top), 200 (middle), and 300 (bottom), with large values for $N$. Each block represents the average of 300 simulations.
3.2 Numerical comparison of $O(N)$ estimates

3.2.2 Stochastic volatility model

In this model, the optimal importance distributions and optimal $\beta$ resampling weights are analytically unavailable. Thus, Algorithms 7 and 11 propagate via the model’s Markov dynamics, and we use $\vec{\beta}_{l-1}^{(i)} = \vec{W}_{l-1}^{(i)}$ and $\vec{\beta}_{l+1}^{(j)} = \vec{W}_{l+1}^{(j)}$. When $t = T/2$, we consider values for $\tau$ that range from one to 98 in steps of seven, and we allow $\nu$ to range from 0.25 to seven in steps of 0.5. Fifty simulations are run per pair $(\nu, \tau)$. When $t$ varies, $\tau$ ranges from one to 99 in steps of 33 and $\nu$ ranges from 0.25 to 9 in steps of three. We run 300 simulations per pair $(\nu, \tau)$. The experiment is again repeated under different values of $N$ and $T$. Algorithms 7 and 13 always require approximately the same computational time (approximately two seconds per run for the case where $T = 100$ and $N = 200$).

Results

- $T = 100, N = 200$, fixed $t$: see Figure 3.2.4
- $T = 200, N = 200$, fixed $t$: see Figure 3.2.4
- $T = 300, N = 300$, fixed $t$: see Figure 3.2.4
- $T = 100, N = 2000$, fixed $t$: see Figure 3.2.4
- $T = 200, N = 2000$, fixed $t$: see Figure 3.2.4
- $T = 300, N = 2000$, fixed $t$: see Figure 3.2.4
- $T = 100, N = 1000$, variable $t$: results not pictured
- $T = 200, N = 1000$, variable $t$: results not pictured
- $T = 300, N = 1000$, variable $t$: results not pictured

The combination step of Algorithm 13 is still a source of variability. In all scenarios, the two-filter algorithm outperforms the forward filter algorithm as one increases the state noise relative to the observation noise (as in the linear Gaussian example of Section 3.2.1). Also, one can observe the increased variability of (3.1.4) when $T$ is increased and its decreased
variability when $N$ is increased. This trend can be traced back to the degeneracy of the backward filter. Recall that we use sub-optimal importance distributions in the forward and backward filters. Thus, changes to $T$ and $N$ have a significant impact on the filters’ degeneracy and variability. When we increase $T$, the backward filter’s effective sample size decreases more than that of the forward filter and the estimate (3.1.4) suffers. When we increase $N$, the backward filter’s degeneracy becomes less of an issue and the estimate (3.1.4) shows less variability. These results are in line with Remark 3.1.1.

Note that while we monitor the effective sample sizes of the two filters throughout the simulations, we do not report the values here. Also, as the results are the same regardless of whether $t$ is fixed or allowed to vary, we omit the plots for a variable $t$. 
3.2 Numerical comparison of $\mathcal{O}(N)$ estimates

\[
\log \left[ \mathcal{V} \left( \hat{Z}_{\theta,1:T}^{\text{Algo}^7} \right) \right] - \log \left[ \mathcal{V} \left( \hat{Z}_{\theta,1:T}^{\text{Algo}^{13}} \right) \right]
\]

![Heatmap images showing log (V) values for different parameter values.](image)

Figure 3.2.4: Left: small values for $N$. Right: large values for $N$. We set $T = 100$ (top), 200 (middle), and 300 (bottom). Each block represents the average of 50 simulations. In the two-filter SMC algorithm, the filters meet at the fixed time point $T/2$.

3.2.3 Primary findings

The simulations from Sections 3.2.1 and 3.2.2 demonstrate that the variability of (3.1.4) is really only significantly reduced in a broad range of scenarios when the pseudo-priors
3. Generalised Two-filter Smoothing and Particle Markov chain Monte Carlo

Each take the form $\pi_\theta(x_n \mid y_{1:n-1})$, and by running the Type A and Type B algorithms, we illustrated just how significant that boost in performance can be. Although the Type B algorithmic settings will generally not be available in a real application of interest, the findings still suggest that one should aim to use a pseudo-prior that approximates $\pi_\theta(x_n \mid y_{1:n-1})$ as closely as possible when trying to obtain low variance estimates of the HMM normalising constant.

3.3 Optimal pseudo-priors

We now take a closer look at the results of Section 3.2.1. Recall the sequence of targets (2.9.3) and (2.9.4) for Algorithm 11:

$$
\tilde{\pi}_\theta(x_{n:T} \mid y_{n:T}) = \frac{\xi_{n,\theta}(x_n) g_\theta(y_n \mid x_n) \prod_{k=n+1}^T g_\theta(y_k \mid x_k) f_\theta(x_k \mid x_{k-1})}{Z_{\theta,n:T}},
$$

(3.3.1)

for $n \in \{t, \ldots, T-1\}$. In the Type B simulations of Section 3.2.1, $\xi_{n,\theta}$ was set to equal $\pi_\theta(x_n \mid y_{1:n-1})$ for all $n \in \{t, \ldots, T\}$. With $\xi_{n,\theta} = \pi_\theta(x_n \mid y_{1:n-1})$, the expression (3.3.1) becomes

$$
\tilde{\pi}_\theta(x_{n:T} \mid y_{1:T}) = \frac{\pi_\theta(x_n \mid y_{1:n-1}) g_\theta(y_n \mid x_n) \prod_{k=n+1}^T g_\theta(y_k \mid x_k) f_\theta(x_k \mid x_{k-1})}{Z_{\theta,n:T}}
$$

(3.3.2)

for $n \in \{t, \ldots, T-1\}$, which is a sequence of targets that each condition on the full set of observations $y_{1:T}$. In other words, the backward filter becomes an $O(N)$ backward smoother. When we use $\xi_{n,\theta} = \pi_\theta(x_n \mid y_{1:n-1})$, the incremental weights of this backward smoother also have a low variance. At times $T - 1$ through $t$,

$$
\tilde{W}_n \propto \frac{\tilde{\pi}_\theta(x_{n:T} \mid y_{1:T})}{\tilde{\pi}_\theta(x_{n+1:T} \mid y_{1:T}) q_\theta(x_n \mid x_{n+1})} \frac{\pi_\theta(x_n \mid x_{n+1:T}, y_{1:T})}{\pi_\theta(x_n \mid x_{n+1:T}, y_{1:T}) q_\theta(x_n \mid x_{n+1})} = \frac{\pi_\theta(x_n \mid x_{n+1:T}, y_{1:T})}{q_\theta(x_n \mid x_{n+1})}.
$$

(3.3.3)

In our simulations, we set $q_\theta(x_n \mid x_{n+1}) = q_{\theta,\text{opt}}(x_n \mid x_{n+1}, y_n) = \pi_\theta(x_n \mid x_{n+1:T}, y_{1:T})$, thereby yielding incremental weights equal to one. These weights offer a significant boost.
3.4 Application of generalised two-filter smoothing in PMCMC

in performance over the forward filter, where the targets can differ significantly from one another from one time point to the next and the incremental weights are equal to \( \pi_\theta (y_n | x_{n-1}) \). The Type B two-filter algorithm performs so well because, essentially, it replaces the latter steps of a degenerate forward filter with a backward filter that does not degenerate with time. For this reason, the Type B two-filter algorithm outperforms the forward filter even more as \( T \) becomes larger.

With the optimal pseudo-priors available (they are optimal in the sense that they lead to incremental weights which are each equal to one and allow for exact sampling of the target), one could ignore the combination step of the two-filter algorithm and reduce the variability of the estimate of the marginal likelihood even further by only running the backward smoother from time \( T \) to time one. One could also perform backward smoothing with very low variance.

Our study of estimating the marginal likelihood via generalised two-filter smoothing now branches into two pursuits. In Sections 3.4 and 3.5, we apply our \( \mathcal{O}(N) \) unbiased estimate and the two-filter smoother in developing new PMCMC algorithms. In Sections 3.6 and 3.7, we use the newly discovered optimal pseudo-priors to explore some alternative smoothing algorithms.

3.4 Application of generalised two-filter smoothing in PMCMC

The previous sections show that the estimate (3.1.4) has a lower variance than (2.7.1) under the appropriately chosen settings. In Algorithm 22, one uses an estimate of the marginal likelihood to compute the acceptance probability. A better estimate of \( Z_{\theta,1:T} \) may lead to a PMMH that converges faster. Thus, we motivate the application of two-filter smoothing within PMMH (i.e., we aim to replace the forward filter used in Algorithm 22 with Algorithm 13). Also, as \( Z_{\theta,1:T} \) is computed via the SMC incremental weights, a better estimate implies that the incremental weights are also more precise. This latter point means that the generalised two-filter formula could potentially aid the performance of a PG sampler as well.

We now apply two-filter smoothing within PMCMC, for the purpose of estimating the static parameter \( \theta \in \Theta \subseteq \mathbb{R}^{d_\theta} \) when \( t \) is variable and \( T \) is fixed. For completeness, we first
present samplers that are of complexity $O(N^2)$ per iteration. We then present samplers that are of complexity $O(N)$.

### 3.4.1 Samplers of complexity $O(N^2)$

The goal is to introduce Algorithm 12 into Algorithms 22 and 24 in order to sample from

$$
\pi(\theta, x_{1:T} | y_{1:T}) \propto \pi(\theta) \frac{\gamma(\theta)(x_{1:T}, y_{1:T})}{Z_{\theta, 1:T}}. \tag{3.4.1}
$$

Just as in Section 2.18, we start by building an extended target $\pi^N$. Algorithm 12 uses forward and backward filters that meet at a time $t$. Thus, an auxiliary random variable $t$ is added to the state space of (3.4.1):

$$
\pi^N(\theta, t, x_{1:T} | y_{1:T}) \propto \pi_T(t) \pi(\theta) \frac{\gamma(\theta)(x_{1:T}, y_{1:T})}{Z_{\theta, 1:T}}, \tag{3.4.2}
$$

where $t$ is drawn from some arbitrary $\pi_T(t)$. Next, we already know from Section 2.18 that when one excludes the very final resampling step, the joint density of the variables simulated by the forward filter through time $(t - 1)$ will follow (2.18.1). The joint density of the variables simulated by the backward filter (Algorithm 11) will similarly be

$$
\psi(\omega_{1:T}, \omega_{t+1:T}) = \prod_{i=1}^{N} q(\omega_T) \prod_{n=1}^{t-1} r(\omega_{T-n+1} | \omega_{T-n}) \prod_{i=1}^{N} q(\omega_{T-n+1} | \omega_{T-n}), \tag{3.4.3}
$$

where $r(\omega_{T-n+1} | \omega_{T-n}) = \prod_{i=1}^{N} \omega_{T-n+1}^{a(i)}$. One can use (2.18.1), (3.4.2), and (3.4.3) to establish an extended target as

$$
\pi^N(\theta, t, x_{1:t-1}, k, \omega_{1:t-1}, k^T, \omega_{t:T}, k^T, \omega_{t+1:T} | y_{1:T}) = \frac{\pi_T(t) \pi(\theta, x_{1:t-1}, x^k_{t-1}, \omega_{T-n+1}^{a(k)} | y_{1:T})}{q(\omega_{1:t-1} | \omega_{1:t-2})} \times \frac{\psi(\omega_{1:t}, \omega_{t+1:T})}{q(\omega_{k,T} | \omega_{T-n+1}^{a(k)})} \times \frac{\psi(\omega_{k,T}, \omega_{t:1:T})}{q(\omega_{k,T} | \omega_{T-n+1}^{a(k)})}, \tag{3.4.4}
$$

where $r(\omega_{T-n+1} | \omega_{T-n}) = \prod_{i=1}^{N} \omega_{T-n+1}^{a(i)}$. One can use (2.18.1), (3.4.2), and (3.4.3) to establish an extended target as
3.4 Application of generalised two-filter smoothing in PMCMC

To keep the discussion as simple as possible, we limit $\pi_T(t)$ so $t$ cannot equal 1 or $T$.

The expression (3.4.4) looks somewhat complicated. To obtain a more usable form, we first replace the normalised weights with the un-normalised weights:

$$
\pi = \pi_T(t) \frac{1}{N^2} \pi(\theta, \tilde{x}^k_{t-1}, \tilde{x}^k_{t:T} \mid y_{1:T}) \left( \prod_{n=1}^{t-1} \frac{1}{N} \sum_{i=1}^{N} \tilde{W}_{n}^{i} \right) \frac{\psi_{\theta}(\tilde{x}^k_{1:t-1}, \tilde{a}^k_{1:t-2})}{q_{\theta}(\tilde{x}^k_{t-1})} \left( \prod_{n=1}^{t-1} \frac{1}{N} \sum_{i=1}^{N} \tilde{W}_{n}^{i} \right) \frac{\psi_{\theta}(\tilde{x}^k_{t}, \tilde{a}^k_{t})}{g_{\theta}(\tilde{x}^k_{t})} \left( \prod_{n=1}^{T-t} \frac{1}{N} \sum_{i=1}^{N} \tilde{W}_{n}^{i} \right) \frac{\psi_{\theta}(\tilde{x}^k_{T-n}, \tilde{a}^k_{T-n+1})}{q_{\theta}(\tilde{x}^k_{T-n})} \left( \prod_{n=1}^{T-t} \frac{1}{N} \sum_{i=1}^{N} \tilde{W}_{n}^{i} \right)
$$

Expanding $\pi(\theta, \tilde{x}^k_{t-1}, \tilde{x}^k_{t:T} \mid y_{1:T})$, expanding the products in the denominator, and simplifying gives us:

$$
= \frac{\pi(\theta)}{Z_{\theta,1:T}} \pi_T(t) \frac{1}{N^2} \cdot g_{\theta}(y_{t-1} \mid \tilde{x}^k_{t-1}) f_{\theta}(\tilde{x}^k_{t} \mid \tilde{x}^k_{t-1}) f_{\theta}(\tilde{x}^k_{t+1} \mid \tilde{x}^k_{t-1}) f_{\theta}(\tilde{x}^k_{t} \mid \tilde{x}^k_{t-1}) \cdot \xi_{t+1,\theta}(\tilde{x}^k_{t+1}) \left( \prod_{n=1}^{N} \frac{1}{N} \tilde{W}_{n}^{i} \right) \left[ \prod_{n=1}^{T-t} \frac{1}{N} \sum_{i=1}^{N} \tilde{W}_{n}^{i} \right] \left[ \prod_{n=1}^{T-t} \frac{1}{N} \sum_{i=1}^{N} \tilde{W}_{n}^{i} \right] \times
$$

Recall the definitions of the weights (2.8.2) and (2.9.7) to obtain the final form:

$$
= \frac{\pi(\theta)}{Z_{\theta,1:T}} \pi_T(t) \frac{1}{N^2} \tilde{Z}_{\theta,1:t-2} \tilde{Z}_{\theta,t+1:T} \frac{1}{N^2} \tilde{W}_{t-1}^{k} \tilde{W}_{t}^{k} \times
$$

Assuming that we can sample $\theta \in \Theta$ using the conditional density $q(\theta \mid \zeta)$, which is a density that conditions on a current accepted value $\zeta \in \Theta$, we can define a PMMH algorithm with proposal density

$$
q^{N}(\theta, t, \tilde{x}^k_{1:t-1}, \tilde{a}^k_{1:t-2}, \tilde{x}^k_{t:T}, \tilde{x}^k_{t+1:T}) \propto
$$
Chapter 3. Generalised Two-filter Smoothing and Particle Markov chain Monte Carlo

\[ q(\theta \mid \zeta) \pi_T(t) \psi_\theta(\overrightarrow{x}_{1:t-1}, \overrightarrow{d}_{1:t-2}) \psi_\theta(\overleftarrow{x}_{t:T}, \overleftarrow{a}_{t+1:T}) \overrightarrow{W}_t f_\theta(\overrightarrow{x}_t \mid \overrightarrow{x}_{t-1}) \overleftarrow{W}_t f_\theta(\overleftarrow{x}_t \mid \overleftarrow{x}_{t-1}) \]

as in Algorithm 26. The acceptance probability of the PMMH will be

\[ 1 \land \frac{\pi(\theta^*) q(\theta \mid \theta^*) \hat{Z}_{\theta^*,1:T}}{\pi(\theta) q(\theta^* \mid \theta) \hat{Z}_{\theta,1:T}}. \]

Particle Gibbs

Furthermore, all random variables can be sampled from their full conditionals defined by the target’s two forms: (3.4.4) and (3.4.5). Plus, given the prior \( \pi(\theta) \), assume that we can sample \( \theta \) using the conditional density

\[ \pi(\theta \mid x_{1:T}, y_{1:T}) \propto \pi(\theta) \prod_{n=1}^{T} g_\theta(y_n \mid x_n) f_\theta(x_n \mid x_{n-1}), \] (3.4.6)

which is invariant to changes to \( t \). A PG sampler can therefore be defined as in Algorithm 27. In Step 1 of the PG sampler, we make use of the fact that

\[ \pi_N(t \mid \cdots) = \pi_N(t \mid \theta, \overrightarrow{x}_{1:t-1}, \overleftarrow{x}_{t:T}, y_{1:T}) = \pi_T(t). \]

In other words, the target path \( x_{1:T} \) is invariant to changes in \( t \), and so the full conditional for \( t \) is the arbitrary density given to \( t \). We additionally implement the discussion point of [89], where we sample the individual elements of \( (\overrightarrow{d}_{1:t-1}, \overleftarrow{a}_{t:T}) \) via their full conditionals (as opposed to implicitly sampling the full paths via \( \overrightarrow{k} \) and \( \overleftarrow{k} \)). This sampling scheme aids in the exploration of all possible ancestries and helps the PG sampler to mix more quickly [20].

3.4.2 Samplers of complexity \( \mathcal{O}(N) \)

Algorithms 26 and 27 have a high computational cost. It would be preferable to develop analogues that are \( \mathcal{O}(N) \) in complexity. Assume again that we want to sample from (3.4.1), and sampling from (3.4.6) is feasible. We now aim to apply Algorithm 13 within the PMCMC framework.
Algorithm 26 $O(N^2)$ two-filter PMMH

- Step 0: Draw $\theta$ and $t$ from their priors. All remaining random variables can be sampled from their full conditionals defined by the target (3.4.5):
  - Sample $\x_1:t-1, \alpha_1:t-2 \sim \tilde{\psi}_\theta(\cdot)$ via Algorithm 7, excluding the final resampling step.
  - Sample $\x_{t:T}, \alpha_{t+1:T} \sim \tilde{\psi}_\theta(\cdot)$ via Algorithm 11, excluding the final resampling step.
  - Choose $k, \tilde{k}$ with probability proportional to $W_{t-1}^k W_{t}^\alpha f_\theta(\x_{k:t}, \alpha_{k:t}) \xi_t, \alpha_t(\x_{k:t})$.

Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta:1:T}$, via (3.1.2).

- Step 1: Sample $\theta^* \sim q(\cdot | \theta)$. All remaining random variables can be sampled from their full conditionals defined by the target (3.4.5):
  - Choose $t^* \sim \pi_T(\cdot)$.
  - Sample $\x_{1:t^*-1}, \alpha_{1:t^*-2} \sim \tilde{\psi}_{\theta^*}(\cdot)$ via Algorithm 7, excluding the final resampling step.
  - Sample $\x_{t^*:T}, \alpha_{t^*+1:T} \sim \tilde{\psi}_{\theta^*}(\cdot)$ via Algorithm 11, excluding the final resampling step.
  - Choose $k^*, \tilde{k}^*$ with probability proportional to $W_{t^*-1}^{k^*} W_{t^*}^{\alpha^*} f_{\theta^*}(\x_{k^*:t^*}, \alpha_{k^*:t^*}) \xi_{t^*}, \alpha_{t^*}(\x_{k^*:t^*})$.

Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta^*:1:T}$, via (3.1.2).

- Step 2: With acceptance probability

$$1 \wedge \frac{\pi(\theta^*) q(\theta | \theta^*)}{\pi(\theta) q(\theta^* | \theta)} \hat{Z}_{\theta^*:1:T} / \hat{Z}_{\theta:1:T}$$

set $\theta=\theta^*, t=t^*, \x_{1:t-1}=\x_{1:t^*-1}, \alpha_{1:t-2}=\alpha_{1:t^*-2}, \x_{t:T}=\x_{t^*:T}, \alpha_{t+1:T}=\alpha_{t^*+1:T}, k=k^*, \tilde{k}=\tilde{k}^*$.

Return to the beginning of Step 1.
Algorithm 27 $O(N^2)$ two-filter PG

- Step 0: Set $l = 0$. Set a path $x_{1:T}$ arbitrarily. Set its genealogy to $s_{1:T} = 1$.
- Step 1: Set $l = l + 1$. Choose $t^l \sim \pi_T(\cdot)$.
- Step 2: Sample $\theta^l | \cdots \sim \pi(\cdot | x_{1:T}, y_{1:T})$ via (3.4.6).
- Step 3: Sample
  
  $$\overleftarrow{W}_{1:t'-1} \mid \theta^l, x_{1:t'-1}, s_{1:t'-2} \propto \frac{\psi_{\theta^l}(\overleftarrow{x}_{1:t'-1}^{l}, \overleftarrow{a}_{1:t'-2}^{l})}{q_{\theta^l}(x_1)} \left(\prod_{n=2}^{t'-1} \frac{\overleftarrow{W}_{n-1}^{l,s}q_{\theta^l}(x_n | x_{n-1})}{\overleftarrow{W}_{n}^{l,s}q_{\theta^l}(x_n)}\right)$$

  via a conditional version of Algorithm 7, where $x_{1:t'-1}$ will survive all resampling steps.
- Step 4: Sample
  
  $$\overleftarrow{W}_{t':T} \mid \theta^l, x_{t':T}, s_{t'+1:T} \propto \frac{\psi_{\theta^l}(\overleftarrow{x}_{t':T}^{l}, \overleftarrow{a}_{t'+1:T}^{l})}{q_{\theta^l}(x_T)} \left(\prod_{n=1}^{T-t'-1} \frac{\overleftarrow{W}_{T-n+1}^{l,s}q_{\theta^l}(x_{T-n} | x_{T-n-1})}{\overleftarrow{W}_{T-n}^{l,s}q_{\theta^l}(x_{T-n})}\right)$$

  via a conditional version of Algorithm 11, where $x_{t':T}$ will survive all resampling steps.
- Step 5: Sample $k^l_1, k^l \propto \overleftarrow{W}_{t'-1}^{l,k^l} \overleftarrow{W}_{t}^{l,k^l} f_{\theta^l}(\overleftarrow{x}_{t'}^{l,k^l} | \overleftarrow{x}_{t}^{l,k^l})$, and set $s_{t-1:t} = \overleftarrow{k}_1^l, \overleftarrow{k}_t^l$.
- Step 6: Sample the ancestral lineage $s_{1:t'-2}$ for a selected path $\overleftarrow{x}_{1:t'-1}^{l,k^l}$ backward in time via
  
  $$\overleftarrow{a}_{n}^{l,k^l} \propto \overleftarrow{W}_{n}^{l,k^l} f_{\theta^l}(\overleftarrow{x}_{n+1}^{l,k^l} | \overleftarrow{x}_{n}^{l,k^l}),$$

  and record the path $x_{1:t'-1} = \overleftarrow{x}_{1:t'-1}^{l,k^l}$ that will survive all resampling steps at the next iteration of the algorithm.
- Step 7: Sample the ancestral lineage $s_{t'+1:T}$ for a selected path $\overleftarrow{x}_{t':T}^{l,k^l}$ forward in time via
  
  $$\overleftarrow{a}_{n}^{l,k^l} \propto \frac{f_{\theta^l}(\overleftarrow{x}_{n}^{l,k^l} | \overleftarrow{x}_{n-1}^{l,k^l}) g_{\theta^l}(y_n | \overleftarrow{x}_{n}^{l,k^l}) f_{\theta^l}(\overleftarrow{x}_{n+1}^{l,k^l} | \overleftarrow{x}_{n}^{l,k^l})}{q_{\theta^l}(\overleftarrow{x}_{n}^{l,k^l} | \overleftarrow{x}_{n+1}^{l,k^l})},$$

  and record the path $x_{t':T} = \overleftarrow{x}_{t':T}^{l,k^l}$ that will survive all resampling steps at the next iteration of the algorithm.

Return to the beginning of Step 1.
To develop an appropriate extended target, it is necessary to introduce some extra random variables that were not needed in (3.4.4) (due to the simulation techniques employed in Algorithm 13). Namely, in order to resample a single path that is drawn via Algorithm 13, we need to introduce $\hat{i}$ to denote the indices of the $N$ resampled forward particles, $\hat{j}$ to denote the indices of the $N$ resampled backward particles, and $u$ to denote the index of a single particle $x_i^u$. Combining (2.18.1), (3.4.2), and (3.4.3), a target probability density is defined as

$$
\pi^N(\theta, t, \tilde{x}^c_{1:t-1}, \tilde{x}^c_{1:t-2}, \tilde{x}^{d}_{t+1:T}, \tilde{x}^{d}_{t+2:T}, x_t, u, \hat{i}, \hat{j}) = \pi_T(t) \prod_{l=1}^N \frac{\tilde{\psi}_\theta(x_l^c \mid x_{l-1}^c, x_{l+1}^c) \prod_{n=2}^{T-1} w_{n-1}^{a(c)} q_\theta(x_n^c \mid x_{n-1}^{a(c)})}{q_\theta(x_{l-1}^c \mid x_{l+1}^c, x_{l+2}^c)} \frac{\tilde{\psi}_\theta(x_{l+1}^d \mid x_{l-1}^d, x_{l+2}^d) \prod_{n=1}^{T-t} w_{n+1}^{a(d)} q_\theta(x_n^d \mid x_{n-1}^{a(d)})}{q_\theta(x_t^d \mid x_{t-1}^d, x_{t+1}^d)} \prod_{l \neq u} \pi(i_{l-1}^{u-1}, u+1) \pi(i_{l+1}^{u+1}, u) \prod_{l \neq u} q_\theta(x_l^f \mid x_{l-1}^f, x_{l+1}^f),
$$

where $u$ is the index of the particle chosen at time $t$, $c = i(u)$, and $d = j(u)$. Furthermore, the probability of selecting $\hat{i}$ and $\hat{j}$ via their full conditional will be

$$
\pi(\hat{i}, \hat{j} \mid \cdots) = \frac{1}{N^2} \prod_{l \neq u} \beta_{l-1}^{j(l)} \beta_{l+1}^{j(l)},
$$

where $\frac{1}{N^2}$ is the probability of selecting $i(u)$ and $j(u)$.

A PMMH algorithm that targets (3.4.7) with the proposal density

$$
q^N(\theta, t, \tilde{x}^c_{1:t-1}, \tilde{x}^c_{1:t-2}, \tilde{x}^{d}_{t+1:T}, \tilde{x}^{d}_{t+2:T}, x_t, u, \hat{i}, \hat{j}) \propto q(\theta \mid \zeta) \pi_T(t) \tilde{\psi}_\theta(\tilde{x}^c_{1:t-1}, \tilde{x}^c_{1:t-2}) \tilde{\psi}_\theta(\tilde{x}^{d}_{t+1:T}, \tilde{x}^{d}_{t+2:T}) \times \left[ \prod_{l=1}^N \beta_{l-1}^{j(l)} \beta_{l+1}^{j(l)} q_\theta(x_l^f \mid x_{l-1}^f, x_{l+1}^f) \right] \tilde{W}_l^{j(u)} \tilde{W}_{l+1}^{j(u)} W_t^{u}
$$

(3.4.8)
Chapter 3. Generalised Two-filter Smoothing and Particle Markov chain Monte Carlo

is given as Algorithm 28. All random variables can be sampled from their full conditionals defined by the target (3.4.7), and so the PG sampler follows (see Algorithm 29).

Algorithms 28 and 29 could potentially outperform the $O(N^2)$ algorithms presented in Section 3.4.1, as the two $O(N)$ methods can complete more iterations within the same computational time frame. However, as the results of Section 3.2 indicate, the variance of Algorithms 28 and 29 may be high if the optimal pseudo-priors (or good approximations of the optimal pseudo-priors) are unavailable; in this case, the $O(N^2)$ algorithms may actually exhibit the superior performance. In the next section, we will test Algorithms 28 and 29 to illustrate the significant effects that the choice of pseudo-prior can have on the $O(N)$ schemes.

3.5 Implementation of the $O(N)$ PMMH and PG algorithms

We use the HMMs from Section 3.2 to compare Algorithms 28 and 29 to the standard PMCMC algorithms from Section 2.18. For the linear Gaussian model (3.2.1), we compare our $O(N)$ PG sampler to Algorithm 24. For the stochastic volatility model (3.2.2), we compare our $O(N)$ PMMH algorithm to Algorithm 22.

In the linear Gaussian example, the algorithms are organised as follows:

- Algorithm A is Algorithm 29 with $\xi_{n,\theta}(x_n) = \pi_\theta(x_n)$.
- Algorithm B is Algorithm 29 with $\xi_{n,\theta}(x_n) = \pi_\theta(x_n | y_{1:n-1})$.
- Algorithm C is Algorithm 24.

A similar organisation scheme is used for the stochastic volatility model, only Algorithm 29 is replaced with Algorithm 28 and Algorithm 24 is replaced with Algorithm 22. Note that the optimal pseudo-prior $\pi_\theta(x_n | y_{1:n-1})$ is analytically unavailable for model 3.2.2, and so we cannot run an Algorithm B for the stochastic volatility example.

In each trial, the aim is to compare the convergence of Algorithms A and/or B to that of Algorithm C as we attempt to infer $\tau$ and $\nu$ for different datasets. Across several trials, we vary the number of observations $T$, the number of particles $N$, and the number of iterations $M$ of each algorithm. We also vary any necessary prior parameters (see below).
3.5 Implementation of the $O(N)$ PMMH and PG algorithms

Algorithm 28 $O(N)$ two-filter PMMH

- Step 0: Draw $\theta$ and $t$ from their priors, and
  - Sample $\overset{\rightarrow}{x}_{t_1:t-1}, \overset{\rightarrow}{y}_{t_1:t-2} | \cdots \sim \overset{\rightarrow}{\psi}_\theta(\cdot)$ via Algorithm 7, excluding the final resampling step.
  - Sample $\overset{\rightarrow}{x}_{t+1:T}, \overset{\rightarrow}{y}_{t+2:T} | \cdots \sim \overset{\rightarrow}{\psi}_\theta(\cdot)$ via Algorithm 11, excluding the final resampling step.
- For $h \in \{1, \ldots, N\}$, sample $i(h) \propto \frac{\beta_{i(h)-1}}{\beta_{i(h)}+1}$ and sample $j(h) \propto \frac{\beta_{j(h)+1}}{\beta_{j(h)-1}}$.
- For $h \in \{1, \ldots, N\}$, sample $x_{t_1:h} \sim q_\theta(x_{t_1:h} | \overset{\rightarrow}{x}_{t_1-1}, \overset{\rightarrow}{x}_{t_1+1})$.
- Choose $u, i(u), j(u)$ with probability proportional to
  \[
  \frac{\overset{\rightarrow}{W}_{t-1}^{z(u)}\overset{\rightarrow}{W}_{t+1}^{j(u)}f_\theta(x_{t_1}^u | \overset{\rightarrow}{x}_{t_1-1}^u)f_\theta(\overset{\rightarrow}{x}_{t_1+1}^j | x_{t_1}^u)}{\xi_{t+1,\theta}(\overset{\rightarrow}{x}_{t+1}^j)}g_\theta(y_t | x_{t_1}^u).
  \]

Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta,1:T}$, via (3.1.4).

- Step 1: Sample $\theta^* \sim q_\cdot(\cdot | \theta)$, and
  - Choose $t^* \sim \pi_T(\cdot)$.
  - Sample $\overset{\rightarrow}{x}_{t_1:t^*-1}, \overset{\rightarrow}{y}_{t_1:t^*-2} | \cdots \sim \overset{\rightarrow}{\psi}_{\theta^*}(\cdot)$ via Algorithm 7, excluding the final resampling step.
  - Sample $\overset{\rightarrow}{x}_{t^*+1:T}, \overset{\rightarrow}{y}_{t^*+2:T} | \cdots \sim \overset{\rightarrow}{\psi}_{\theta^*}(\cdot)$ via Algorithm 11, excluding the final resampling step.
- For $h \in \{1, \ldots, N\}$, sample $i(h)^* \propto \frac{\beta_{i(h)^*-1}}{\beta_{i(h)^*}+1}$ and sample $j(h)^* \propto \frac{\beta_{j(h)^*+1}}{\beta_{j(h)^*-1}}$.
- For $h \in \{1, \ldots, N\}$, sample $x_{t_1:h}^* \sim q_{\theta^*}(x_{t_1:h}^* | \overset{\rightarrow}{x}_{t_1-1}^*, \overset{\rightarrow}{x}_{t_1+1}^*)$.
- Choose $u^*, i(u^*), j(u^*)$ with probability proportional to
  \[
  \frac{\overset{\rightarrow}{W}_{t-1}^{z(u^*)}\overset{\rightarrow}{W}_{t^*+1}^{j(u^*)}f_\theta(x_{t_1}^{u^*} | \overset{\rightarrow}{x}_{t_1-1}^{u^*})f_\theta(\overset{\rightarrow}{x}_{t_1+1}^{j(u^*)} | x_{t_1}^{u^*})}{\xi_{t^*+1,\theta}(\overset{\rightarrow}{x}_{t^*+1}^{j(u^*)})}g_\theta(y_t | x_{t_1}^{u^*}).
  \]

Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta^*,1:T}$, via (3.1.4).

- Step 2: With acceptance probability
  \[
  \frac{1}{\pi(\theta^*) q(\theta^* | \theta^*) \hat{Z}_{\theta^*,1:T}} \frac{\pi(\theta) q(\theta | \theta^*) \hat{Z}_{\theta,1:T}}{\hat{Z}_{\theta^*,1:T} \hat{Z}_{\theta,1:T}}.
  \]
  set $\theta=\theta^*, t=t^*, \overset{\rightarrow}{x}_{t_1:t-1}^{*} = \overset{\rightarrow}{x}_{t_1:t-1}, \overset{\rightarrow}{x}_{t_1:t-2}^{*} = \overset{\rightarrow}{x}_{t_1:t-2}, \overset{\rightarrow}{x}_{t+1:T}^{*} = \overset{\rightarrow}{x}_{t+1:T},$
  $\overset{\rightarrow}{x}_{t+2:T} = \overset{\rightarrow}{x}_{t+2:T}, u = u^*, i = i^*, j = j^*$.
  Return to the beginning of Step 1.
Chapter 3. Generalised Two-filter Smoothing and Particle Markov chain Monte Carlo

Algorithm 29 $O(N)$ two-filter PG

- Step 0: Set $l = 0$. Set a path $x_{1:T}$ arbitrarily. Set its genealogy to $s_{1:T} = 1$.

- Step 1: Set $l = l + 1$. Choose $t^l \sim \pi_T(.)$.

- Step 2: Sample $\theta^l \sim \pi(\cdot | x_{1:T}, y_{1:T})$ via (3.4.6).

- Step 3: Sample

$$\frac{\tilde{\omega}^l}{\tilde{\omega}^l_{1:t^l-1}} | \theta^l, x_{1:t^l-1}, s_{1:t^l-2} \propto \frac{\tilde{\psi}^l \theta^l (\tilde{\omega}^l_{1:t^l-1}, \tilde{\omega}^l_{1:t^l-2})}{q_{\theta^l}(x_1)} \left( \frac{\prod_{n=2}^{t^l-1} \tilde{w}_n^{l,s} q_{\theta^l}(x_n | x_{n-1})}{\prod_{n=1}^{T-t^l+1} \tilde{w}_{T-n+1}^{l,s} q_{\theta^l}(x_{T-n} | x_{T-n+1})} \right)$$

via a conditional version of Algorithm 7, where $x_{1:t^l-1}$ will survive all resampling steps.

- Step 4: Sample

$$\frac{\tilde{\omega}^l}{\tilde{\omega}^l_{t^l+1:T}} | \theta^l, x_{t^l+1:T}, s_{t^l+2:T} \propto \frac{\tilde{\psi}^l \theta^l (\tilde{\omega}^l_{t^l+1:T}, \tilde{\omega}^l_{t^l+2:T})}{q_{\theta^l}(x_T)} \left( \frac{\prod_{n=1}^{T-t^l} \tilde{w}_n^{l,s} q_{\theta^l}(x_{T-n} | x_{T-n+1})}{\prod_{n=1}^{T-t^l+1} \tilde{w}_{T-n+1}^{l,s} q_{\theta^l}(x_{T-n} | x_{T-n+1})} \right)$$

via a conditional version of Algorithm 11, where $x_{t^l+1:T}$ will survive all resampling steps.

- Step 5: For $h \in \{1, \ldots, N\}$, sample $i(h)^l \propto \tilde{\beta}^{i(h)^l}_{t^l-1}$ and sample $j(h)^l \propto \tilde{\beta}^{j(h)^l}_{t^l+1}$.

- Step 6: For $h \in \{1, \ldots, N\}$, sample $x_{t^l+1}^{l,h} \sim q_{\theta^l}(\cdot | x_{t^l+1}^{l,h}, x_{t^l+1}^{l,j(h)^l})$.

- Step 7: Choose $u^l, i(u^l)^l, j(u^l)^l$ with probability proportional to

$$\frac{W_{t^l-1}^{l,i(u^l)^l} W_{t^l+1}^{l,j(u^l)^l} f_{\theta^l}(x_{t^l+1}^{l,u^l} | x_{t^l-1}^{l,i(u^l)^l}) f_{\theta^l}(x_{t^l+1}^{l,j(u^l)^l} | x_{t^l+1}^{l,u^l}) \xi_{t^l+1,\theta^l}(x_{t^l+1}^{l,u^l})}{\xi_{t^l+1,\theta^l}(x_{t^l+1}^{l,i(u^l)^l})}$$

and set $s_{t^l-1:t^l+1} = i(u^l)^l, u^l, j(u^l)^l$, respectively. Set $x_{t^l} = x_{t^l+1}^{l,u^l}$.

- Step 8: Sample the ancestral lineage $s_{1:t^l-2}$ for a selected path $\tilde{x}^{l,i(u^l)^l}_{1:t^l-1}$ backward in time via

$$\tilde{a}^{l,u^l}_n \propto W_{t^l+1}^{l,i(u^l)^l} f_{\theta^l}(x_{t^l+1}^{l,u^l} | x_{t^l+1}^{l,j(u^l)^l}) g_{\theta^l}(y_n | x_{t^l+1}^{l,u^l}) q_{\theta^l}(x_{t^l+1}^{l,u^l} | x_{t^l+1}^{l,j(u^l)^l})$$

and record the path $x_{1:t^l-1} = \tilde{x}^{l,i(u^l)^l}_{1:t^l-1}$.

- Step 9: Sample the ancestral lineage $s_{t^l+2:T}$ for a selected path $\tilde{x}_{t^l+1:T}$ forward in time via

$$\tilde{a}^{l,u^l}_n \propto f_{\theta^l}(x_{t^l+1}^{l,u^l} | x_{t^l+1}^{l,j(u^l)^l}) g_{\theta^l}(y_n | x_{t^l+1}^{l,u^l}) q_{\theta^l}(x_{t^l+1}^{l,u^l} | x_{t^l+1}^{l,j(u^l)^l})$$

and record the path $x_{t^l+1:T} = \tilde{x}_{t^l+1:T}^{l,j(u^l)^l}$.

Return to the beginning of Step 1.
We are trying to explore and find those scenarios where one might see a striking difference in performance between the algorithms. Thus, for the sake of running as many trials as possible in a timely fashion, we only repeat five simulations of each algorithm per trial (with the random number generator seeded differently for each simulation and with the algorithms initialised by sampling random variables from their appropriate priors). All algorithms have approximately equal running times (approximately 30 hours per run for the case where $T = 1000$, $N = 100$, and $M = 10000$, using a Linux workstation with an Intel Core 2 Quad Q9550 CPU at 2.83 GHz).

To track convergence, we produce a set of three graphs per algorithm per trial, following the ideas from [11]:

- **Graph i**: This figure monitors the convergence of the observation and state noises together. $\hat{R}_p$ is defined in [11] and is essentially a scalar value measuring the relative sizes of $\hat{V}$ and $W$. One would like to see $\hat{R}_p$ approach one. The variables $\hat{V}$, $W$, and $B/n$ are also defined in [11], and they are the posterior variance-covariance matrix, the within-sequence covariance matrix, and the between-sequence covariance matrix, respectively. One would like to see the determinants of $W$ and $B/n$ stabilise.

- **Graphs ii and iii**: These figures monitor the convergence of the observation and state noises separately. $\hat{R}$, $\hat{V}$ and $W$ are defined in [11] and take the same meanings as in Graph ii. We would like to see $\hat{R}$ approach one, to show that each of the simulations is close to the target. We would also like to see $\sqrt{\hat{V}}$ and $\sqrt{W}$ stabilise at the same value.

### 3.5.1 Linear Gaussian model

Per trial, we generate sets of observations $y_{1:T}$ from an HMM. Further details of the trials follow below. Under each of the following subsections, we present the output per trial and explain how the graphs are to be read. We explain the meaning of these results in the context of our research goals in Section 3.5.3; that section also summarises the results of the simulations performed on the stochastic volatility HMM, which are presented in Section 3.5.2 below.
Chapter 3. Generalised Two-filter Smoothing and Particle Markov chain Monte Carlo

Results: trial 1

The generated data has a length of $T = 1000$, with $\tau^2 = 2000$ and $\nu^2 = 2000$. All algorithms use $\frac{1}{\tau^2} \sim \mathcal{G}(2, 0.001)$ and $\frac{1}{\nu^2} \sim \mathcal{G}(2, 0.0003)$.

- Algorithm A, $N = 100$, $M = 10000$: see Figure 3.5.1
- Algorithm B, $N = 100$, $M = 10000$: see Figure 3.5.1
- Algorithm C, $N = 100$, $M = 10000$: see Figure 3.5.1

We run this trial to see what happens when $\nu^2 = \tau^2$; this is a scenario under which the Type B two-filter SMC performed better than the forward-filter SMC in Section 3.2.1. We find that Algorithms A and C do not appear to converge within 10000 iterations. This is immediately obvious from the fact that the plots of $\hat{R}^p$ and $\hat{R}$ do not stabilise at one. Algorithm B seems to converge after approximately 4000 iterations. The determinants of $W$ and $B/n$ stabilise, and $\sqrt{V}$ and $\sqrt{W}$ stabilise at the same values. The inferred values for $\nu$ and $\tau$ which are sampled via Algorithm B also appear to be accurate (output not shown).
Figure 3.5.1: These graphs measure the convergence of the PG algorithm (Type A in top row, Type B in middle row, and Type C in bottom row). The minimum value on the vertical axis of each $\hat{R}_\nu$ and $\hat{R}$ graph is one, and the horizontal axis on all graphs is “Iteration number (including burn-in”).

Results: trial 2

The generated data has a length of $T = 500$, with $\tau^2 = 1000$ and $\nu^2 = 3000$. All algorithms use $\tau^2 \sim Ga(2, 0.001)$ and $\nu^2 \sim Ga(2, 0.000003)$.

- Algorithm A, $N = 500$, $M = 5000$: see Figure 3.5.2
- Algorithm B, $N = 500$, $M = 5000$: see Figure 3.5.2
- Algorithm C, $N = 500$, $M = 5000$: see Figure 3.5.2

In this trial, the goal is to try to boost the performance of Algorithm A. Relying on the results from Section 3.2.1, we set the state noise to be much greater than the observation noise. We also set $N = T$ to try to improve the performance of Algorithm C.
Algorithm B converges after about 2500 iterations and provides an accurate inference (sampled $\tau^2$ and $\nu^2$ values are not shown). However, Algorithms A and C have some trouble. Regarding Algorithm A, the graph of the determinant of $W$ does not seem to stabilise, and the graphs of $\sqrt{V}$ and $\sqrt{W}$ in the plot for $\tau^2$ do not stabilise at the same value. Regarding Algorithm C, the graph of the determinant of $W$ does not seem to stabilise.

Figure 3.5.2: These graphs measure the convergence of the PG algorithm (Type A in top row, Type B in middle row, and Type C in bottom row). The minimum value on the vertical axis of each $\hat{R}_p$ and $\hat{R}$ graph is one, and the horizontal axis on all graphs is “Iteration number (including burn-in)”.

Results: trial 3

The generated data has a length of $T = 500$, with $\tau^2 = 4000$ and $\nu^2 = 1000$. All algorithms use $\frac{1}{\tau^2} \sim \mathcal{G}(2, 0.00025)$ and $\frac{1}{\nu^2} \sim \mathcal{G}(2, 0.000125)$.

- Algorithm A, $N = 500$, $M = 5000$: see Figure 3.5.3
3.5 Implementation of the $O(N)$ PMMH and PG algorithms

- Algorithm B, $N = 500$, $M = 5000$: see Figure 3.5.3
- Algorithm C, $N = 500$, $M = 5000$: see Figure 3.5.3

Here, we set the variance of the observations to be relatively high. Algorithm A does not seem to converge, as the graph of the determinant of $W$ does not stabilise. Algorithm B also has some trouble converging here. The graph of the determinant of $W$ does not seem to stabilise, and the graphs of $\sqrt{V}$ and $\sqrt{W}$ in the plot for $\tau^2$ do not stabilise. Algorithm C has some trouble converging as well. The graphs of $\hat{R}^p$ and $\hat{R}$ appear to be a little jumpy.

![Graphs of Convergence of $\tau^2$ and $\nu^2$, $\tau^2$, and $\nu^2$.](image)

Figure 3.5.3: These graphs measure the convergence of the PG algorithm (Type A in top row, Type B in middle row, and Type C in bottom row). The minimum value on the vertical axis of each $\hat{R}^p$ and $\hat{R}$ graph is one, and the horizontal axis on all graphs is “Iteration number (including burn-in)”.

![Graphs of Convergence of $\tau^2$ and $\nu^2$, $\tau^2$, and $\nu^2$.](image)
3.5.2 Stochastic volatility model

In the previous section, we only considered synthetic data. We will now run the algorithms against actual datasets. We choose three different datasets consisting of S&P 500 daily closing values to serve as our direct observations. The first dataset (3rd January 2012 through 7th August 2012, \( T = 150 \) trading days) represents a time of relatively low market volatility. The second dataset (1st May 2008 through 1st December 2009, \( T = 400 \) trading days) represents a time of relatively high market volatility. We also choose this dataset for its larger value of \( T \). The third dataset (3rd January 2000 through 31st December 2001, \( T = 500 \) trading days) is chosen for its even larger value of \( T \). Also, the market volatility during this third period seems to be higher than that of the first dataset and lower than that of the second dataset.

To sample \( \theta \), a log normal random walk of the form \( \log(\theta^*) = \log(\theta) + \epsilon, \epsilon \sim \mathcal{N}(0, 0.005) \) is used. Further details of the trials follow below. We remind the reader that, under each of the following subsections, we present the output per trial and explain how the graphs are to be read. We explain the meaning of these results in the context of our research goals in Section 3.5.3.

Results: trial 1

The data consists of daily S&P 500 closing values from 3rd January 2012 through 7th August 2012. For all algorithms, the priors \( \tau \sim \mathcal{G}(2, 2) \) and \( \nu \sim \mathcal{G}(2, 2) \) are used.

- Algorithm A, \( N = 100, M = 40000 \): see Figures 3.5.4 and 3.5.5
- Algorithm C, \( N = 100, M = 40000 \): see Figure 3.5.4
- Algorithm A, \( N = 1000, M = 50000 \): results not pictured
- Algorithm C, \( N = 1000, M = 50000 \): see Figure 3.5.4

Algorithm A seems to converge regardless of the value of \( N \), whereas Algorithm C seems to have some trouble. The graphs of \( \sqrt{V} \) and \( \sqrt{W} \) do not stabilise at the same values for any value of \( N \), and the graph of \( \hat{R}_p \) does not reach a value of one when \( N = 100 \). The graphs of the determinant of \( W \) do not seem to stabilise either, for both values of \( N \).
3.5 Implementation of the $O(N)$ PMMH and PG algorithms

Figure 3.5.4: These graphs measure the convergence of the PMMH algorithm (Type A with $N = 100$ in top row, Type C with $N = 100$ in middle row, and Type C with $N = 1000$ in bottom row). The minimum value on the vertical axis of each $\hat{R}_\nu$ and $\hat{R}$ graph is one, and the horizontal axis on all graphs is “Iteration number $\times 10^4$ (including burn-in)”. As the output for the Type A simulations does not vary with $N$, we omit the plots for $N = 1000$. 
Figure 3.5.5: Here we present the inferred values for $\tau$ and $\nu$ (on the log scale), as sampled via Algorithm A with $N = 100$.

Results: trial 2

The data consists of daily S&P 500 closing values from 1st May 2008 through 1st December 2009. For all algorithms, the priors $\tau \sim \mathcal{G}(2, 2)$ and $\nu \sim \mathcal{G}(2, 2)$ are used.

- Algorithm A, $N = 100$, $M = 23000$: see Figures 3.5.6 and 3.5.7
- Algorithm C, $N = 100$, $M = 23000$: see Figure 3.5.6
- Algorithm A, $N = 1000$, $M = 23000$: see Figure 3.5.6
- Algorithm C, $N = 1000$, $M = 23000$: results not pictured

Algorithm C seems to converge for both values of $N$. Algorithm A appears to have some trouble, as the graphs of $\sqrt{\hat{V}}$ and $\sqrt{\hat{W}}$ for $\tau$ do not stabilise at the same values (for both values of $N$).
3.5 Implementation of the \( O(N) \) PMMH and PG algorithms

Figure 3.5.6: These graphs measure the convergence of the PMMH algorithm (Type A with \( N = 100 \) in top row, Type C with \( N = 100 \) in middle row, and Type A with \( N = 1000 \) in bottom row). The minimum value on the vertical axis of each \( \hat{R}\nu \) and \( \hat{R} \) graph is one, and the horizontal axis on all graphs is “Iteration number \( \times 10^4 \) (including burn-in)”. As the output for the Type C simulations does not vary with \( N \), we omit the plots for \( N = 1000 \).
Figure 3.5.7: Here we present the inferred values for $\tau$ and $\nu$ (on the log scale), as sampled via Algorithm C with $N = 100$.

Results: trial 3

The data consists of daily S&P 500 closing values from 3rd January 2000 through 31st December 2001. For all algorithms, the priors $\tau \sim \text{Ga}(2,2)$ and $\nu \sim \text{Ga}(2,2)$ are used.

- Algorithm A, $N = 100$, $M = 20000$: see Figure 3.5.8
- Algorithm C, $N = 100$, $M = 20000$: see Figure 3.5.8
- Algorithm A, $N = 1000$, $M = 20000$: see Figures 3.5.8 and 3.5.9
- Algorithm C, $N = 1000$, $M = 20000$: see Figures 3.5.8 and 3.5.9

Algorithm C converges for the larger value of $N$, but there is evidence that more iterations may be required for the smaller value of $N$. The graphs of $\hat{R}$ do not reach a value of one, and the graphs of $\sqrt{\hat{V}}$ and $\sqrt{W}$ do not stabilise at the same values. Algorithm A also converges for $N = 1000$, but there is evidence that more iterations (or more simulations) may be required for the $N = 100$ case. The graphs of $\sqrt{\hat{V}}$ and $\sqrt{W}$ do not stabilise, and the graph of the determinant of $W$ is thrown off by a single simulation around iteration 12000.
3.5 Implementation of the $O(N)$ PMMH and PG algorithms

Figure 3.5.8: These graphs measure the convergence of the PMMH algorithm (Type A with $N = 100$ in top row, Type C with $N = 100$ in second row, Type A with $N = 1000$ in third row, and Type C with $N = 1000$ in bottom row). The minimum value on the vertical axis of each $\hat{R}$ and $\hat{R}$ graph is one, and the horizontal axis on all graphs is “Iteration number \times 10^4 (including burn-in)”. 
Figure 3.5.9: Here we present the inferred values for $\tau$ and $\nu$ (on the log scale), as sampled via Algorithm A (top) and Algorithm C (bottom) with $N = 1000$.

### 3.5.3 Primary findings

Similar to Section 3.2, the results of Sections 3.5.1 and 3.5.2 show that the main benefits to using our $O(N)$ two-filter PMCMC algorithms are only realised when the optimal pseudoprior $\xi_{n,\theta}(x_n) = \pi_{\theta}(x_n | y_{1:n-1})$ is used:

- For the linear Gaussian example, Algorithm B usually converged faster than Algorithms A and C. This superior performance can be attributed to the low variance incremental backward weights yielded by the optimal version of Algorithm 13, which
3.6 New smoothing algorithms

was employed within Algorithm B.

- In the stochastic volatility example, little to no benefit is seen when one uses Algorithm A over Algorithm C. This finding gives further evidence that the estimator (3.1.4) does not benefit PMCMC unless one has access to the optimal pseudo-priors. Thus, we would recommend that one should only use our \( \mathcal{O} (N) \) two-filter PMCMC algorithms when (at least a very good approximation of) \( \pi_\theta (x_n \mid y_{1:n-1}) \) is analytically available, which implies that our PMCMC algorithms will have a limited applicability. Almost every real world application of interest will look more like the stochastic volatility example of Section 3.5.2 than it will resemble the linear Gaussian example of Section 3.5.1 (in the sense that, in the latter, the practitioner has unrealistic knowledge of the exact, easily calculable, closed form of \( \pi_\theta (x_n \mid y_{1:n-1}) \)).

### 3.6 New smoothing algorithms

We now break from PMCMC to revisit an idea from Section 3.3. In that section, we explained how Algorithm 1 yielded a sequence of optimal pseudo-priors in the Type B simulations of Section 3.2.1, which explained their very good performance. One could use this idea to develop new ways to perform smoothing.

Re-write the FFBS recursion (2.2.6) in the form

\[
\pi_\theta (x_t, x_{t+1} \mid y_{1:T}) = \pi_\theta (x_t \mid y_{1:t}) \pi_\theta (x_{t+1} \mid y_{1:T}) \frac{\pi_\theta (x_{t+1} \mid y_{1:T}) f_\theta (x_{t+1} \mid x_t)}{\pi_\theta (x_{t+1} \mid y_{1:t})},
\]  

(3.6.1)

and re-write the two-filter smoothing formula (2.2.7) in the form

\[
\pi_\theta (x_{t+1} \mid y_{1:T}) = \pi_\theta (x_{t+1} \mid y_{1:T}) \frac{\int f_\theta (x_{t+1} \mid x_t) \pi_\theta (x_t \mid y_{1:T}) \, dx_t}{p_\theta (y_{t+1:T} \mid y_{1:t})},
\]

\[
\Rightarrow \pi_\theta (x_t, x_{t+1} \mid y_{1:T}) = \pi_\theta (y_{t+1:T} \mid x_{t+1}) f_\theta (x_{t+1} \mid x_t) \pi_\theta (x_t \mid y_{1:t})
\]

(3.6.2)

If we combine (3.6.2) and (2.9.5), we find

\[
\pi_\theta (x_t, x_{t+1} \mid y_{1:T}) = \frac{\tilde{p} (y_{t+1:T}) \pi_\theta (x_{t+1} \mid y_{t+1:T}) f_\theta (x_{t+1} \mid x_t) \pi_\theta (x_t \mid y_{1:t})}{\xi_{t+1,\theta} (x_{t+1}) p_\theta (y_{t+1:T} \mid y_{1:t})}.
\]  

(3.6.3)
Choosing the optimal pseudo-prior $\xi_{t+1,\theta}(x_{t+1}) = \pi_{\theta}(x_{t+1} | y_{1:t})$, and recalling how this modifies the target (3.3.2) of the backward filter, we demonstrate that the generalised two-filter smoothing formula (3.6.3) is actually a special case of the FFBS recursion (3.6.1) when the optimal pseudo-prior is used:

$$
\pi_{\theta}(x_t, x_{t+1} | y_{1:T}) = \frac{\tilde{p}(y_{t+1:T}) \tilde{\pi}_{\theta}(x_{t+1} | y_{1:T}) \pi_{\theta}(x_t | y_{1:t})}{\pi_{\theta}(x_{t+1} | y_{1:t}) \pi_{\theta}(x_t | y_{1:t})}
$$

This section develops three new smoothing algorithms in which one uses an approximation of $\pi_{\theta}(x_{t+1} | y_{1:t})$ and a modified version of Algorithm 11 to target $\tilde{\pi}_{\theta}(x_{t+1} | y_{1:T})$. The first algorithm uses a parametric approximation of $\pi_{\theta}(x_{t+1} | y_{1:t})$. The second algorithm uses Algorithm 7 to obtain a non-parametric kernel approximation:

$$
\tilde{\pi}_{\theta}(x_{t+1} | y_{1:t}) = \frac{1}{N} \sum_{j=1}^{N} f_{\theta}(x_{t+1} | \tilde{x}_{t}^{j}),
$$

where $\tilde{x}_{t}^{j}$ have been resampled. We will see that this second algorithm is of complexity $O(N^2)$ per iteration. To bypasses this $O(N^2)$ complexity, we will develop a third algorithm in which the $O(N^2)$ algorithm is approximated via the use of random weights (see Section 2.5.1).

### 3.6.1 Parametric and marginal smoothers

Recall that we would like to target the sequence of densities (3.3.2) for $n \in \{1, \ldots, T-1\}$. The parametric backward smoother is simply Algorithm 11 with the pseudo-prior $\xi_{n,\theta}(x_n)$ replaced with a parametric approximation, which is denoted by $\pi_{n,\theta}^{\text{param}}(x_n | y_{1:n-1})$. The algorithm is formalised as Algorithm 30.

Using the parametric approximations is not ideal, as the backward importance weights lose their unbiasedness. We prefer to target the sequence of densities (3.3.2) using a backward SMC filter where the pseudo-priors $\{\xi_n(\theta)(x_n)\}_{n \in \{1, \ldots, T\}}$ are replaced with unbiased
Algorithm 30 Backward parametric smoother for HMMs

- Step 1: For $i \in \{1, \ldots, N\}$, sample $X^i_T \sim q_\theta(\cdot)$ and compute the un-normalised weight:

$$W^i_T = \frac{\pi_{T, \theta}(x^i_T | y_{1:T-1}) g_\theta(y_T | x^i_T)}{q_\theta(x^i_T)}.$$

For $i \in \{1, \ldots, N\}$, sample $A^i_T \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $j^{th}$ probability $w^j_T \propto W^j_T$. The sample $\{a^1_N\}$ are the indices of the resampled particles. Set all normalised weights equal to $1/N$, and set $n = T - 1$.

- Step 2: If $n = 0$, stop. Otherwise, for $i \in \{1, \ldots, N\}$, sample $X^i_n | x^{a(i)}_{n+1} \sim q_\theta(\cdot | x^{a(i)}_{n+1})$ and compute the un-normalised weight:

$$W^i_n = \frac{\pi_{n, \theta}(x^i_n | y_{1:n-1}) f_\theta(x^{a(i)}_{n+1} | x^i_{n+1}) g_\theta(y_n | x^i_n)}{\pi_{n+1, \theta}(x^{a(i)}_{n+1} | y_{1:n}) g_\theta(x^i_n | x^{a(i)}_{n+1})}.$$

For $i \in \{1, \ldots, N\}$, sample $A^i_{n:T} \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $j^{th}$ probability $w^j_n \propto W^j_n$. Set all normalised weights equal to $1/N$, and set $n = n - 1$. Return to the start of Step 2.

approximations. To obtain an $O(N^2)$ unbiased algorithm, one can run a forward SMC filter similar to Algorithm 7. A backward SMC filter with $\xi_{n, \theta}(x_n)$ approximated by (3.6.4) at each time step $n$ would then constitute the backward smoother (see Algorithm 31). We call this smoother a “marginal smoother” because in the next section we will present an algorithm that admits it as a marginal.

3.6.2 Random weight smoothing

We now develop an $O(N)$ backward smoother that admits Algorithm 31 as a marginal. The key trick employed in the algorithm of this section uses the principles of Section 2.5.1: we will sample random weights and use these random weights to approximate the weights (3.6.5).

Referring to (3.3.2) and (3.6.4), one can introduce a discrete valued auxiliary variable
Algorithm 31 \( O(N^2) \) marginal smoother for HMMs

- **Step 0:** Run Algorithm 7 to obtain resampled \( \{\hat{x}_n^i\}_{n \in \{1, \ldots, T-1\}} \).

- **Step 1:** For \( i \in \{1, \ldots, N\} \), sample \( \hat{X}_T^i \sim q_\theta(\cdot) \) and compute the un-normalised weight:
  \[
  \hat{W}_T^i = \frac{g_\theta(y_T | \hat{X}_T^i) \sum_{h=1}^N f_\theta(\hat{x}_T^i | \hat{x}_T^{h-1})}{q_\theta(\hat{x}_T^i)}.
  \]

  For \( i \in \{1, \ldots, N\} \), sample \( \hat{A}_T^i \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j \)th probability \( \frac{\hat{W}_T^j}{\sum_{j=1}^N \hat{W}_T^j} \). The sample \( \{\hat{A}_T^j; N\} \) are the indices of the resampled particles. Set all normalised weights equal to \( 1/N \), and set \( n = T - 1 \).

- **Step 2:** If \( n = 0 \) stop. Otherwise, for \( i \in \{1, \ldots, N\} \), sample \( \hat{x}_n^i | \hat{x}_{n+1}^{a(i)} \sim q_\theta(\cdot | \hat{x}_{n+1}^{a(i)}) \) and compute the un-normalised weight:
  \[
  \hat{W}_n^i = \frac{g_\theta(y_n | \hat{x}_n^i) f_\theta(\hat{x}_{n+1}^{a(i)} | \hat{x}_n^i) \sum_{h=1}^N f_\theta(\hat{x}_n^i | \hat{x}_n^{h-1})}{q_\theta(\hat{x}_n^i | \hat{x}_{n+1}^{a(i)}) \sum_{h=1}^N f_\theta(\hat{x}_n^{a(i)} | \hat{x}_n^{h})}.
  \]  \hspace{1cm} (3.6.5)

  For \( i \in \{1, \ldots, N\} \), sample \( \hat{A}_n^i \in \{1, \ldots, N\} \) from a discrete distribution on \( \{1, \ldots, N\} \) with \( j \)th probability \( \frac{\hat{W}_n^j}{\sum_{j=1}^N \hat{W}_n^j} \). Set all normalised weights equal to \( 1/N \), and set \( n = n - 1 \). Return to the start of Step 2.
Thus, we should select which means the incremental weights (3.6.7) would now be proportional to

\[
\tilde{\pi}_\theta (j, x_{n:T} \mid y_{1:T}) \propto f_\theta (x_n \mid \frac{x_j}{x_{n-1}}) g_\theta (y_n \mid x_n) \left[ \prod_{j=n+1}^{T} g_\theta (y_l \mid x_l) f_\theta (x_l \mid x_{l-1}) \right] \tag{3.6.6}
\]

for \( n \in \{1, \ldots, T\} \). The incremental weights (2.9.7) for Algorithm 11 then take the form

\[
\tilde{W}_n \propto \frac{\tilde{\pi}_\theta (j, \frac{x_i}{x_n}, \frac{x_j}{x_{n+1}} \mid y_{n+1:T})}{\tilde{\pi}_\theta (\frac{x_j}{x_{n+1:T}} \mid y_{n+1:T})} q_\theta (j, \frac{x_i}{x_n} \mid \frac{x_j}{x_{n+1}}) g_\theta (y_n \mid \frac{x_i}{x_n}) f_\theta \left( \frac{x_j}{x_{n+1}} \mid \frac{x_i}{x_n} \right) f_\theta \left( \frac{x_j}{x_{n+1}} \mid \frac{x_j}{x_{n-1}} \right) q_\theta (j, \frac{x_i}{x_n} \mid \frac{x_j}{x_{n+1}}) \tilde{\pi}_\theta \left( \frac{x_j}{x_{n+1}} \mid y_{1:n} \right).
\tag{3.6.7}
\]

To obtain an exact draw from the target (3.6.6), the optimal importance density would be

\[
q^\text{opt}_\theta \left( j, \frac{x_i}{x_n} \mid \frac{x_j}{x_{n+1}} \right) \propto g_\theta (y_n \mid \frac{x_i}{x_n}) f_\theta \left( \frac{x_j}{x_{n+1}} \mid \frac{x_i}{x_n} \right) f_\theta \left( \frac{x_j}{x_{n+1}} \mid \frac{x_j}{x_{n-1}} \right) p_\theta \left( \frac{x_i}{x_n}, \frac{x_j}{x_{n+1}}, \frac{x_j}{x_{n-1}} \right) p_\theta \left( y_n, \frac{x_j}{x_{n+1}} \mid \frac{x_j}{x_{n-1}} \right).
\tag{3.6.8}
\]

Thus, we should select

\[
q^\text{opt}_\theta \left( j \mid \frac{x_j}{x_{n+1}} \right) \propto p_\theta \left( y_n, \frac{x_j}{x_{n+1}} \mid \frac{x_j}{x_{n-1}} \right) \propto p_\theta \left( y_n, \frac{x_j}{x_{n+1}}, \frac{x_j}{x_{n-1}} \right) \tag{3.6.8}
\]

\[
\propto \frac{p_\theta \left( y_n, \frac{x_j}{x_{n+1}}, \frac{x_j}{x_{n-1}} \right)}{\sum_{j=1}^{N} p_\theta \left( y_n, \frac{x_j}{x_{n+1}}, \frac{x_j}{x_{n-1}} \right)}
\]

and

\[
q^\text{opt}_\theta \left( \frac{x_i}{x_n} \mid j, \frac{x_j}{x_{n+1}} \right) \propto p_\theta \left( \frac{x_i}{x_n} \mid y_n, \frac{x_j}{x_{n+1}}, \frac{x_j}{x_{n-1}} \right), \tag{3.6.9}
\]

which means the incremental weights (3.6.7) would now be proportional to

\[
\frac{\sum_{j=1}^{N} p_\theta \left( y_n, \frac{x_j}{x_{n+1}}, \frac{x_j}{x_{n-1}} \right)}{\tilde{\pi}_\theta \left( \frac{x_j}{x_{n+1}} \mid y_{1:n} \right)}. \tag{3.6.10}
\]
Based on the above derivations, we could develop a smoothing algorithm that looks similar to Algorithm 11, only now the proposal densities would take the forms (3.6.8) and (3.6.9) and the incremental weights would be (3.6.10). However, we must address how the new incremental weights will be calculated, as expression (3.6.10) is still $O(N)$ in complexity (and performing an $O(N)$ calculation for each of $N$ particles leads to an $O(N^2)$ algorithm). In practice, we could approximate (3.6.8) with $p_{\theta}(y_n \mid \hat{x}_{n-1}^j)$ so that we do not need to evaluate the sum in the numerator of (3.6.10). If we can determine a suitable method for evaluating the denominator, then we would break the $O(N)$ complexity of (3.6.10).

There does not appear to be an auxiliary variable trick available to bypass calculating $\hat{\pi}_{\theta}\left(\hat{x}_{n+1}^{\alpha(i)} \mid y_{1:n}\right)$, but we can obtain an unbiased estimate of $\frac{1}{\hat{\pi}_{\theta}\left(\hat{x}_{n+1}^{\alpha(i)} \mid y_{1:n}\right)}$ (and thus an unbiased estimate of (3.6.10)) as follows. We make the assumption that $f_{\theta}(\hat{x}_{n+1}^{\alpha(i)} \mid \hat{x}_{n}^j) \leq C$, where $C$ might depend on $\hat{x}_{n+1}^{\alpha(i)}$ but is independent of $\hat{x}_{n}^j$. Also, consider a discrete probability distribution over $\{1, \ldots, N\}$:

$$\pi_{\theta}(j) = \gamma_{\theta}(j) = \frac{1}{N} f_{\theta}\left(\hat{x}_{n+1}^{\alpha(i)} \mid \hat{x}_{n}^j\right),$$

(3.6.11)

where $Z_{\theta}$ is a normalising constant and is equal to the denominator of (3.6.10). We can sample from (3.6.11) via rejection sampling using the uniform proposal $\tilde{q}(j) = \frac{1}{N}$, as

$$\frac{\gamma_{\theta}(j)}{\tilde{q}(j)} = f_{\theta}\left(\hat{x}_{n+1}^{\alpha(i)} \mid \hat{x}_{n}^j\right) \leq C.$$

Within a rejection sampling scheme, the acceptance probability of a candidate index $j$ is given by

$$\mathbb{P}_{\theta}\left(U \leq \frac{\gamma_{\theta}(j)}{C \tilde{q}(j)}\right) = \sum_{i=1}^{N} \frac{\gamma_{\theta}(i)}{C \tilde{q}(i)} \tilde{q}(i) = \frac{Z_{\theta}}{C},$$

where $U \sim \mathcal{U}(0, 1)$. The number of trials until a successful draw from (3.6.11) will follow
### 3.7 Implementation of the new smoothing algorithms

\( G \left( \frac{Z_\theta}{C} \right) \). Hence, the expectation of this number of trials is going to be \( \frac{C}{Z_\theta} \). To obtain an unbiased estimate of \( \frac{1}{Z_\theta} \), we can run rejection sampling until \( M \) samples from (3.6.11) have been obtained. Then, denoting by \( N_m \) the number of trials needed to obtain sample \( m \) after having obtained sample \( (m-1) \), an estimate is given by

\[
\hat{\frac{1}{Z_\theta}} = \frac{1}{MC} \sum_{m=1}^{M} N_m.
\]

(3.6.12)

To set the variance (with respect to the rejection sampling scheme) of this estimate to some desired value, consider the following:

\[
\mathbb{V}_\theta \left[ \frac{1}{Z_\theta} \right] = \frac{M}{(MC)^2} \mathbb{V}[N_m] = \frac{1}{MC^2} \left( \frac{1 - \frac{Z_\theta}{C}}{Z^2_\theta/C^2} \right) = \frac{1}{M} \left( \frac{C - Z_\theta}{CZ^2_\theta} \right).
\]

(3.6.13)

Now if we set (3.6.13) equal to some value \( \epsilon \), we see that the optimal choice for \( M \) will be

\[
M^{\text{opt}} = \left\lfloor \frac{(C - Z_\theta)/\epsilon}{CZ_\theta^2} \right\rfloor.
\]

One may not be able to calculate \( M^{\text{opt}} \) in a real world application. Furthermore, \( M^{\text{opt}} \) may be larger than \( N \) (depending on \( \epsilon \)), thereby defeating the purpose of trying to approximate the marginal smoother. Thus, we suggest setting \( M = \lceil \log(\mathcal{N}) \rceil \) to boost algorithm speed. We formalise the \( O(MN) \) random weight smoother in Algorithm 32.

### 3.7 Implementation of the new smoothing algorithms

In the following numerical tests, we compare the new smoothing algorithms, as well the ideal backward smoother noted in Section 3.3, for the linear Gaussian model and the stochastic volatility model of Section 3.2. The four algorithms are labelled as follows:

- **Algorithm A**: random weight smoother (Algorithm 32).
- **Algorithm B**: marginal smoother (Algorithm 31).
- **Algorithm C**: backward parametric smoother (Algorithm 30).
Algorithm 32 $O(MN)$ random weight smoother for HMMs

- Step 0: Run Algorithm 7 to obtain resampled $\{\tilde{x}_n\}_{n \in \{1, \ldots, T-1\}}$.

- Step 1: For $i \in \{1, \ldots, N\}$,
  - Sample $J \sim q_\theta(\cdot) = p_\theta(y_T | \tilde{x}^j_{T-1})$.
  - Sample $\tilde{X}^i_T | J \sim q_\theta(\cdot | J) \propto p_\theta(\tilde{x}_T^i | y_T, \tilde{x}^j_{T-1})$.
  - Compute the un-normalised weight:
    $$\tilde{W}^i_T \propto \frac{g_\theta(y_T | \tilde{x}_T^i) f_\theta(\tilde{x}_T^i | \tilde{x}^j_{T-1})}{p_\theta(y_T | \tilde{x}^j_{T-1}) p_\theta(\tilde{x}^j_{T-1})}.$$ 

For $i \in \{1, \ldots, N\}$, sample $\tilde{A}^i_T \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $k^{th}$ probability $\tilde{W}^i_T \propto \tilde{W}^i_T$. The sample $\{\tilde{A}^i_T\}_{i \in \{1, \ldots, N\}}$ are the indices of the resampled particles. Set all normalised weights equal to $1/N$, and set $n = T - 1$.

- Step 2: If $n = 0$ stop. Otherwise, for $i \in \{1, \ldots, N\}$,
  - Sample $J \sim q_\theta^{opt}(\cdot | \tilde{x}_{n+1}^a(i)) \approx p_\theta(y_n | \tilde{x}_{n-1}^j)$.
  - Sample $\tilde{X}^i_{n+1} | J, \tilde{x}_{n+1}^a(i) \sim q_\theta^{opt}(\cdot | J, \tilde{x}_{n+1}^a(i)) \propto p_\theta(\tilde{x}_{n+1}^i | y_n, \tilde{x}_{n+1}^a(i), \tilde{x}_{n-1}^j)$.
  - Run a rejection sampling algorithm to obtain $M$ samples from (3.6.11), and compute $\pi_\theta(1/a_{n+1}^i | y_{1:n})$ via (3.6.12).
  - Compute the un-normalised weight:
    $$\tilde{W}^i_n \propto \frac{g_\theta(y_n | \tilde{x}_n^i) f_\theta(\tilde{x}_n^i | \tilde{x}_{n+1}^a(i)) f_\theta(\tilde{x}_n^i | \tilde{x}_{n-1}^j)}{p_\theta(y_n | \tilde{x}_{n-1}^j) p_\theta(\tilde{x}_{n-1}^j) \pi_\theta(\tilde{x}_{n+1}^a(i) | y_{1:n})}.$$ 

For $i \in \{1, \ldots, N\}$, sample $\tilde{A}_{n,T}^i \in \{1, \ldots, N\}$ from a discrete distribution on $\{1, \ldots, N\}$ with $k^{th}$ probability $\tilde{W}_n^i \propto \tilde{W}_n^i$. Set all normalised weights equal to $1/N$, and set $n = n - 1$. Return to the start of Step 2.
Algorithm D: ideal backward smoother of Section 3.3 (recall that in this algorithm, one runs Algorithm 11 with \( \xi_{n,\theta}(x_n) = \pi_{\theta}(x_n \mid y_{1:n-1}) \)).

We implement all four algorithms for the linear Gaussian model. For the stochastic volatility model, only the first three algorithms are implemented because the ideal backward smoother is analytically unavailable. For Algorithms B, C, and D, we use the same importance distributions as in Section 3.2. For Algorithm A, we use

\[
q_\theta(j \mid \tilde{x}^{a(i)}_{n+1}) \approx p_\theta(y_n \mid \tilde{x}^j_{n-1}) \propto \mathcal{N}(y_n \mid GF^\top x_{n-1}, R + GQG')
\]

\[
q_\theta(\tilde{x}^i_n \mid j, \tilde{x}^{a(i)}_{n+1}) \propto p_\theta(\tilde{x}^i_n \mid y_n, \tilde{x}^{a(i)}_{n+1}, \tilde{x}^j_{n-1}) \propto \mathcal{N}(\tilde{x}^i_n \mid \mu^{(j,i)}, \Sigma)
\]

with

\[
\Sigma = \left( Q^{-1} + G' R^{-1} G + F' Q^{-1} F \right)^{-1}
\]

\[
\mu^{(j,i)} = \Sigma \left( Q^{-1} F \tilde{x}^j_{n-1} + G' R^{-1} y_n + F' Q^{-1} \tilde{x}^{a(i)}_{n+1} \right)
\]

for the linear Gaussian model, and we use

\[
q_\theta(j \mid \tilde{x}^{a(i)}_{n+1}) \approx \tilde{W}^j_{n-1}
\]

\[
q_\theta(\tilde{x}^i_n \mid j, \tilde{x}^{a(i)}_{n+1}) \propto p_\theta(\tilde{x}^i_n \mid \tilde{x}^{a(i)}_{n+1}, \tilde{x}^j_{n-1}) \propto \mathcal{N}(\tilde{x}^i_n \mid 0.5 \left( \tilde{x}^{a(i)}_{n+1} + \tilde{x}^j_{n-1} \right), 0.5\nu^2)
\]

for the stochastic volatility model. In both examples, \( M = \lceil \log(N) \rceil \) when performing the rejection sampling step of Algorithm A.

Using the unbiased estimate

\[
\hat{Z}_{\theta,1:T} = \prod_{n=1}^{T} \left[ \frac{1}{N} \sum_{l=1}^{N} \tilde{W}^l_{n} \right], \tag{3.7.1}
\]

we compare the variances of the algorithms’ estimates of the marginal likelihood for different pairings of values of the observation noise and the state noise. We also examine the algorithms’ degeneracies via their effective sample sizes. The numerical tests below show that the random weight smoother is able to approximate the marginal smoother quite
well. However, the incremental weights of the marginal smoother have a high variance by construction, and so the marginal smoother does not offer a suitable alternative to the ideal backward smoother.

3.7.1 Linear Gaussian model

For each of the four algorithms (A, B, C, and D), we consider values for $\nu^2$ and $\tau^2$ that range from one to 98 in steps of seven. Per algorithm, we run 50 simulations for each pair $(\nu^2, \tau^2)$.

In this example, the optimal pseudo-priors are known to be Gaussian distributions. Thus, for Algorithm C, Student’s t-distribution is used to approximate the optimal pseudo-priors. We run three versions of Algorithm C, where we set the degrees of freedom of the t-distributions to either one, five, or 10.

Results

- Algorithms A-D: $T = 100, N = 100$: see Figures 3.7.1 and 3.7.2
- Algorithms A-D: $T = 100, N = 1000$: see Figures 3.7.3 and 3.7.4

Algorithm A does not perform much worse than Algorithm B. There is some additional variance, but this is expected as the value $M = \lceil \log(N) \rceil$ is far smaller than $M^\text{opt}$. The most important result is in Figures 3.7.1 and 3.7.3, where we see that Algorithm B has a low effective sample size. As Algorithm B iterates, the estimate (3.6.4) conditions on an increasing number of observations. Therefore, the tails of the approximation become thinner. When one considers the backward incremental weights (3.6.5), one will notice that the tails in the numerator will be thicker than the tails in the denominator. Unless the observations are very informative, the weights will not be well behaved. This is why the simulations do better when $\tau^2 \ll \nu^2$. When the exact expressions for the optimal pseudo-priors are available (or when $N$ in the estimate (3.6.4) is very large), we do not observe this problem. Thus, by construction, Algorithm B cannot approximate Algorithm D well in the general case.
3.7 Implementation of the new smoothing algorithms

In general, we would not recommend Algorithms A or B to calculate unbiased estimates of the HMM normalising constant. Perhaps if one knew for certain that $\tau^2 \ll \nu^2$ then we could recommend either algorithm for smoothing, as in this case both have very high effective sample sizes and (with large values of $N$) the variances of the algorithms can be close to that obtained by the ideal backward smoother. However, this insight does not encourage us to apply Algorithms A or B in a PMCMC framework because parameters such as $\tau^2$ and $\nu^2$ would likely be unknown if one were running PMCMC. Furthermore, we can also not recommend Algorithm C in a PMCMC framework. The effective sample size of this algorithm can be quite high when parametric approximations of the optimal pseudo-priors are very accurate, but one must not forget that we cannot apply biased estimates of $\hat{Z}_{\theta,1:T}$ within PMCMC without introducing bias to PMCMC.
Effective sample size

Algorithm A

Algorithm B

Algorithm C, DOF=1

Algorithm C, DOF=5

Algorithm C, DOF=10

Figure 3.7.1: The graphs display the effective sample size of each version of Algorithms A, B, and C for the case $T = 100$ and $N = 100$. Each block represents the average of 50 simulations. The effective sample size is measured after the final iteration of each algorithm. Note that the effective sample size of Algorithm D is not graphed because its effective sample size is always exactly equal to $N$. 
3.7 Implementation of the new smoothing algorithms

Figure 3.7.2: Each graph measures the difference between the log of the variance of estimates of $Z_{\theta,1:T}$ obtained by two smoothing algorithms. We set $T = 100$ and $N = 100$. Each block represents the average of 50 simulations.
Figure 3.7.3: The graphs display the effective sample size of each version of Algorithms A, B, and C for the case $T = 100$ and $N = 1000$. Each block represents the average of 50 simulations. The effective sample size is measured after the final iteration of each algorithm. Note that the effective sample size of Algorithm D is not graphed.
3.7 Implementation of the new smoothing algorithms

Figure 3.7.4: Each graph measures the difference between the log of the variance of estimates of $Z_{\theta,1:T}$ obtained by two smoothing algorithms. We set $T = 100$ and $N = 1000$. Each block represents the average of 50 simulations.
3.7.2 Stochastic volatility model

Algorithm D is analytically unavailable for this stochastic volatility model, and so we only look at Algorithms A, B, and C for this example. For each of the three algorithms, we consider values for $\tau$ that range from one to 98 in steps of seven. We allow $\nu$ to range from 0.25 to seven in steps of 0.5. Per algorithm, we run 50 simulations for each pair $(\tau, \nu)$.

In this example, the form of the optimal pseudo-priors is unknown. Thus, we use Algorithm 7 to output samples from the optimal pseudo-priors. Using an expectation maximisation algorithm, we fit a mixture of two Gaussian distributions to each sample to create a sequence of parametric pseudo-priors. These parametric pseudo-priors are used in Algorithm C.

Results

- Algorithms A-C: $T = 100, N = 100$: see Figures 3.7.5 and 3.7.6
- Algorithms A-C: $T = 100, N = 1000$: results not pictured

These results fortify the conclusion from Section 3.7.1: we cannot recommend Algorithms A, B or C to calculate estimates of the HMM normalising constant or to be used within PMCMC. We notice the same trends as for the linear Gaussian example, which is that Algorithm A performs about as well as Algorithm B, but as the weights of Algorithm B are not well behaved by construction both algorithms exhibit low effective sample sizes. Algorithm C performs similarly to Algorithm B, which is not surprising, as the parametric approximations to the optimal pseudo-priors are based off of the output of Algorithm 7. The results are similar for $N = 100$ and $N = 1000$, and so they are not pictured.
Effective sample size

### Algorithm A

<table>
<thead>
<tr>
<th>State noise, ( \nu )</th>
<th>Observation noise, ( \tau )</th>
<th>Neff of random weight smoother</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.5</td>
<td>20</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>1</td>
<td>1.5</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>2.5</td>
<td>2.5</td>
<td>100</td>
</tr>
</tbody>
</table>

### Algorithm B

<table>
<thead>
<tr>
<th>State noise, ( \nu )</th>
<th>Observation noise, ( \tau )</th>
<th>Neff of marginal smoother</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.5</td>
<td>20</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>1</td>
<td>1.5</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>2.5</td>
<td>2.5</td>
<td>100</td>
</tr>
</tbody>
</table>

### Algorithm C

<table>
<thead>
<tr>
<th>State noise, ( \nu )</th>
<th>Observation noise, ( \tau )</th>
<th>Neff of parametric smoother</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.5</td>
<td>20</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>1</td>
<td>1.5</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>2.5</td>
<td>2.5</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 3.7.5: The graphs display the effective sample sizes of Algorithms A, B, and C for the case \( T = 100 \) and \( N = 100 \). Each block represents the average of 50 simulations. The effective sample size is measured after the final iteration of each algorithm.
3.8 Discussion

In this chapter, we initially introduced the following:

- an $O(N)$ unbiased estimate of the marginal likelihood of an HMM that is calculated via generalised two-filter smoothing (for which we also proved a central limit theorem) and
3.8 Discussion

- an $O(N)$ PMMH algorithm and an $O(N)$ PG sampler that employ generalised two-filter smoothing within their sampling procedures.

In the wider literature, only a single particle filter had previously been used to calculate $\hat{Z}_{\theta,1:T}$, and the original PMCMC algorithms of [2] also use a single particle filter. The motivation for using two-filter smoothing came from the hypothesis that perhaps using two filters as opposed to one would reduce the variance of the importance weights that are used in that estimate and by those algorithms. Through extensive numerical studies in Sections 3.2 and 3.5, we found that using generalised two-filter smoothing is only a benefit here when one can set $\xi_{n,\theta}(x_n) = \pi_\theta(x_n | y_{1:n-1})$ in Algorithm 11 within Algorithm 13. Without this setting, the weight of a sampled path from time one to time $T$ is actually greater when one uses two filters as opposed to one because the point at which the two filters meet is a source of great variance. However, when one uses $\xi_{n,\theta}(x_n) = \pi_\theta(x_n | y_{1:n-1})$, the incremental weights of the backward particle filter have a very low variance because one is able to sample directly from the target of interest. Thus, using two filters as opposed to one is not actually all that beneficial when calculating $\hat{Z}_{\theta,1:T}$ or running PMCMC. Rather, employing a backward filter with $\xi_{n,\theta}(x_n) = \pi_\theta(x_n | y_{1:n-1})$ as opposed to a forward filter leads to superior performance.

The pseudo-prior $\xi_{n,\theta}(x_n) = \pi_\theta(x_n | y_{1:n-1})$ is typically not analytically available in a real world application of interest. In fact, if one can calculate $\pi_\theta(x_n | y_{1:n-1})$, then there is likely no need to resort to particle methods at all. The findings of this first part of the chapter do hint, though, that if one has access to a good approximation of $\pi_\theta(x_n | y_{1:n-1})$, then our methods introduced in this chapter should be used. An even better strategy would be to use the approximation of $\pi_\theta(x_n | y_{1:n-1})$ to run an approximation of the ideal backward SMC smoother of Section 3.3 all the way through time point one and forego using the forward filter entirely.

Approximations of $\pi_\theta(x_n | y_{1:n-1})$ may not be easy to determine. Thus, the latter part of this chapter includes an exploration of alternative smoothing algorithms that approximate the ideal backward SMC smoother of Section 3.3. We developed an $O(N^2)$ marginal smoother and an $O(MN)$ random weight smoother. While numerical tests showed that the random weight smoother is able to approximate the marginal smoother quite well, the
same tests also demonstrated that the incremental weights of the marginal smoother have a high variance by construction. Thus, our smoothers do not offer suitable alternatives to the ideal backward smoother. In future work, it may be worthwhile to make other attempts at approximating the ideal backward SMC smoother.

Appendix

A Proof of the central limit theorem from Section 3.1.2

We first describe a Feynman-Kac representation (see [23]). Let $t \in \{3, \ldots, T-2\}$, with $T > 4$ also fixed. To simplify the notation in this section, we drop the arrows from the latent process (i.e., $\vec{x}$ and $\widehat{x}$) that we have used throughout. It should remain clear when we are referring to a forward particle and when we are referring to a backward particle.

For $n \in \{1, \ldots, t-1\}$, define the forward Feynman-Kac un-normalised $n$–time marginal as

$$\gamma_{n,\theta}(dx_n) = \int [\prod_{p=1}^{n-1} \vec{W}_p(x_p) M_p(x_{p-1}, dx_p)] M_n(x_{n-1}, dx_n),$$

with $x_p = (x'_p, \hat{x}_p) \in \mathbb{R}^{2d_x}$, $M_1(x_0, dx_1) = \delta_{\hat{x}_0}(dx'_1) q_{1,\theta}(\hat{x}_1|x'_1) d\hat{x}_1$ and

$$M_p(x_{p-1}, dx_p) = \delta_{\hat{x}_{p-1}}(dx'_p) q_{p,\theta}(\hat{x}_p|x'_p) d\hat{x}_p.$$

The normalised measure is

$$\eta_{n,\theta}(dx_n) = \gamma_{n,\theta}(dx_n) / \gamma_{n,\theta}(1),$$

and the particle approximation of this measure is denoted $\eta_{n,\theta}^N(dx_n)$. We also define the forward semi-group operator as

$$Q_{p,n}(x_p, dx_n) = \int \prod_{q=p}^{n-1} \vec{W}_q(x_q) M_{q+1}(x_q, dx_{q+1}),$$
with $1 \leq p \leq n \leq t - 1$. The selection mutation operator is

$$\vec{\Phi}_q(\vec{\eta}_{q-1,\theta})(\cdot) = \frac{\vec{\eta}_q \vec{W}_{q-1}^2 M_q(\cdot)}{\vec{\eta}_{q-1,\theta}(\vec{W}_{q-1})} \quad q \in \{0, \ldots, t - 1\},$$

with the conventions $\vec{\Phi}_1(\vec{\eta}_{0,\theta}) = \vec{\eta}_{1,\theta}$.

For $n \in \{0, \ldots, T - t - 1\}$, we define the backward Feynman-Kac un-normalised $n$–time marginal as

$$\vec{\gamma}_{T-n,\theta}(dx_n) = \int \prod_{p=0}^{T-n-1} \vec{W}_{T-p}(x_{T-p}) M_{T-p}(x_{T-p+1}, dx_{T-p}) M_n(x_{n+1}, dx_n),$$

with $x_n = (x'_n, \tilde{x}_n) \in \mathbb{R}^{2d_x}$, $M_T(d\tilde{x}_T) = q_T(\tilde{x}_T)d\tilde{x}_T$ and

$$M_n(x_{n+1}, dx_n) = q_{n,\theta}(\tilde{x}_n|x'_n)dx_n\delta_{\tilde{x}_{n+1}}(dx'_n), n \in \{t + 1, \ldots, T - 1\}.$$

The normalised measure is

$$\vec{\gamma}_{T-n,\theta} = \frac{\vec{\gamma}_{T-n,\theta}(dx_n)}{\vec{\gamma}_{T-n,\theta}(1)},$$

and the particle approximation of this measure is denoted $\vec{\gamma}_{T-n,\theta}$. Also, we define the backward semi-group operator as

$$\vec{Q}_{p,n}(x_p, dx_n) = \int \prod_{s=0}^{p-n-1} \vec{W}_{p-s}(x_{p-s}) M_{p-s-1}(x_{p-s}, dx_{p-s-1}),$$

with $T \geq p \geq n \geq t + 1$. The selection mutation operator is

$$\vec{\Phi}_{T-q}(\vec{\gamma}_{T-q+1,\theta})(\cdot) = \frac{\vec{\gamma}_{T-q+1,\theta}(\vec{W}_{T-q+1} M_{T-q}(\cdot))}{\vec{\gamma}_{T-q+1,\theta}(\vec{W}_{T-q+1})} \quad q \in \{0, \ldots, T - t - 1\},$$

and $\vec{\Phi}_T(\vec{\gamma}_{T+1}) = \vec{\gamma}_T$. 
We use the notation
\[
I_{gf}(\tilde{x}_{t-1}, \tilde{x}_{t+1}) = \int_{\mathbb{R}^d} g_\theta(y_t|x_t) f_\theta(\tilde{x}_{t+1}|x_t) f_\theta(x_t|\tilde{x}_{t-1}) dx_t
\]
\[
W_{t+1}^\xi(x_{t+1}) = \frac{\tilde{W}_{t+1}(x_{t+1})}{\xi_{t+1,\theta}(\tilde{x}_{t+1})}
\]
with the definitions
\[
\mu_{-1}(\tilde{W}_{t-1}^\gamma_{t-1,\theta}|\tilde{W}_{t+1}^\xi I_{gf}(\cdot, \cdot)) = \int \mu_{-1}(dx_{t-1}) \tilde{W}_{t-1}(x_{t-1}) \times \\
[\int \tilde{\gamma}_{t+1,\theta}(dx_{t+1}) \tilde{W}_{t+1}^\xi(x_{t+1}) I_{gf}(\tilde{x}_{t-1}, \tilde{x}_{t+1})]
\]
\[
\mu_{+1}(\tilde{W}_{t+1}^\xi \tilde{\gamma}_{t-1,\theta}(\tilde{W}_{t-1} I_{gf}(\cdot, \cdot))) = \int \mu_{+1}(dx_{t+1}) \tilde{W}_{t+1}^\xi(x_{t+1}) \times \\
[\int \tilde{\gamma}_{t-1,\theta}(dx_{t-1}) \tilde{W}_{t-1}(x_{t-1}) I_{gf}(\tilde{x}_{t-1}, \tilde{x}_{t+1})]
\]
for \(\sigma\)-finite measures \(\mu_{-1}, \mu_{+1}\).

Using the above notations, we can write (3.1.4) as
\[
\hat{Z}_{\theta;1:T} = \tilde{\gamma}_{t-1}^N(1) \tilde{\gamma}_{t+1}^N(1) \times \\
\sum_{l=1}^N \tilde{W}_{t-1}(x_{l-1}^T) \tilde{W}_{t+1}(x_{l+1}^T) f_\theta(x_{l}^T|\tilde{x}_{l}^T) f_\theta(\tilde{x}_{l+1}^T|x_{l}^T) \times \\
g_\theta(y_t|x_t)
\]
with
\[
\tilde{\gamma}_{t-1}^N(1) = \prod_{p=1}^{t-2} \frac{1}{N} \sum_{i=1}^N \tilde{W}_{p}(x_{p}^T)
\]
\[
\tilde{\gamma}_{t+1}^N(1) = \prod_{p=0}^{T-t-2} \frac{1}{N} \sum_{i=1}^N \tilde{W}_{T-p}(x_{T-p})
\]

To prove the central limit theorem, we make use of the following assumption, which is similar to \((H)_m\) \((m = 2)\) of [16]. It is used to control remainder terms when constructing a central limit theorem. It implies that the backward Markov proposal kernels mix very quickly.
1. The incremental weights all satisfy

\[ \delta_\theta = \sup_{x,y,n} \frac{\tilde{W}_n(x)}{\tilde{W}_n(y)} < \infty \]

where \( 1 \leq n \leq t - 1 \) and

\[ \delta_\theta = \sup_{x,y,n} \frac{\tilde{W}_n(x)}{\tilde{W}_n(y)} < \infty \]

where \( t + 1 \leq n \leq T \). For each \( \theta \in \Theta \), there exist \( 0 < C_\theta < \overline{C}_\theta < \infty \) such that for every \( x, x' \in \mathbb{R}^d \), \( n \in \{1, \ldots, T\} \), \( y_n \in \mathbb{R}^d \),

\[ C_\theta \leq f_\theta(x'|x) \leq \overline{C}_\theta, \quad C_\theta \leq \xi_{n,\theta}(x) \leq \overline{C}_\theta, \quad C_\theta \leq g_\theta(y_n|x) \leq \overline{C}_\theta. \]

In addition, for each \( \theta \in \Theta \), there exist \( 0 < C_\theta < \overline{C}_\theta < \infty \) as above, such that for each \( x_t, x_{t-1}, x_t \in \mathbb{R}^d \) and for \( i \in \{1, \ldots, N\} \),

\[ C_\theta \leq q_{t,\theta}(x_t|x_{t-1}, x_{t+1}) \leq \overline{C}_\theta \]

\[ C_\theta \leq \beta_{t-1}^{(i)} \leq \overline{C}_\theta, \quad C_\theta \leq \beta_{t+1}^{(i)} \leq \overline{C}_\theta. \]

2. For some sequence of numbers \( \omega_p^{(2)} \in [1, \infty) \) such that for each \( p \in \{-1, \ldots, T - t - 2\} \) and any \( (x, x') \in \mathbb{R}^{2d} \) we have

\[ M_{T-p,T-p-2}(x,dy) \leq \omega_p^{(2)} M_{T-p,T-p-2}(x',dy) \]

where \( M_{p,q} = M_{p-1} \ldots M_q, p \geq q. \)

**Proof of Theorem 3.1.1** Consider the following:

\[
\mathbb{E}[\tilde{Z}_{\theta,1:T} | \mathcal{F}_{t-1}^N \otimes \mathcal{F}_{t+1}^N] = \gamma_{t-1}^N(1) \gamma_{t+1}^N(1) \frac{1}{N^2} \sum_i \sum_j (\tilde{W}_{t-1}^i \tilde{W}_{t+1}^j I_{gf})
\]

\[
= \gamma_{t-1}^N(1) \gamma_{t+1}^N(1) \tilde{W}_{t-1}^\xi \tilde{W}_{t+1}^\varepsilon (\tilde{W}_{t-1}^\xi \tilde{W}_{t+1}^\varepsilon I_{gf})
\]

\[
= \gamma_{t-1}^N \gamma_{t+1}^N (\tilde{W}_{t-1}^\xi \tilde{W}_{t+1}^\varepsilon I_{gf}),
\]
Chapter 3. Generalised Two-filter Smoothing and Particle Markov chain Monte Carlo

where $\mathcal{F}_{t-1}^N$ and $\mathcal{F}_{t+1}^N$ are the filtrations generated by the forward and backward particle systems up-to times $t-1$ and $t+1$, respectively. We use $\otimes$ to denote the product sigma algebra. Let $w = \tilde{W}_{t-1}(x_{t-1})\tilde{W}_{t+1}(x_{t+1})I_{gf}(x_{t-1}, x_{t+1})$. Through standard manipulations, we obtain

$$
\mathbb{E}[\hat{Z}_{\theta,1:T} | \mathcal{F}_{t-1}^N \otimes \mathcal{F}_{t+1}^N] - Z_{\theta,1:T} = (\gamma_{t-1}^N \otimes \gamma_{t+1}^N(w)) - (\gamma_{t-1} \otimes \gamma_{t+1}(w))
$$

which can be rewritten as

$$
= \left[ (\gamma_{t-1}^N - \gamma_{t-1}) \otimes \gamma_{t+1}(w) \right] + \left[ (\gamma_{t-1} \otimes (\gamma_{t+1}^N - \gamma_{t+1}(w)) \right] + \left[ (\gamma_{t-1}^N - \gamma_{t-1}) \otimes (\gamma_{t+1}^N - \gamma_{t+1}(w)) \right].
$$

We can use the decomposition of [23, Proposition 7.4.1] to obtain the following formula:

$$
\mathbb{E}[\hat{Z}_{\theta,1:T} | \mathcal{F}_{t-1}^N \otimes \mathcal{F}_{t+1}^N] - Z_{\theta,1:T} = \alpha(N) + \beta(N) + R(N)
$$

(A1)

where

$$
\alpha(N) = \sum_{q=1}^{t-1} \tilde{\gamma}_q(1)[\tilde{\gamma}_q^N - \Phi_q(\tilde{\eta}_q^N)](\tilde{\gamma}_{q,t-1}[\tilde{W}_{t-1}^N \gamma_{t+1}(\tilde{W}_{t+1}^N I_{gf}(\cdot, \cdot))])
$$

$$
\beta(N) = \sum_{q=0}^{T-t-1} \tilde{\gamma}_{T-q}(1)[\tilde{\gamma}_{T-q}^N - \Phi_{T-q}(\tilde{\eta}_{T-q}^N)](\tilde{Q}_{T-q,t}[\tilde{W}_{t+1}^N \gamma_{t-1}(\tilde{W}_{t-1} I_{gf}(\cdot, \cdot))])
$$

$$
R(N) = \sum_{q=1}^{t-1} \tilde{\gamma}_q(1)[\tilde{\gamma}_q^N - \Phi_q(\tilde{\eta}_q^N)](\tilde{Q}_{q,t-1}[\tilde{W}_{t-1}^N \gamma_{t+1} - \gamma_{t+1}\beta](\tilde{W}_{t+1}^N I_{gf}(\cdot, \cdot))])
$$

As each of $\alpha(N)$, $\beta(N)$, and $R(N)$ is a sum of martingale differences, it is straightforward to verify that the expectation of each is exactly zero. Thus, the unbiasedness property is
established.

By using the Marciniewicz-Zygmund inequality, we obtain

\[
\mathbb{E}\left[ \sqrt{N} \left( \hat{Z}_{\theta,1:T} - \mathbb{E}[\hat{Z}_{\theta,1:T}] \right) \right] = \mathbb{E}\left[ \sqrt{N} \left( \frac{\hat{Z}_{\theta,1:T} - \mathbb{E}[\hat{Z}_{\theta,1:T}]}{N^3} \right) \right] \leq \mathbb{E}\left[ \frac{\sqrt{N}}{N^3} \sum_{l=1}^{N} |w_l - \mathbb{E}[w_l]| \right] \leq C_N \mathbb{E}\left[ \sum_{l=1}^{N} |w_l - \mathbb{E}[w_l]| \right]
\]

for some \( C_N < +\infty \). For any fixed \( t, T \), we have

\[
\sup_{N \geq 1} \mathbb{E}[\gamma_{l-1}(1)^2]^{1/2} < \infty
\]

\[
\sup_{N \geq 1} \mathbb{E}[\gamma_{l+2}(1)^2]^{1/2} < \infty
\]

by the proof of Lemma A.1 below. Thus, via Cauchy-Schwarz, the expression (A2) approaches zero in the limit as \( N \to \infty \), and we can deduce that

\[
\sqrt{N} \left( \hat{Z}_{\theta,1:T} - \mathbb{E}[\hat{Z}_{\theta,1:T}] \right) \to_p 0,
\]

where \( \to_p \) denotes convergence in probability. Thus, recalling (A1), we have

\[
\sqrt{N} \left( \hat{Z}_{\theta,1,T} - Z_{\theta,1:T} \right) - \sqrt{N} \alpha(N) - \sqrt{N} \beta(N) - \sqrt{N} R(N) \to_p 0.
\]

The weak convergence of \( \sqrt{N} \alpha(N) \) and \( \sqrt{N} \beta(N) \) can be obtained by the independence of the terms and [23, Proposition 9.4.1]. By Lemma A.1, the remainder \( \sqrt{N} R(N) \) converges to zero in probability. \( \square \)
Lemma A.1 Assume (A1). Then for fixed $T > 2$, $t \in \{3, \ldots, T-2\}$ we have

$$\sqrt{N} R(N) \to \mathbb{P} 0.$$ 

Proof To shorten the subsequent notations, define:

$$\xi^{N}_{q,t-1}(x) = \bar{Q}_{q,t-1}[\hat{W}_{t-1}^{\xi} N_{t+1} - \bar{\gamma}_{t+1,0}] (\hat{W}_{t+1}^{\xi} I_{gf}(\cdot, \cdot))(x)$$

(A3)

$$\bar{\xi}^{N}_{q,t-1} = \sup_{x} \hat{W}_{t-1}(x) \sup_{x} Q_{q,t-1}[[\bar{\gamma}_{t+1} - \bar{\gamma}_{t+1,0}] (\hat{W}_{t+1}^{\xi} I_{gf}(\cdot, \cdot))(x).$$

(A4)

It is remarked that for any bounded function $\varphi$, $\sup_{x} Q_{q,t-1}(\varphi)(x) < \infty$ by assumption.

We will now show that $\sqrt{N} R(N)$ approaches zero in $\mathbb{L}_1$. To that end, we can consider the expectation of each summand in the series for $R(N)$. Via Cauchy-Schwarz, we have

$$\mathbb{E}[\bar{\gamma}^{N}_{q}(1)|\bar{\gamma}^{N}_{q} - \Phi_{q}(\bar{\gamma}^{N}_{q-1})](\frac{\xi^{N}_{q,t-1}}{\bar{\xi}^{N}_{q,t-1}}) \leq \mathbb{E}[(\bar{\xi}^{N}_{q,t-1})^{2}]^{1/2}.$$ 

(A5)

For the first expectation on the R.H.S., one can condition on $\bar{\xi}^{N}_{q,t-1} \otimes \bar{\xi}^{N}_{t+1}$ and apply the Marcinkiewicz-Zygmund inequality (noting that $\sup_{x} Q_{q,t-1}(x)/\bar{\xi}^{N}_{q,t-1}$ is upper-bounded by a finite deterministic constant) to obtain

$$\mathbb{E}[\bar{\gamma}^{N}_{q}(1)|\bar{\gamma}^{N}_{q} - \Phi_{q}(\bar{\gamma}^{N}_{q-1})](\frac{\xi^{N}_{q,t-1}}{\bar{\xi}^{N}_{q,t-1}}) \lesssim C/\sqrt{N} \mathbb{E}[\bar{\gamma}^{N}_{q}(1)]^{2}]^{1/2}.$$ 

Note that for each $q$, $\mathbb{E}[\bar{\gamma}^{N}_{q}(1)]^{2}]^{1/2} < \infty$ due to the upper-bound on $\bar{W}_{q}$. Now consider the second expectation on the R.H.S. of (A5). From the definition (A4), we have

$$\mathbb{E}[(\bar{\xi}^{N}_{q,t-1})^{2}]^{1/2} = \sup_{x} \hat{W}_{t-1}(x) \sup_{x} Q_{q,t-1}(1) \mathbb{E}[\bar{Q}_{q,t-1}[[\bar{\gamma}_{t+1} N_{t+1} - \bar{\gamma}_{t+1,0}] (\hat{W}_{t+1}^{\xi} I_{gf}(\cdot, \cdot)))]^{2}]^{1/2},$$ 

where $\bar{Q}_{q,t-1}(x) := \sup_{x} Q_{q,t-1}(x)/\sup_{x} Q_{q,t-1}(1)$. Application of Jensen’s inequality
A Proof of the central limit theorem from Section 3.1.2

and Fubini’s theorem leads to $\mathbb{E}[(\bar{\xi}_{q,t}^N - 1)^2]^{1/2}$ being less than or equal to

$$\sup_x \tilde{W}_{t-1}(x) \sup_x \tilde{Q}_{q,t-1}(1) \tilde{Q}_{q,t-1}\left(\mathbb{E}[|\bar{\xi}_{t+1}^N - \xi_{t+1}^N|\tilde{W}_{t}^I_{gf}(.,.)(\mathbb{1})^2]\right)^{1/2}.$$  

Then by [16, Theorem 5.1, Corollary 5.2] (it is remarked that the corollary of that paper can be adapted to deal when $\xi_{t+1}^N$ integrates a bounded function), it follows for $N$ large enough relative to $T - t$ (we will take $N$ to infinity and $T - t$ is fixed) that there exists some finite constant $C(T,t)$ which depends upon $T,t$ but not $q$ or $x_{t-1}$ such that

$$\mathbb{E}[(\bar{\xi}_{q,t-1}^N)^2]^{1/2} \leq \sup_x \tilde{W}_{t-1}(x) \sup_x \tilde{Q}_{q,t-1}(1) \frac{C(T,t)}{\sqrt{N}}.$$  

Hence we have

$$\sqrt{N} \mathbb{E}[|R(N)|] \leq \frac{C(T,t)}{\sqrt{N}},$$  

where $C(T,t)$ is some finite constant that may grow with $T$. We thus conclude the proof, as $T < \infty$. \qed
Chapter 4

Twisting the Alive Particle Filter

The focus now switches to a particular subset of HMMs whose observations have unknown or intractable likelihood densities (see Section 2.2.3). This chapter develops new SMC and PMMH algorithms that enable one to perform Bayesian inference for these models. Similar algorithms have appeared in the literature before; see [51], [52], and [67], and also Section 2.10. However, this work uses twisted proposals as in Section 2.12 to reduce the variance of SMC estimates of $Z_\theta$ in order to improve the convergence of PMMH. We call our new algorithms “alive twisted SMC” and “alive twisted PMMH”, as we are twisting the alive algorithms of [51].

The chapter begins by introducing new notation in Section 4.1, which is similar to the Feynman-Kac notation of [23] and [91]. Doing this allows for a more concise statement of the alive twisted algorithms and the accompanying theoretical work. Section 4.2 states the new SMC algorithm, which is a twisted version of the alive SMC algorithm of [51]. Section 4.3 states the choice of additive functional with which the proposals of this algorithm should be twisted. The function is often times not analytically available in a real world application (as in [91]), and so it must be approximated when possible. The assumptions and the main theorem that justify this choice of additive functional are stated as well. A proof of this result is given in the chapter appendix (Section A), and it very closely follows the work of [91].

We compare the alive twisted SMC algorithm to Algorithm 15 in Section 4.4 by implementing both on a linear Gaussian HMM (for which the optimal additive functional
can be very accurately approximated). The numerical example shows that the new algorithm’s estimate of the normalising constant has a lower variance when the true variances of the targeted HMM are low. This encouraging performance of the alive twisted particle filter prompts us to embed it within a PMMH algorithm in Section 4.5, and the alive twisted PMMH is compared to a PMMH which employs Algorithm 15 in Section 4.6. In that second numerical illustration, both algorithms are implemented on stochastic volatility models. In one model, the optimal choice of additive functional can be very closely approximated, and in the other it cannot. This numerical illustration demonstrates that it is only when the choice of additive functional can be very closely approximated that our alive twisted algorithm will outperform the non-twisted method. Section 4.7 concludes the chapter with a summary and discussion.

4.1 Notation and definitions

We begin by defining some notation that is specific to this chapter. We prove our results in a similar way to [91], and to make it easier for the interested reader to compare our work to that of [91], we use as similar a notation as possible to that original article.

Consider the HMM from Section 2.2.3. For the probability space \((\Omega, \mathcal{F}, \mathbb{P})\), let \(\Omega = H^\mathbb{Z}\) be the set of doubly infinite sequences valued in \(H\) and let \(\mathcal{F} = \mathcal{H}^{\otimes \mathbb{Z}}\). Then for \(\omega = \{\omega(n)\}_{n \in \mathbb{Z}} \in \Omega\), we can write each observation random variable as \(Y_n(\omega) = \omega(n)\). We use \(z\) to define a shift operator \(z : \Omega \to \Omega\) as \((z\omega)(n) = \omega(n+1)\), where applying \(z\) \(m\)-times is written \(z^m\). Thus, for example, \(Y_n(z\omega) = Y_{n+1}(\omega)\) and \(Y_n(z^m\omega) = Y_{n+m}(\omega)\).

At any time point, define

\[
M_\theta (\omega, x, dx) = f_\theta (dk \mid k) g_\theta (u(z\omega) \mid dk) = f_\theta (dk \mid k) g_\theta (du(\omega) \mid dk).
\]

Let \(M_{\theta}^{m_1, m_2} : \Omega \times E^{m_1} \times E^{\otimes m_2} \to [0, 1]\) be the transition density of a particle system (that uses multinomial resampling) with \(m_1\) ancestral paths and \(m_2\) propagated samples:

\[
M_{\theta}^{m_1, m_2} (\omega, x, dx) = \prod_{i=1}^{m_2} \Phi_\theta^{\omega, m_1} (\eta_\omega^{m_1}) (dx_i) \quad (4.1.1)
\]
\[
\Phi_{\theta}^{\omega,m_1_1} (\eta_{\omega,m_1} (dx^i)) = \frac{1}{m_1} \sum_{j=1}^{m_1} W(\omega, x^j) M_\theta (\omega, x^j, dx^j) / \frac{1}{m_1} \sum_{j=1}^{m_1} W(\omega, x^j) = \frac{1}{m_1} \sum_{j=1}^{m_1} W(\omega, x^j),
\]

where \( W(\omega, x^j) \) takes the form of (2.2.12) with \( y_n = y(\omega) \) and \( x_n = x^j \). Also, define \( W_{\tilde{m}_1} (\omega, x) = \frac{1}{m_1} \sum_{j=1}^{m_1} W(\omega, x^j) \), and define the additive functional \( f_{\tilde{m}_1} (\omega, x) = \frac{1}{m_1} \sum_{j=1}^{m_1} f(\omega, x^j) \), where \( f : \Omega \times E \rightarrow (0, \infty) \). To make our proofs easier to follow, we further establish some kernel and operator notation in Table B1 in Section B of the chapter appendix.

Let \( \tilde{M}_{\theta}^{m_1,m_2} : \Omega \times E^{m_1} \times E^{\otimes m_2} \rightarrow [0, 1] \) be some other sampling law, which may or may not be the same as \( M_{\theta}^{m_1,m_2} \), and define a family of kernels \( M_{\tilde{m}_1,m_2} \) similar to as in [91]:

**Definition 4.1.1** Any \( \tilde{M}_{\theta}^{m_1,m_2} \) is said to be a member of \( M_{\theta}^{m_1,m_2} \) if and only if there exist positive, finite constants \( (\tilde{\epsilon}_-, \tilde{\epsilon}_+ \) \) and probability measures \( \nu \in \mathcal{P}(E) \) and \( \tilde{\nu} \in \mathcal{P}(E^{m_2}) \) such that

1. \( \tilde{\nu} (\cdot) \leq \tilde{M}_{\theta}^{m_1,m_2} (\omega, x, \cdot) \leq \tilde{\epsilon}_+ \tilde{\nu} (\cdot) \quad \forall (\omega, x) \in \Omega \times E^{m_1}, \)

2. \( \tilde{\nu}^{\otimes m_2} \) is dominated by \( \tilde{\nu} \), and

3. \( \int_{E^{m_2}} \left( \frac{d\nu^{\otimes m_2}}{d\tilde{\nu}} (x') \right)^2 \tilde{\nu} (dx') < \infty. \)

Furthermore, when \( \tilde{M}_{\theta}^{m_1,m_2} \) is a member of \( M_{\theta}^{m_1,m_2} \), we write

\[
\phi_{\theta}^{\omega,m_1,m_2} (x, dx) = \frac{dM_{\theta}^{m_1,m_2} (\omega, x, \cdot)}{dM_{\theta}^{m_1,m_2} (\omega, x, \cdot)} (dx), \tag{4.1.2}
\]

which allows us to define the following:

\[
R_{\theta}^{m_1,m_2} (\omega, x, dx) = W_{\theta}^{m_1} (\omega, x)^2 \phi_{\theta}^{\omega,m_1,m_2} (x, x')^2 \tilde{M}_{\theta}^{m_1,m_2} (\omega, x, dx'),
\]

\[
J_{\theta}^{m_1,m_2} (\omega, x) = \int_{E^{m_2}} W_{\theta}^{m_1} (\omega, x)^2 \phi_{\theta}^{\omega,m_1,m_2} (x, x')^2 \tilde{M}_{\theta}^{m_1,m_2} (\omega, x, dx'), \quad \text{and}
\]

\[
L_{\theta}^{m_1,m_2} (\omega, x, dx') = \frac{R_{\theta}^{m_1,m_2} (\omega, x, dx')}{J_{\theta}^{m_1,m_2} (\omega, x)}. \]
Finally, this chapter frequently refers to the ratio
\[
\tilde{V}_{\omega, \theta, n} = \frac{\int_Y e^{m} \prod_{k=1}^{n} W(z^k, x) z^{x^k, m_k, m_{k+1}}(x, x')^2 \tilde{M}^{m_k, m_{k+1}}_{\theta}(z^k, x, dx')}{\left[ \int_Y \prod_{k=1}^{n} W(z^k, x) M_{\theta}(z^k, x, du) \right]^2}
\]
(4.1.3)
and to the additive functional
\[
h^{m_2}(\omega, x) = \frac{1}{m_2} \sum_{j=1}^{m_2} h(\omega, x^j),
\]
(4.1.4)
where \( h : \Omega \times E \to (0, \infty) \).

### 4.2 Alive twisted sequential Monte Carlo

Recall that Algorithm 15 can be used to sample from the HMM of Section 2.2.3. In an effort to try to reduce the variance of Algorithm 15's estimate of \( Z_\theta \), let us introduce a change of measure on the particle system generated by that algorithm. First, we can use the notation established in the previous section and simply re-write the stopping time (2.10.1) of Algorithm 15 as
\[
T_{\omega} = \inf \left\{ p \geq N : \sum_{i=1}^{p} W(\omega, x^i) \geq N \right\}.
\]
(4.2.1)
Next, we can use some additive functional of the form (4.1.4) to introduce the change of measure (similar to as in [91] and Section 2.12). The conditional density (2.10.3) of the alive particle filter then becomes
\[
\tilde{\pi}\left( x_n^{1:z_n-1:1}, t_{z_n-1:1} \mid \mathcal{F}_{n-1} \right)
\]
(4.2.2)

\[
\propto \pi\left( x_n^{1:z_n-1:1}, t_{z_n-1:1} \mid \mathcal{F}_{n-1} \right) \frac{1}{t_{z_n-1:1} - 1} \sum_{i=1}^{t_{z_n-1:1}-1} h(z_n^{i-1}, x_n^i).
\]
This change of measure allows us to better guide the sampling procedure of Algorithm 15. We will show below in Sections 4.3, 4.4, and 4.6, that when \( h \) is chosen appropriately (i.e., as (4.3.1)), the change in measure can be used to introduce future information about the model which is being sampled to a current time point of the particle system at the time of sampling, thereby lowering the variance of the unbiased estimate of the normalising constant.

The expression (4.2.2) can be normalised by changing the summand to

\[
\begin{align*}
&h\left(z^{n-1}_\omega, x^n_i\right) = h\left(z^{n-1}_\omega, x^n_i\right) \times \\
&\left[W\left(z^{n-1}_\omega, x^n_i\right) \cdot I\{a,a \geq N\} \left(t_{z^n-1}_\omega \right) \cdot \frac{N - 1}{t_{z^n-1}_\omega - 1}ight] \\
&+ \left(1 - W\left(z^{n-1}_\omega, x^n_i\right) \cdot I\{a,a \geq N+1\} \left(t_{z^n-1}_\omega \right) \cdot \frac{t_{z^n-1}_\omega - N}{t_{z^n-1}_\omega - 1}\right)
\end{align*}
\]

and dividing the entire R.H.S. of (4.2.2) by

\[
\Phi^{n-2}_\omega \cdot T_{z^n-1}_\omega \left( h\left(z^{n-1}_\omega, \cdot \right) \right).
\]

One can sample from the normalised version of (4.2.2) via Algorithm 33, which is known hereafter as the alive twisted particle filter, or alive twisted SMC. Algorithm 33 may appear complicated at first sight, but one must keep in mind that we are only changing the sampling of a single particle. All other particles are sampled as they would have been sampled in Algorithm 15. From a coding standpoint, there is little additional effort required by the practitioner. There is also no real increase in computational cost.

For a simulated path \( z_{1:n} \) generated by Algorithm 33, where \( T_{z^n-1}_\omega \) samples of \( x_p \) have been obtained, we have

\[
\hat{Z}_{\theta,1:n} = \prod_{k=0}^{n-1} \frac{N - 1}{T_{z^n-1}_\omega - 1} \cdot \frac{\sum_{i=1}^{T_{z^n-1}_\omega} Q_{\theta}^{k-1}\left(h(z^k_\omega, \cdot)\right)\left(x^{q(i)}_k\right)}{\sum_{i=1}^{T_{z^n-1}_\omega} W\left(z^k_\omega, x^i_{k+1}\right) h\left(z^k_\omega, x^i_{k+1}\right)}
\]

(4.2.3)

This estimate is clearly unbiased when the expectation is taken with respect to the transition
Algorithm 33 Alive twisted particle filter for HMMs

In the following, it is assumed that the first observation of the HMM is given the index of one (e.g., $y_1 = y_1(z^n_1 \omega)$ and $y_2 = y_1(z^n_1 \omega)$).

- **Step 0:** Set $n = 0$.

- **Step 1:** Sample the twisted particle $X_{n+1}^1$ from the probability
  \[
  \frac{\Phi_{z^n_1 \omega, t_{z^n_1 \omega}^{-1}}(\eta_{z^n_1 \omega}^{t_{z^n_1 \omega}^{-1}}) \left(h \left(z^n_1 \omega, x_{n+1}^1 \right) \right)}{\Phi_{z^n_1 \omega, t_{z^n_1 \omega}^{-1}}(\eta_{z^n_1 \omega}^{t_{z^n_1 \omega}^{-1}}) \left(h \left(z^n_1 \omega, \cdot \right) \right)}
  \]
  and compute the un-normalised weight:
  \[
  W \left(z^n_1 \omega, x_{n+1}^1 \right) = \mathbb{I}_{R \times B_{n+1}, \epsilon}(y_1(z^n_1 \omega)) \left(x_{n+1}^1 \right).
  \]

- **Step 2:** Set $r = 2$ and $S = 0$.

- **Step 3:** Sample the non-twisted particle $X_{n+1}^r$ from the probability
  \[
  \Phi_{z^n_1 \omega, t_{z^n_1 \omega}^{-1}}(\eta_{z^n_1 \omega}^{t_{z^n_1 \omega}^{-1}}) \left(x_{n+1}^r \right)
  \]
  and compute the un-normalised weight:
  \[
  W \left(z^n_1 \omega, x_{n+1}^r \right) = \mathbb{I}_{R \times B_{n+1}, \epsilon}(y_1(z^n_1 \omega)) \left(x_{n+1}^r \right).
  \]
  Compute $S = \sum_{i=1}^{r} W \left(z^n_1 \omega, x_{n+1}^i \right)$. If $S < N$, then set $r = r + 1$ and return to the beginning of Step 3. Otherwise, set $T_{z^n_1 \omega} = r$.

- **Step 4:** Sample $K_{n+1}$ from the discrete uniform distribution on $\{1, \ldots, t_{z^n_1 \omega} - 1\}$; this is the index of the twisted particle in Step 1. Set $n = n + 1$ and return to the start of Step 1.
densities of Algorithm 33:

\[ \tilde{M}_{\theta}^{T_{\omega}^{-1},T_{z\omega}^{-1}}(\omega, x, D) = \frac{\int_{D} M_{\theta}^{T_{\omega}^{-1},T_{z\omega}^{-1}}(\omega, x, dx') h_{T_{z\omega}^{-1}}(z\omega, x')} {\int_{E^{T_{z\omega}^{-1}}} M_{\theta}^{T_{\omega}^{-1},T_{z\omega}^{-1}}(\omega, x, du) h_{T_{z\omega}^{-1}}(z\omega, u)} , \]

for all \( D \in \mathcal{E}^{(T_{z\omega}^{-1})} \), and \( \tilde{M}_{\theta}^{T_{\omega}^{-1},T_{z\omega}^{-1}} \) is a member of \( \tilde{M}_{T_{\omega}^{-1},T_{z\omega}^{-1}} \). Of course, any generic choice of the function \( h \) is not guaranteed to induce a low variance for (4.2.3). Section 4.3 states the unique optimal choice of \( h \) which leads to the low variance.

### 4.3 Optimal change in measure

The purpose of introducing the change of measure on the alive particle filter is to reduce the variance of the algorithm’s estimate of the normalising constant, \( \hat{Z}_{\theta} \). More specifically, we would like to achieve

\[
\frac{1}{n} \log \tilde{V}_{\theta,n}(\omega) = \frac{1}{n} \log \left( \frac{\mathbb{E}[\hat{Z}_{\theta,1:n}^2]}{Z_{\theta,1:n}^2} \right) \rightarrow \Upsilon(\tilde{M}_{\theta}) = 0 \quad \text{as} \quad n \rightarrow \infty, \quad \mathbb{P} - a.s.,
\]

which is similar to expression (2.12.1) of Section 2.12. The non-negative, finite constant \( \Upsilon(\tilde{M}_{\theta}) \) is some limiting value which depends on the transition density of Algorithm 33.

We show below that the optimal choice of \( h \) which leads to \( \Upsilon(\tilde{M}_{\theta}) = 0 \) is

\[
h(\omega, x) = \lim_{n \rightarrow \infty} \frac{Q_{\theta,n}^\omega(1)(x)}{\Phi_{\theta,n}^{\omega,n}(\sigma) Q_{\theta,n}^\omega(1)}, \quad (4.3.1)
\]

which also happens to be an eigenfunction and the unique solution to the system of equations

\[
\eta^\omega Q_{\theta}^\omega(\cdot) = \lambda_{\omega} \eta^\omega(\cdot) \quad (4.3.2)
\]

\[
Q_{\theta}^\omega(h(z\omega, \cdot))(x) = \lambda_{\omega} h(\omega, x)
\]

\[
\eta^\omega(h(\omega, x)) = 1,
\]
4.3 Optimal change in measure

for the limit

$$\eta^\omega (A) = \lim_{n \to \infty} \Phi_{\theta,n}^{z^n} (\sigma) (A)$$

and the $\mathcal{F}$ -measurable eigenvalue

$$\lambda : \omega \in \Omega \to \eta^\omega (W^\omega).$$

A similar result has been proven before in [91]. However, that work only considers the case where $M_\theta (\omega, x, dx)$ can be evaluated pointwise (i.e., the likelihood density $g_\theta (u(z\omega) \mid dk)$ can be computed). The result in this thesis extends the work of [91] to the case where that likelihood density is not computable.

To prove our result, we adopt slightly different assumptions from those of [91]:

(B1) The shift operator $z$ preserves $\mathbb{P}$ and is ergodic.

(B2) At any time point,

$$\sup_{\omega \in \Omega} \sup_{(x,u) \in E^2} M_\theta^\omega (W) (x) \leq \Delta_1,$$

for some $\Delta_1 \in (0, \infty)$. Furthermore, there exist positive, finite constants $(\epsilon_-, \epsilon_+)$ and a probability measure $\nu \in \mathcal{P}(E)$ such that

$$\nu (\cdot) \epsilon_- \leq M_\theta (\omega, x, \cdot) \leq \epsilon_+ \nu (\cdot) \quad \forall (\omega, x) \in \Omega \times E.$$  \hspace{1cm} (4.3.4)

Given the definition of the incremental weights, note that (4.3.3) and (4.3.4) imply, for all $\sigma, \sigma_1, \sigma_2 \in \mathcal{P}(E)$,

$$\sup_{n \geq 1} \sup_{\omega \in \Omega} \sup_{(x,u) \in E^2} \frac{\sigma_1 Q_{\theta,n}^\omega (x)}{\sigma_2 Q_{\theta,n}^\omega (u)} \leq \Delta_2, \quad 0 < \sup_{n \geq 1} \sup_{\omega \in \Omega} \sigma Q_{\theta,n}^\omega (1) < \infty,$$

$$0 < \sup_{n \geq 1} \sup_{(x,u) \in E^2} \frac{Q_{\theta,n}^\omega (1) (x)}{Q_{\theta,n}^\omega (1) (u)} < \infty$$

for some $\Delta_2 \in (0, \infty)$.

(B3) We always fix $N$ such that $1 < N < \infty$, and for all $\omega \in \Omega$, $\epsilon$ is always set in such a way that $T_\omega$ as in (4.2.1) is finite.
Essentially, the first assumption means that the process producing the observations is stationary and ergodic [91]. The second and third assumptions effectively place upper and lower bounds on the estimate of the normalising constant, and they place a finite restriction on the running time of Algorithm 33. We acknowledge that these assumptions may be difficult to verify in a real world example. Nonetheless, they allow us to prove an optimality result which gives us a sense of how twisting the alive particle filter might be beneficial. We note that the algorithms presented in this chapter are still valid regardless of whether the assumptions hold (although they might not be optimal).

In the chapter appendix in Section A, the following theorem is proven for when \( h \) is defined as in (4.3.1):

**Theorem 4.3.1** Assume (B1), (B2) and (B3), and let \( h \) be defined as in (4.3.1). For each \( \tilde{M}^{T_0,T_ω−1}_θ \) any member of a \( \tilde{M}^{T_0,T_ω−1}_θ \), the following are equivalent:

1. \( \Upsilon (\tilde{M}_θ) = 0 \).

2. For \( \mathbb{P} \)–almost all \( ω \in Ω \), \( \exists A_ω \in \mathcal{E}^{⊗(T_ω−1)} \) such that \( \nu^{⊗(T_ω−1)} (A_ω^c) = 0 \) and \( ∀x \in A_ω \),

\[
\tilde{M}^{T_0,T_ω−1}_θ (ω, x, D) = \frac{\int_D M^{T_0,T_ω−1}_θ (ω, x, dx')}{\int_{E^{⊗(T_ω−1)}} M^{T_0,T_ω−1}_θ (ω, x, du) h^{T_ω−1} (zω, x')} \]  

for all \( D \in \mathcal{E}^{⊗(T_ω−1)} \).

3. For \( \mathbb{P} \)–almost all \( ω \in Ω \), \( sup_n \tilde{V}_n < \infty \).

This theorem states that there is a unique choice for the change in measure of the particle system that, when analytically available, leads to \( \Upsilon (\tilde{M}_θ) = 0 \). However, that optimal \( h \) often needs to be approximated. In the following section, we implement Algorithm 33 on an example where the exact form of \( h \) needs to be approximated. The numerical illustration shows that under certain scenarios, the approximation of \( h \) is sufficient to reduce the variance of \( \hat{Z}_θ \).
4.4 Implementation of alive twisted SMC on a linear Gaussian model

We compare the variability of the alive particle filter’s normalising constant estimator (2.10.2) to that of the alive twisted particle filter’s normalising constant estimator (4.2.3). We consider the linear Gaussian HMM of [91, Section 4.4]:

\[
X_0 \sim \mathcal{N}(0, \nu^2) \tag{4.4.1}
\]

\[
X_{n+1} \mid (X_{1:n} = x_{1:n}, Y_{1:n} = y_{1:n}) \sim \mathcal{N}(0.9x_n, \nu^2) = f_\theta(x_{n+1} \mid x_n)
\]

\[
Y_n \mid (X_{1:n} = x_{1:n}, Y_{1:n-1} = y_{1:n-1}) \sim \mathcal{N}(x_n, \tau^2) = g_\theta(y_n \mid x_n).
\]

In our numerical illustrations, we assume it is undesirable to repetitively calculate the density \(g_\theta\) to obtain exact importance weights, but it is possible to simulate from the corresponding distribution.

We know from [91] that the best approximation of \(h\) appropriate for a twisted bootstrap particle filter targeting an HMM is

\[
h(z^{n-1} \omega, x_n) = \pi_\theta(y_1(z^{n-1+l} \omega) \mid x_n) = \pi_\theta(y_n(z^l \omega) \mid x_n),
\]

where \(l \geq 0\) is a lag length. As this expression is analytically available for 4.4.1, we use this \(h\) in our simulations. [91, Section 4.4] illustrates that the ideal lag length appears to be five for this example (at least when \(\nu = \tau = 1\)), and so we use \(l = 5\). Furthermore, given this form of \(h\), it is possible to obtain the closed form expression

\[
Q_{z^{n-1} \omega}^n (h(z^n \omega, \cdot)) (x_n^{a(i)}) = \mathbb{I}_{R \times B_{n+1,\cdot}(y_1(z^n \omega))) (\cdot) \pi_\theta(y_1(z^{n+l} \omega) \mid x_n^{a(i)})
\]

In our analysis, we calculate

\[
\log \left[ \mathbb{V} \left[ \hat{Z}_{\theta,1:T}^{\text{Algo15}} \right] \right] - \log \left[ \mathbb{V} \left[ \hat{Z}_{\theta,1:T}^{\text{Algo33}} \right] \right]
\]

for different pairs of values of the state noise \(\nu\) and the observation noise \(\tau\), where the variance is taken with respect to the appropriate algorithm. We first consider values for \(\nu\) and \(\tau\) that range from one to 84 in steps of 14, and we run 40 simulations per pair \((\nu, \tau)\).
The experiment is repeated under different values of $N$ for a fixed $T = 100$. We also consider different values of $\epsilon$, which is defined as

$$\epsilon = \frac{|u_n - y_1(z^{n-1}\omega)|}{|y_1(z^{n-1}\omega)|} = \frac{|u_n - y_n(\omega)|}{|y_n(\omega)|},$$

with $u$ being the simulated data.

As the below output shows, the most striking differences between the algorithms’ performances in the tests just described appear when $N$ is large and when $\nu = \tau = 1$. To better understand the behaviour of the filters in this specific case, we run further simulations where values for $(\nu, \tau)$ remain concentrated around $(1, 1)$; see below.

**Results**

- $N = 100$, $\epsilon = 0.25$, broad grid: see Figure 4.4.1
- $N = 100$, $\epsilon = 0.75$, broad grid: see Figure 4.4.1
- $N = 100$, $\epsilon = 1.5$, broad grid: see Figure 4.4.1
- $N = 100$, $\epsilon = 3.5$, broad grid: see Figure 4.4.1
- $N = 1000$, $\epsilon = 1.5$, broad grid: see Figure 4.4.2
- $N = 1500$, $\epsilon = 3.5$, broad grid: see Figure 4.4.2
- $N = 750$, $\epsilon = 1.5$, concentrated grid: see Figure 4.4.3
- $N = 1250$, $\epsilon = 1.5$, concentrated grid: see Figure 4.4.3
- $N = 1500$, $\epsilon = 1.5$, concentrated grid: see Figure 4.4.3

In Figure 4.4.1, we first observe that changing $\epsilon$ has little effect on the output, despite the fact that the twisted algorithm twists its proposal more as $\epsilon$ increases. More twisting occurs also when $\nu$ is large relative to $\tau$, but again, without any real impact on the output.

Secondly, comparing Figures 4.4.1 and 4.4.2 shows that increasing $N$ does seem to have some effect on the output. When $N$ is large, the alive twisted algorithm outperforms
the alive algorithm by several orders of magnitude in the case where \( \nu = \tau = 1 \). When we focus the simulations around the case where \( N \) is large and \( \nu \) and \( \tau \) each remain close to one (see Figure 4.4.3), we see that the best performance happens when \( \nu \) and \( \tau \) are both less than or equal to one. Thus, these results do suggest that twisting can improve the performance of the alive particle filter when a) the variances of the HMM are low and b) one has access to a good approximation of the true \( h \) (i.e., when \( \pi_\theta (y_n(z^k_\omega) \mid x_n) \) can be computed or closely approximated). Granted, the subset of HMMs whose observations have unknown or intractable likelihood densities and have a known, useful approximation for \( \pi_\theta (y_n(z^5_\omega) \mid x_n) \) is small, but examples do exist. In Section 4.6, we consider an HMM whose observations are drawn from a stable distribution with stability parameter close to two (a setting under which the stable distribution is Gaussian and \( \pi_\theta (y_n(z^5_\omega) \mid x_n) \) can easily be computed).

Finally, comparing the actual formulas (2.10.2) and (4.2.3) more closely explains why the alive twisted algorithm outperforms the alive algorithm more as \( N \) increases. The two expressions are identical, except for this additional incremental term in (4.2.3):

\[
\frac{\sum_{i=1}^{T_{z_k^\omega} - 1} Q_\theta^{z_k^\omega - 1} \omega \left( h(z_k^\omega, \cdot) \right) \left( x_k^{a(i)} \right)}{\sum_{i=1}^{T_{z_k^\omega} - 1} W \left( z_k^\omega, x_{k+1}^i \right) h \left( z_k^\omega, x_{k+1}^i \right)}.
\]  

(4.4.2)

Of the \((T_{z_k^\omega} - 1)\) simulated paths, suppose that we were to order the paths so that those with zero weight were all placed after the \((N - 1)\) paths with non-zero weight. Then, as we are using incremental weights that equal either zero or one, (4.4.2) is effectively equivalent to

\[
\frac{\sum_{i=1}^{N-1} Q_\theta^{z_k^\omega - 1} \omega \left( h(z_k^\omega, \cdot) \right) \left( x_k^{a(i)} \right)}{\sum_{i=1}^{N-1} h \left( z_k^\omega, x_{k+1}^i \right)} = \frac{1}{N-1} \sum_{i=1}^{N-1} Q_\theta^{z_k^\omega - 1} \omega \left( h(z_k^\omega, \cdot) \right) \left( x_k^{a(i)} \right) \frac{1}{N-1} \sum_{i=1}^{N-1} h \left( z_k^\omega, x_{k+1}^i \right).
\]

We could think of the numerator and the denominator as Monte Carlo estimates of some integrals that are only getting more accurate as \( N \) increases, thereby reducing the variance of (4.2.3) even more relative to (2.10.2) as \( N \) increases.
Figure 4.4.1: Each graph measures the difference between the log of the variance of estimates of $Z_{\theta,1:T}$ obtained by the two algorithms. We set $T = 100$ and $N = 100$ in each case. Each block represents 40 simulations. The weights have an $\epsilon$ of 0.25 (top left), 0.75 (top right), 1.5 (bottom left), and 3.5 (bottom right). Twisting ranges from happening 10% of the time when $\epsilon = 0.25$ to 90% of the time when $\epsilon = 3.5$. 
4.4 Implementation of alive twisted SMC on a linear Gaussian model

\[
\log \left[ \mathbb{V} \left[ \hat{Z}_{\theta,1:T}^{\text{Algo15}} \right] \right] - \log \left[ \mathbb{V} \left[ \hat{Z}_{\theta,1:T}^{\text{Algo33}} \right] \right]
\]

Figure 4.4.2: Each graph measures the difference between the log of the variance of estimates of \( Z_{\theta,1:T} \) obtained by the two algorithms. We set \( T = 100 \) in each case. Each block represents 40 simulations. Left: \( N = 1000 \) and the weights have an \( \epsilon \) of 1.5. Right: \( N = 1500 \) and the weights have an \( \epsilon \) of 3.5.
Figure 4.4.3: Each graph measures the difference between the log of the variance of estimates of $Z_{\theta,1:T}$ obtained by the two algorithms. We set $T = 100$ and $\epsilon = 1.5$ in each case. Each block represents 300 simulations, where $N = 750$ at top left, $N = 1250$ at top right, and $N = 1500$ at bottom. Notice that the bottom plots are more concentrated around $\nu = \tau = 1$ than those plots at top left and top right.

### 4.5 Alive twisted particle marginal Metropolis-Hastings

The previous section shows that Algorithm 33 can indeed reduce the variance of unbiased estimates of the normalising constant in certain scenarios. Thus, following the same strategy from Chapter 3, it is sensible to embed the alive twisted particle filter in PMMH to attempt to expedite its convergence.
4.5 Alive twisted particle marginal Metropolis-Hastings

With an unbiased estimate of the normalising constant available, it is straightforward to define an alive twisted PMMH as Algorithm 34, whose extended target density \( \pi^N \) is structured as follows. The joint density of the simulated variables through time \( n \) of Algorithm 33 is

\[
\psi_\theta(x_{1:n}, a_{1:n-1}, t_\omega, \ldots, t_{zn-1}) \propto \left[ \left( \frac{t_\omega - 1}{N - 1} \right) \prod_{i=1; i \neq k_n}^{t_\omega} M_\theta \left( z_{\omega-1}^i, x_0^i, x_1^i \right) \right] \times
\left[ \prod_{j=2}^{n} \left( \frac{t_{z_j-1}^j - 1}{N - 1} \right) \prod_{i=1; i \neq k_j}^{t_{z_j-1}^j} \frac{W \left( z_{j-2}^j, x_{j-1}^a \right) M_\theta \left( z_{j-2}^j, x_{j-1}^a, x_j^1 \right)}{\sum_{l=1}^{t_{z_j-1}^j-1} W \left( z_{j-2}^j, x_{j-1}^a \right)} \right] \times
\left[ M_\theta \left( z_{\omega}^0, x_k^a, x_1^k \right) h(\omega, x_k^1) \right] \times
\left[ \prod_{j=2}^{n} Q_\theta^{z_j-2}( \cdot | x_{j-1}^a, x_j^1 ) M_\theta \left( z_{j-2}^j, x_{j-1}^a, x_j^1 \right) h \left( z_{j-1}^j, x_j^1 \right) \right],
\]

where we slightly abuse the notation and use \( a_{1:n-1} \) to denote the full ancestry of the twisted and non-twisted particles (note also that \( k_n \) is being used to denote the index of the twisted particle at time step \( n \)). One can use this expression to establish an extended target as

\[
\pi^N \left( k, \theta, x_{1:n}, a_{1:n-1}, t_\omega, \ldots, t_{zn-1} \mid y_{1:n} \right) \propto \frac{\pi(\theta) \psi_\theta(x_{1:n}, a_{1:n-1}, t_\omega, \ldots, t_{zn-1}) \times W \left( z_{n-1}^n, x_n^k \right)}{\sum_{l=1}^{t_{n-1}} \frac{W \left( z_{n-1}^n, x_n^k \right)}{W \left( z_{n-1}^n, x_n^l \right)}},
\]

where \( \pi(\theta) \) is an appropriate prior for the parameter \( \theta \). In both of the above expressions, it is assumed that, at any time step \( k \), each sample \((x_k, t_{z_{k-1}})\) satisfies the following:

\[
\sum_{l=1}^{t_{z_{k-1}}-1} W \left( z_{k-1}, x_l^k \right) = N - 1 \quad \text{and} \quad W \left( z_{k-1}, x_{z_{k-1}}^k \right) = 1.
\]
Algorithm 34 Alive twisted PMMH for HMMs

- **Step 0:** Draw $\theta$ from its prior. All remaining random variables can be sampled from their full conditionals defined by the target (4.5.1):
  - Sample $\mathcal{Z}_{1:n}, a_{1:n-1}, t_\omega, \ldots, t_{z-1}\omega \mid \cdots$ via Algorithm 33 using parameter value $\theta$.
  - Choose $k$ with probability $\frac{W(z_{n-1}\omega, x_k^n)}{\sum_{l=1}^{t_{z-1}\omega} W(z_{n-1}\omega, x_l^n)}$.

Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta,1:n}$, via (4.2.3).

- **Step 1:** Sample $\theta^* \sim q(\cdot \mid \theta)$. All remaining random variables can be sampled from their full conditionals defined by the target (4.5.1):
  - Sample $\mathcal{Z}_{1:n}^*, a_{1:n-1}^*, t_\omega^*, \ldots, t_{z-1}\omega^* \mid \cdots$ via Algorithm 33 using parameter value $\theta^*$.
  - Choose $k^*$ with probability $\frac{W(z_{n-1}\omega, x_k^n)}{\sum_{l=1}^{t_{z-1}\omega} W(z_{n-1}\omega, x_l^n)}$.

Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta^*,1:n}$, via (4.2.3).

- **Step 2:** With acceptance probability
  $\frac{1}{\pi(\theta^*) q(\theta \mid \theta^*) \hat{Z}_{\theta^*,1:n}}$ $\hat{Z}_{\theta,1:n} / \pi(\theta) q(\theta \mid \theta)$ set $k = k^*$, $\theta = \theta^*$, $\mathcal{Z}_{1:n} = \mathcal{Z}_{1:n}^*$, $a_{1:n-1} = a_{1:n-1}^*$, and $t_\omega, \ldots, t_{z-1}\omega = t_\omega^*, \ldots, t_{z-1}\omega^*$. Return to the beginning of Step 1.

Similarly, the proposal density of the PMMH takes the form

$$q^N(k, \theta, \mathcal{Z}_{1:n}, a_{1:n-1}, t_\omega, \ldots, t_{z-1}\omega) \propto q(\theta \mid \zeta) \psi_\theta(\mathcal{Z}_{1:n}, a_{1:n-1}, t_\omega, \ldots, t_{z-1}\omega) 
\frac{W(z_{n-1}\omega, x_k^n)}{\sum_{l=1}^{t_{z-1}\omega} W(z_{n-1}\omega, x_l^n)},$$

where $q(\theta \mid \zeta)$ is the density that proposes a new value $\theta \in \Theta$ conditional on a current accepted value $\zeta \in \Theta$. 


4.6 Implementation of alive twisted PMMH on a stochastic volatility model

In the next numerical illustration, we compare the convergence of Algorithm 34 to that of a PMMH employing the non-twisted alive particle filter; the PMMH employing the non-twisted alive particle filter first appeared in [51]. We consider a stochastic volatility model which is similar to the one appearing in [51]:

\[ X_0 \sim \mathcal{N}(0, \nu^2) \]  
\[ X_{n+1} \mid (X_{1:n} = x_{1:n}, Y_{1:n} = y_{1:n}) \sim \mathcal{N}(Fx_n, \nu^2) = f_\theta(x_{n+1} \mid x_n) \]  
\[ Y_n \mid (X_{1:n} = x_{1:n}, Y_{1:n-1} = y_{1:n-1}) \sim \exp\left(\frac{x_n}{2}\right)S(\alpha, 0.05, \gamma, 0). \]

This model is more challenging than the linear Gaussian HMM (4.4.1) because the probability density functions of the observations are not defined for all parameter values of the stable distribution. However, the stable distribution is Gaussian when the stability parameter is \( \alpha = 2 \). Thus, this section uses the same approximation for \( h \) that was used in Section 4.4, and when calculating

\[ h(z^{n-1}\omega, x_n) = \pi_\theta(y_1(z^{n-1+l}\omega) \mid x_n)\{l=5\} = \pi_\theta(y_n(z^l\omega) \mid x_n)\{l=5\}, \]

we use a Gaussian density as an approximation for the density of the observations.

The observations are daily logarithmic returns of the S&P 500. We consider three datasets that each begin with 10th December 2009 and run for \( T = 200 \), \( T = 500 \), or \( T = 700 \) time steps (see Figure 4.6.1). The datasets are chosen for their different time lengths, to study how \( T \) affects the relative performance of the algorithms.
Both PMMH algorithms are used to infer the scalars $F$, $\frac{1}{\nu^2}$, and $\frac{1}{\gamma}$ (with priors $F \sim \mathcal{N}(0, 0.15)$, $\frac{1}{\nu^2} \sim \mathcal{G}(2, 100)$, and $\frac{1}{\gamma} \sim \mathcal{G}(2, 1)$). We set the number of alive particles $N$, the number of PMMH iterations $M$, and the stability parameter $\alpha$ to different values across simulations, with a fixed $\epsilon = 3.5$ (see below). In order to run the algorithms under many different settings in a timely fashion, we only repeat five simulations of each algorithm per trial. All algorithms have approximately equal running times for equal values of $N$. We track the convergence of the algorithms using the autocorrelation functions (ACFs) and the trace plots of $F$, $\nu^2$, and $\gamma$ (see Section 2.14.3).

Results: trial 1

The data has a length of $T = 200$, and the stability parameter is set as $\alpha = 1.75$. The proposals for the parameters $\nu^2$ and $\gamma$ are log normal random walks: $\log(\theta^*) = \log(\theta) + \iota$, $\iota \sim \mathcal{N}(0, 0.5)$. The proposal for $F$ is a normal random walk: $\theta^* \sim \mathcal{N}(\theta, 1)$.

- Alive PMMH, $N = 50$, $M = 100000$: see Figure 4.6.2
- Alive twisted PMMH, $N = 50$, $M = 100000$: see Figure 4.6.2
- Alive PMMH, $N = 100$, $M = 100000$: see Figures 4.6.2 and 4.6.3
- Alive twisted PMMH, $N = 100$, $M = 100000$: see Figures 4.6.2 and 4.6.3
4.6 Implementation of alive twisted PMMH on a stochastic volatility model

- Alive PMMH, $N = 1000, M = 20000$: see Figure 4.6.2
- Alive twisted PMMH, $N = 1000, M = 20000$: see Figure 4.6.2

Both algorithms seem to perform similarly. The ACF plots (see Figure 4.6.2) demonstrate that the alive twisted PMMH converges more slowly than the alive PMMH. This issue is likely a result of the eigenfunction $h$ being a poor approximation in the case where $\alpha = 1.75$. However, the trace plots (see Figure 4.6.3) show that both algorithms explore the state spaces of the unknown parameters thoroughly and without sticking too much.

![Autocorrelation graphs](image)

Figure 4.6.2: Results for $N = 50$ (left), $N = 100$ (middle), and $N = 1000$ (right). ACF plots for $F$, $\nu^2$, and $\gamma$ (from top to bottom). The alive PMMH corresponds to the blue lines, and the alive twisted PMMH corresponds to the red lines.
Chapter 4. Twisting the Alive Particle Filter

Figure 4.6.3: Results for $N = 100$. Left: alive PMMH; right: alive twisted PMMH. Trace plots for $F$, $\log \nu$, and $\log \gamma$ (from top to bottom). Only one of the five repetitions of each simulation is shown.
Results: trial 2

The data has a length of $T = 200$, and the stability parameter is set as $\alpha = 1.95$. The proposals for the parameters $\nu^2$ and $\gamma$ are log normal random walks: $\log(\theta^*) = \log(\theta) + \iota$, $\iota \sim \mathcal{N}(0, 0.5)$. The proposal for $F$ is a normal random walk: $\theta^* \sim \mathcal{N}(\theta, 1)$.

- Alive PMMH, $N = 50$, $M = 100000$: see Figure 4.6.4
- Alive twisted PMMH, $N = 50$, $M = 100000$: see Figure 4.6.4
- Alive PMMH, $N = 100$, $M = 100000$: see Figures 4.6.4 and 4.6.5
- Alive twisted PMMH, $N = 100$, $M = 100000$: see Figures 4.6.4 and 4.6.5
- Alive PMMH, $N = 1000$, $M = 20000$: see Figure 4.6.4
- Alive twisted PMMH, $N = 1000$, $M = 20000$: see Figure 4.6.4

Both algorithms seem to perform similarly, although now the ACF plots (see Figure 4.6.4) show the alive twisted PMMH slightly outperforming the non-twisted alive PMMH. It appears our eigenfunction $h$ is a fair approximation to the true, optimal $h$ when $\alpha = 1.95$; this is not surprising, as when $\alpha = 1.95$ the stable distribution is nearly Gaussian, and we know from [91] and from Section 4.4 that our eigenfunction $h$ is a good approximation for linear Gaussian HMMs. It is also not surprising to see a boost in performance when $N = 1000$, as this is a scenario in Section 4.4 where the alive twisted SMC performed much better than the alive SMC. Finally, the trace plots (see Figure 4.6.5) again show that both algorithms explore the state spaces of the unknown parameters thoroughly.

The output of Trials 1 and 2 illustrate that if the true, optimal $h$ cannot be well approximated, then twisting alive PMMH is not useful. In this particular numerical example, that means twisting will only be useful when the HMM (4.6.1) is nearly linear Gaussian. The performance of the twisted algorithm when $\alpha = 1.95$ versus when $\alpha = 1.75$ also suggests that the twisted scheme will continue to perform worse as the model moves further away from the Gaussian settings. Thus, the range of models to which our method may be applicable seems limited to a small subset. Given that implementing the alive twisted PMMH is not much more difficult than implementing the standard alive PMMH, however, one could
argue that the twisted algorithm is still a useful tool when one is working within that small subset of HMMs.

![Autocorrelation plots](image)

Figure 4.6.4: Results for $N = 50$ (left), $N = 100$ (middle), and $N = 1000$ (right). ACF plots for $F$, $\nu^2$, and $\gamma$ (from top to bottom). The alive PMMH corresponds to the blue lines, and the alive twisted PMMH corresponds to the red lines.
Figure 4.6.5: Results for $N = 100$. Left: alive PMMH; right: alive twisted PMMH. Trace plots for $F$, $\log \nu$, and $\log \gamma$ (from top to bottom). Only one of the five repetitions of each simulation is shown.
Results: trial 3

The data has a length of $T = 500$, and the stability parameter is set as $\alpha = 1.75$. The proposals for the parameters $\nu^2$ and $\gamma$ are log normal random walks: $\log(\theta^*) = \log(\theta) + \iota$, $\iota \sim \mathcal{N}(0, 0.5)$. The proposal for $F$ is a normal random walk: $\theta^* \sim \mathcal{N}(\theta, 1)$.

- Alive PMMH, $N = 50$, $M = 100000$
- Alive twisted PMMH, $N = 50$, $M = 100000$
- Alive PMMH, $N = 100$, $M = 100000$
- Alive twisted PMMH, $N = 100$, $M = 100000$

The two PMMH algorithms show similar performance to one another. There is no significant benefit to twisting the embedded SMC algorithm in this case. As the results are similar to those of Trial 1, it appears to be that increasing $T$ slightly from $T = 200$ to $T = 500$ has little effect on the output (we do not present the output for this reason).

Results: trial 4

The data has a length of $T = 500$, and the stability parameter is set as $\alpha = 1.95$. The proposals for the parameters $\nu^2$ and $\gamma$ are log normal random walks: $\log(\theta^*) = \log(\theta) + \iota$, $\iota \sim \mathcal{N}(0, 0.5)$. The proposal for $F$ is a normal random walk: $\theta^* \sim \mathcal{N}(\theta, 1)$.

- Alive PMMH, $N = 50$, $M = 100000$
- Alive twisted PMMH, $N = 50$, $M = 100000$
- Alive PMMH, $N = 100$, $M = 100000$
- Alive twisted PMMH, $N = 100$, $M = 100000$

The results (not shown) are similar to those of Trial 2, which corroborates the hypothesis that increasing $T$ slightly from $T = 200$ to $T = 500$ has little effect on the output. This point is even further exemplified by the results of Trials 5 and 6 below.
Results: trial 5

The data has a length of $T = 700$, and the stability parameter is set as $\alpha = 1.75$. The proposals for the parameters $\nu^2$ and $\gamma$ are log normal random walks: $\log(\theta^*) = \log(\theta) + \iota$, $\iota \sim \mathcal{N}(0, 0.5)$. The proposal for $F$ is a normal random walk: $\theta^* \sim \mathcal{N}(\theta, 1)$.

- Alive PMMH, $N = 50$, $M = 100000$
- Alive twisted PMMH, $N = 50$, $M = 100000$
- Alive PMMH, $N = 100$, $M = 100000$
- Alive twisted PMMH, $N = 100$, $M = 100000$

The results are similar to those of Trial 1, and thus not shown.

Results: trial 6

The data has a length of $T = 700$, and the stability parameter is set as $\alpha = 1.95$. The proposals for the parameters $\nu^2$ and $\gamma$ are log normal random walks: $\log(\theta^*) = \log(\theta) + \iota$, $\iota \sim \mathcal{N}(0, 0.5)$. The proposal for $F$ is a normal random walk: $\theta^* \sim \mathcal{N}(\theta, 1)$.

- Alive PMMH, $N = 50$, $M = 100000$
- Alive twisted PMMH, $N = 50$, $M = 100000$
- Alive PMMH, $N = 100$, $M = 100000$
- Alive twisted PMMH, $N = 100$, $M = 100000$

The results are similar to those of Trial 2, and thus not shown.

4.7 Discussion

The authors of [91] show numerically (and also prove theoretically) that with a good approximation of (4.3.1) available, twisting can significantly reduce the variance of the unbiased SMC estimate of $Z_0$ of an HMM whose observations have tractable likelihood densities. In this chapter, we applied the same twisting concept to a particle filter that targets
Chapter 4. Twisting the Alive Particle Filter

HMMs whose observations have unknown or intractable likelihood densities. That is, we introduced a change of measure on the alive particle filter of [51] to reduce the variance of its estimate of the normalising constant. By adopting similar assumptions as in [91], we demonstrated that nearly the same theoretical arguments of [91] can be used to determine the unique, optimal change of measure for the alive algorithm. That optimal choice is (4.3.1) – i.e., the same unique choice discovered in [91].

Our theoretical findings were used to formalise an alive twisted particle filter and an alive twisted PMMH. Both methods were implemented on HMMs, with the PMMH being used in a real world example. The numerical analyses illustrated that when the change of measure on the alive algorithms is not a close approximation of the ideal change in measure, twisting may not be worthwhile. However, when a very good approximation of the ideal change in measure was available, our algorithms did exhibit reduced variability. In Section 4.4, the approximation of (4.3.1) is very close to the optimal measure, and we saw a significant reduction in the twisted algorithm’s variance when the variances of the underlying linear Gaussian model were themselves very low. In Section 4.6, the approximation of (4.3.1) was less close to the true measure, and we saw evidence of only a modest boost in the performance of the twisted algorithm over the non-twisted algorithm; furthermore, any observable benefits were quickly erased as the approximation of (4.3.1) became just slightly worse.

We know generally that \( \pi_{\theta}(y_n(\omega^5) \mid x_n) \) is a good approximation for (4.3.1) for the class of HMMs which were studied in this chapter. However, the subset of HMMs whose observations have unknown or intractable likelihood densities and have a known, useful approximation for \( \pi_{\theta}(y_n(\omega^5) \mid x_n) \) is probably small. So far, we do know that when the underlying distributions of the HMM are nearly Gaussian (e.g., stable distributions with stability parameters that are very close to two), a useful approximation for \( \pi_{\theta}(y_n(\omega^5) \mid x_n) \) is known (see Section 4.6). We currently do not know of any other HMMs (whose observations have unknown or intractable likelihood densities) outside of this subset for which a good twisting measure has been tested.

We close by acknowledging that the assumptions used to prove our theoretical results may be difficult (or even impossible) to verify in a real world application. Assumption (B1), in particular, would be very hard to verify when one knows little about the process
producing the observations. A future work might consider proving the same results under weaker assumptions, although that will likely not be a straightforward task.

Appendix

A Proof of the main result from Section 4.3

We first illustrate that $Z_{\theta,1,n}$ is actually finite in the limit as $n \to \infty$, for otherwise there would be no circumstance under which $\Upsilon \left( \hat{M}_\theta \right) = 0$. The result is established as the following proposition.

**Proposition A.1** Assume (B1) and (B2). There exists a finite, real-valued constant $\Lambda$, which is independent of the initial distribution $\mu$, such that

$$
\frac{1}{n} \log \mu Q_{\theta,n}^\omega (1) \to \Lambda
$$

as $n \to \infty$, $\mathbb{P}$—almost surely.

**Proof of Proposition A.1** This proof closely follows the proof of [91, Proposition 1], with only some minor modifications. Assuming (B2), we can define a constant $g = \inf_{(\omega,x)} \nu Q_{\theta}^\omega (x) > 0$ which also must be finite. Consider a sequence of random variables \( \{\kappa_n^\omega\}_{n \geq 1} \) where

$$
\kappa_n^\omega = \nu Q_{\theta,n-1}^\omega (1) g.
$$

We know $\kappa_n^\omega > 0$, and as

$$
\kappa_{n+p}^\omega = \nu Q_{\theta,n-1}^\omega (1) g \\
= \nu Q_{\theta,n-1}^\omega Q_{\theta,n-1}^\omega (1) g \\
= \nu Q_{\theta,n-1}^\omega Q_{\theta,n-1}^\omega (1) g \\
\geq \kappa_n^\omega \kappa_n^\omega,
$$

we have

$$
- \log \kappa_{n+p}^\omega \leq - \log \kappa_n^\omega - \log \kappa_n^\omega.
$$

(A1)
Furthermore, (B2) and the definition of $g$ ensure that each $\kappa_n$ is finite, and so

$$\int_{\Omega} - \log \kappa_n^\omega \mathbb{P}(d\omega) > -\infty.$$  \hfill (A2)

Considering A1, A2, and the ergodicity of the shift operator assumed by (B1), we can apply Kingman’s subadditive ergodic theory [57] to obtain

$$\frac{1}{n} \log \kappa_n^\omega \to \Lambda,$$ \hfill (A3)

as $n \to \infty$, $\mathbb{P}$—almost surely, where $\Lambda$ is a finite constant.

As (B2) implies

$$0 < \frac{\kappa_n^\omega}{\mu Q_{\theta,n}^\omega (1)} \leq \frac{\nu Q_{\theta,n}^\omega (1)}{\mu Q_{\theta,n}^\omega (1)} = \frac{\nu Q_{\theta,n}^\omega Q_{\theta,n-1}^\omega (1)}{\mu Q_{\theta,n}^\omega Q_{\theta,n-1}^\omega (1)} \leq \Delta_2,$$

we have

$$\sup_{\omega \in \Omega} \left| \frac{1}{n} \log \kappa_n^\omega - \frac{1}{n} \log \mu Q_{\theta,n}^\omega (1) \right| \leq \frac{1}{n} \log \Delta_2.$$ \hfill (A4)

Considering (A3) and (A4), we find

$$\frac{1}{n} \log \mu Q_{\theta,n}^\omega (1) \to \Lambda,$$

as $n \to \infty$, $\mathbb{P}$—almost surely. $\blacksquare$

The next two propositions clearly define the triple $(\eta, h, \lambda)$ that uniquely satisfies the system of equations (4.3.2). It is not yet shown that $h$ is the optimal function by which Algorithm 33 should be twisted. Before proving that, we have to first show that the measure exists. The fact that the triple $(\eta, h, \lambda)$ uniquely satisfies (4.3.2) is used in calculations in later parts of the proof of the main result.

**Proposition A.2** Assume (B2).

1. Fixing $\sigma \in \mathcal{P}(E)$, the limits

$$\eta^\omega (A) = \lim_{n \to \infty} \Phi_{\theta,n}^{-\omega} (\sigma) (A) \quad \text{and} \quad h (\omega, x) = \lim_{n \to \infty} \frac{Q_{\theta,n}^\omega (1) (x)}{\Phi_{\theta,n}^{-\omega} (\sigma) Q_{\theta,n}^\omega (1)}$$
exist, where \( \eta^\omega(A) \) is a member of a family of probability measures, \( \eta = \{ \eta^\omega \in \mathcal{P}(E); \omega \in \Omega \} \), and \( h(\omega, x) \) is a member of a family of real-valued, \( \mathcal{F} \otimes \mathcal{E} \)-measurable functions, \( h : \Omega \times E \to \mathbb{R} \).

2. The families of probability measures and measurable functions just defined are independent of \( \sigma \), and there exist constants \( C < \infty \) and \( \rho < 1 \) such that

\[
\sup_{\omega \in \Omega} \sup_{\sigma \in \mathcal{P}(E)} \left| \Phi_{\theta,n}^{-n\omega}(\sigma) - \eta^\omega(\cdot) \right| (\varphi) \leq \sup_x |\varphi(x)| C \rho^n \tag{A5}
\]

and

\[
\sup_{\omega \in \Omega} \sup_{x \in E} \sup_{\sigma \in \mathcal{P}(E)} \left| \frac{Q_{\theta,n}^\omega(1)}{\Phi_{\theta,n}^{-n\omega}(\sigma)} Q_{\theta,n}^\omega(1) - h(\omega, x) \right| \leq C \rho^n \tag{A6}
\]

for \( \varphi \in \mathcal{B}_b(E) \) and \( n \geq 1 \).

3. The function \( \lambda : \omega \in \Omega \to \eta^\omega(W^\omega) \) is \( \mathcal{F} \)-measurable, and

\[
\sup_{(\omega,\omega') \in \Omega^2} \frac{\lambda_\omega}{\lambda_{\omega'}} < \infty, \quad \sup_{(\omega,\omega',x,x') \in \Omega^2 \times E^2} \frac{h(\omega, x)}{h(\omega', x')} < \infty. \tag{A7}
\]

4. Consider the triples that consist of (a) a family of probability measures on \((E, \mathcal{E})\) indexed by \( \Omega \), (b) an \( \mathbb{R}^+ \)-valued measurable function on \( \Omega \times E \), and (c) a measurable function on \( \Omega \). For all \( \omega \in \Omega \), the triple \((\eta, h, \lambda)\) uniquely satisfies

\[
\eta^\omega Q^\omega_\theta(\cdot) = \lambda_\omega \eta^\omega(\cdot)
\]

\[
Q^\omega_\theta(h(z\omega, \cdot))(x) = \lambda_\omega h(\omega, x)
\]

\[
\eta^\omega(h(\omega, x)) = 1.
\]

**Proof of Proposition A.2** The proof is the same as the proof of [91, Proposition 2]. Even though our assumption (B2) differs slightly from the analogue in [91], the necessary implications are the same. \( \square \)

**Proposition A.3** Assume (B1) and (B2). Then for \( \Lambda \) as in Proposition A.1 and \( \lambda \) as in
Chapter 4. Twisting the Alive Particle Filter

Proposition A.2, we have

\[ \Lambda = \mathbb{E}[\log \lambda] = \int_{\Omega} \frac{Q_{\theta}^\omega (h(z_\omega, \cdot)) (x)}{h(\omega, x)} \mathbb{P} (d\omega), \]

for any \( x \in E \).

Proof of Proposition A.3 The proof is the same as the proof of [91, Proposition 3]. It only relies on the assumptions (B1) and (B2) because it makes use of Propositions A.1 and A.2.

Now that the triple \((\eta, h, \lambda)\) is clearly defined and it is known that \(\hat{Z}_{\theta,1:n}\) is approximating a finite value in the limit as \(n \to \infty\), we can begin to establish how the optimal \(h\) affects the particle filter. That illustration begins by showing how the particle filter behaves when its transition density is any \(\tilde{M}_\theta\) (which is a member of \(\mathcal{M}\)) and not necessarily one twisted with \(h\).

The functions \(J_\theta\) and \(L_\theta\) of Definition 4.1.1 can be used to construct 4.1.3. Thus, for any \(\tilde{M}_\theta\), we establish bounds on those functions via Lemmas A.1 and A.2 below to show that

\[ \frac{1}{n} \log \tilde{Y}_{\theta,n} \to \Upsilon(\tilde{M}_\theta) \quad \text{as} \quad n \to \infty, \]

\(\mathbb{P}\)-almost surely, in Proposition A.4 below.

Lemma A.1 Assume (B2) and (B3). For all \(\omega, \omega' \in \Omega, x, x' \in E, T_\omega \geq N, \) and \(T_{\omega'} \geq N, \)

\[ \frac{W^{T_\omega-1}(\omega, x)}{W^{T_{\omega'}-1}(\omega', x')} \leq \Delta_3 \]

for some \(\Delta_3 \in (0, \infty)\). Furthermore,

\[ e^{T_{\omega-1} \nu \otimes (T_{\omega-1})} (\cdot) \leq \tilde{M}_\theta^{T_{\omega-1},T_{\omega-1}} (\omega, x, \cdot) \leq e^{T_{\omega-1} \nu \otimes (T_{\omega-1})} (\cdot). \]

Proof of Lemma A.1 Under (B3), it is clear that any

\[ W^{T_\omega-1}(\omega, x) = \frac{1}{T_\omega - 1} \sum_{j=1}^{T_\omega - 1} W(\omega, x^j) \]
is positive and finite, and so a positive and finite upper bound on
\[
\frac{W^{T_\omega^{-1}}(\omega, x)}{W^{T_{\omega'}^{-1}}(\omega', x')}
\]
is obvious.

Recalling (B2) and Definition (4.1.1), we can calculate
\[
M^{T_\omega^{-1},T_{\omega'}^{-1}}_\theta (\omega, x, dx) = \prod_{i=1}^{T_\omega^{-1}} \frac{T_\omega^{-1} \sum_{j=1}^{T_\omega^{-1}} W(\omega, x^j) \, M_\theta (\omega, x^j, dx)}{T_\omega^{-1} \sum_{j=1}^{T_\omega^{-1}} W(\omega, x^j)} \leq \prod_{i=1}^{T_\omega^{-1}} \frac{(N - 1) \epsilon \nu (dx^i)}{N - 1} = \epsilon^{T_\omega^{-1} \nu \otimes (T_{\omega'}^{-1})} (dx)
\]
Similarly, we have the lower bound
\[
M^{T_\omega^{-1},T_{\omega'}^{-1}}_\theta (\omega, x, dx) \geq \prod_{i=1}^{T_\omega^{-1}} \frac{(N - 1) \epsilon \nu (dx^i)}{N - 1} = \epsilon^{T_\omega^{-1} \nu \otimes (T_{\omega'}^{-1})} (dx).
\]

□

Lemma A.2 For any \( \omega \in \Omega \), assume (B2) and (B3), let \( \tilde{M}^{T_\omega^{-1},T_{\omega'}^{-1}}_\theta \) be any member of \( M^{T_\omega^{-1},T_{\omega'}^{-1}} \), and let \( \tilde{\nu} \) be as in the definition of \( M^{T_\omega^{-1},T_{\omega'}^{-1}} \). There exist constants \( \alpha \in (0, \infty) \) and \( (\delta_-, \delta_+) \in (0, \infty)^2 \) and a probability measure \( \sigma \in \mathcal{P}(E^{T_\omega^{-1}}) \) such that
\[
\frac{J^{T_\omega^{-1},T_{\omega'}^{-1}}_\theta (\omega, x)}{J^{T_{\omega'}^{-1},T_{\omega'}^{-1}}_\theta (\omega', x')} \leq \alpha
\]
for all \((\omega, \omega', x, x') \in \Omega^2 \times E^{T_\omega^{-1}} \times E^{T_{\omega'}^{-1}}\) and
\[
\delta_- \sigma (\cdot) \leq L^{T_\omega^{-1},T_{\omega'}^{-1}}_\theta (\omega, x, \cdot) \leq \delta_+ \sigma (\cdot)
\]
for all \((\omega, x) \in \Omega \times E^{T_\omega^{-1}}\), where
\[
\sigma (dx) \propto \left( \frac{d\nu \otimes (T_{\omega'}^{-1})}{d\tilde{\nu}} (x) \right)^2 \tilde{\nu} (dx).
\]
Proof of Lemma A.2 For any $A \in E^{\otimes(T_{\omega^{-1}})}$

$$\int_A W^{T_{\omega^{-1}}} (\omega, x)^2 \phi_{\theta}^\omega T_{\omega^{-1}, T_{\omega^{-1}}} (x, x')^2 \tilde{M}_{\theta}^{T_{\omega^{-1}, T_{\omega^{-1}}} (\omega, x, dx')} \quad (A8)$$

$$= W^{T_{\omega^{-1}}} (\omega, x)^2 \times$$

$$\int_A \left( \frac{dM_{\theta}^{T_{\omega^{-1}, T_{\omega^{-1}}} (\omega, x, \cdot)}}{d\nu^{\otimes(T_{\omega^{-1}})} (x', \theta)} \frac{d\nu^{\otimes(T_{\omega^{-1}})} (x', \theta)}{d\tilde{\nu}^{\otimes(T_{\omega^{-1}})} (x', \theta)} \right)^2 \times$$

$$\tilde{M}_{\theta}^{T_{\omega^{-1}, T_{\omega^{-1}}} (\omega, x, dx')} \leq \sup_{(\omega', z) \in \Omega \times E^{T_{\omega^{-1}}}} W^{T_{\omega^{-1}}} (\omega', z)^2 \left( \frac{\Delta_3 T_{\omega^{-1}}}{\tilde{\epsilon}_+} \right)^2 \tilde{\epsilon}_+ \int_A \left( \frac{d\nu^{\otimes(T_{\omega^{-1}})} (x', \theta)}{d\tilde{\nu}^{\otimes(T_{\omega^{-1}})} (x', \theta)} \right)^2 \tilde{\nu} (dx' < \infty,$$

by Lemma A.1 and Definition (4.1.1). Similarly,

$$\int_A W^{T_{\omega^{-1}}} (\omega, x)^2 \phi_{\theta}^\omega T_{\omega^{-1}, T_{\omega^{-1}}} (x, x')^2 \tilde{M}_{\theta}^{T_{\omega^{-1}, T_{\omega^{-1}}} (\omega, x, dx')} \quad (A9)$$

$$= W^{T_{\omega^{-1}}} (\omega, x)^2 \times$$

$$\int_A \left( \frac{dM_{\theta}^{T_{\omega^{-1}, T_{\omega^{-1}}} (\omega, x, \cdot)}}{d\nu^{\otimes(T_{\omega^{-1}})} (x', \theta)} \frac{d\nu^{\otimes(T_{\omega^{-1}})} (x', \theta)}{d\tilde{\nu}^{\otimes(T_{\omega^{-1}})} (x', \theta)} \right)^2 \times$$

$$\tilde{M}_{\theta}^{T_{\omega^{-1}, T_{\omega^{-1}}} (\omega, x, dx')} \geq \inf_{(\omega', z) \in \Omega \times E^{T_{\omega^{-1}}}} W^{T_{\omega^{-1}}} (\omega', z)^2 \left( \frac{\epsilon T_{\omega^{-1}}}{\Delta_3 \epsilon_+} \right)^2 \epsilon_- \int_A \left( \frac{d\nu^{\otimes(T_{\omega^{-1}})} (x', \theta)}{d\tilde{\nu}^{\otimes(T_{\omega^{-1}})} (x', \theta)} \right)^2 \tilde{\nu} (dx'$$,

by Lemma A.1 and Definition (4.1.1). Taking $A = E^{T_{\omega^{-1}}}$ and dividing (A8) by (A9), we have

$$\frac{J_{\theta}^{T_{\omega^{-1}, T_{\omega^{-1}}} (\omega, x)}}{J_{\theta}^{T_{\omega^{-1}, T_{\omega^{-1}}} (\omega', x')}} \leq \left( \frac{\Delta_3 \tilde{\epsilon}_+ T_{\omega^{-1}}}{\epsilon_- \epsilon T_{\omega^{-1}} - \tilde{\epsilon}_+} \right)^2 \epsilon_- = \alpha.$$  

Finally, as $L_{\theta}^{T_{\omega^{-1}, T_{\omega^{-1}}} (\omega, x, \cdot)}$ is a kernel that has $J_{\theta}^{T_{\omega^{-1}, T_{\omega^{-1}}} (\omega, x)}$ as a normalising constant, it is clear via (A8) and (A9) that

$$\delta_- = \inf_{(\omega', z) \in \Omega \times E^{T_{\omega^{-1}}}} W^{T_{\omega^{-1}}} (\omega', z)^2 \left( \frac{\epsilon T_{\omega^{-1}}}{\Delta_3 \epsilon_+} \right)^2 \epsilon_- \left( \int_E \left( \frac{d\nu^{\otimes(T_{\omega^{-1}})} (x', \theta)}{d\tilde{\nu}^{\otimes(T_{\omega^{-1}})} (x', \theta)} \right)^2 \tilde{\nu} (dx') \right)^{-1}$$
and
\[
\delta_+ = \sup_{(\omega', z) \in \Omega \times E} W^{T_{\omega'}^{-1}} \left( \omega'_+, z \right)^2 \left( \frac{\Delta_3 T_{\omega'}^{-1}}{\epsilon_+} \right)^2 \tilde{\epsilon}_+ \left( \int_E \frac{d\nu \otimes (T_{\omega'}^{-1})}{d\tilde{\nu}} (x') \right)^2 \left( \tilde{\nu} (dx') \right)^{-1}
\]
when
\[
\sigma (dx) \propto \left( \frac{d\nu \otimes (T_{\omega'}^{-1})}{d\tilde{\nu}} (x) \right)^2 \tilde{\nu} (dx).
\]

\[\square\]

**Proposition A.4** Assume (B1), (B2) and (B3). For each \(\tilde{\mathcal{M}}_{T_{\omega'}^{-1}, T_{\omega}^{-1}, T_{zn\omega}^{-1}}\), any member of a \(\tilde{\mathcal{M}}_{T_{\omega'}^{-1}, T_{\omega}^{-1}, T_{zn\omega}^{-1}}\), there exists a non-negative, finite constant \(\Upsilon (\tilde{\mathcal{M}}_{\theta})\), which is independent of the initial distribution \(\mu\), such that
\[
\frac{1}{n} \log \tilde{V}_{\omega, \theta, n} \rightarrow \Upsilon (\tilde{\mathcal{M}}_{\theta})
\]
as \(n \rightarrow \infty\), \(\mathbb{P}\)-almost surely.

**Proof of Proposition A.4** Assume (B1) and (B2). By Proposition A.1, for any \(\mu \in \mathcal{P}(E)\),
\[
\frac{2}{n} \log \mu Q_{\theta, n}^\omega (1) \rightarrow 2\Lambda
\]
as \(n \rightarrow \infty\), \(\mathbb{P}\)-almost surely.

Next, assume (B3) and consider the bounds presented in Lemma A.2. Following the exact same steps as in the proof of [91, Proposition 1], one can show that there exists a constant \(\Xi \in (-\infty, \infty)\) such that
\[
\frac{1}{n} \log \mu^\otimes (T_{\omega}^{-1}) R_{\theta, n}^{\omega, T_{\omega}^{-1}, T_{zn\omega}^{-1}} (1) \rightarrow \Xi
\]
as \(n \rightarrow \infty\), \(\mathbb{P}\)-almost surely.

By the definition (4.1.3), we have
\[
\tilde{V}_{\theta, n}^\omega = \frac{\mu^\otimes (T_{\omega}^{-1}) R_{\theta, n}^{\omega, T_{\omega}^{-1}, T_{zn\omega}^{-1}} (1)}{(\mu Q_{\theta, n}^\omega (1))^2},
\]
and so $\Upsilon \left( \widetilde{M}_\theta \right) = \Xi - 2\Lambda$. □

There is one final lemma which is needed to prove Theorem 4.3.1. The following is a technical result establishing that an additive functional of the form (4.1.4), with optimal $h$ as defined in Proposition A.2, is an eigenfunction for $Q_\theta$.

**Lemma A.3** Assume (B2). Then for any $\omega \in \Omega$,

$$Q_\theta^{\omega,m_1,m_2} \left( h_{m_2}^{m_2} (z_\omega, \cdot) \right) (x) = \lambda_\omega h_{m_1}^{m_1} (\omega, x),$$

where $\lambda_\omega$ is as in Proposition A.2.

**Proof of Lemma A.3**

$$Q_\theta^{\omega,m_1,m_2} \left( h_{m_2}^{m_2} (z_\omega, \cdot) \right) (x) = \int_{E_{m_2}} Q_\theta^{m_1,m_2} \left( \omega, x, du \right) h_{m_2}^{m_2} (z_\omega, u)$$

$$= \frac{1}{m_2} \sum_{j=1}^{m_2} \int_{E_{m_2}} Q_\theta^{m_1,m_2} \left( \omega, x, du \right) h \left( z_\omega, u^j \right)$$

$$= \frac{1}{m_2} \sum_{j=1}^{m_2} \frac{W_{m_1}^{m_1} (\omega, x)}{ \int_{E} W \left( \omega, x^i \right) M_\theta \left( \omega, x^i, du^j \right) h \left( z_\omega, u^j \right) } \sum_{i=1}^{m_1} W \left( \omega, x^i \right) M_\theta \left( \omega, x^i, du^j \right) h \left( z_\omega, u^j \right)$$

$$= \frac{1}{m_1 m_2} \sum_{j=1}^{m_2} \sum_{i=1}^{m_1} \lambda_\omega h \left( \omega, x^i \right)$$

$$= \lambda_\omega \frac{1}{m_1} \sum_{i=1}^{m_1} h \left( \omega, x^i \right)$$

where we have applied Proposition A.2. □

Finally, we use the optimal $h$ of Proposition A.2 and prove that the specific, unique $\widetilde{M}_\theta$ defined in Theorem 4.3.1 achieves $\Upsilon \left( \widetilde{M}_\theta \right) = 0$. The main result is presented as Theorem 4.3.1 in Section 4.3, and its proof follows.
Proof of Theorem 4.3.1 It is obvious that 3) $\Rightarrow$ 1). Thus, we show 1) $\Rightarrow$ 2) and 2) $\Rightarrow$ 3). This proof is nearly identical to the proof of [91, Theorem 1].

1) $\Rightarrow$ 2): Assume (B2) and (B3), and consider the bounds presented in Lemma A.2. Following the same steps as in the proofs of [91, Proposition 2] and [91, Proposition 3], one can show the following:

- there exists a random variable $\xi_{\omega, m_1} : \Omega \to \mathbb{R}^+$ and a function $l_{m_1} : \Omega \times E_{m_1} \to \mathbb{R}^+$ which is measurable with respect to $\mathcal{F} \otimes E_{m_1}$ such that

$$\begin{align*}
\sup_{\omega, \omega'} \xi_{\omega, T_{\omega} - 1} &< \infty, \\
\sup_{\omega, \omega', x, x'} l_{T_{\omega} - 1}^{-1}(\omega, x) &< \infty, \\
R_{\theta}^{\omega, T_{\omega} - 1, T_{\omega'} - 1} \left( l_{T_{\omega} - 1}^{-1}(\omega, \cdot) \right) (x) &= \xi_{\omega, T_{\omega} - 1} l_{T_{\omega'} - 1}(\omega, x); \\
\end{align*}$$

(A11)

- for any $x \in E_{T_{\omega} - 1}$, we have

$$\Xi = \mathbb{E}[\log \xi_{T_{\omega} - 1}] = \int_{\Omega} R_{\theta}^{\omega, T_{\omega} - 1, T_{\omega'} - 1} \left( l_{T_{\omega} - 1}^{-1}(\omega, \cdot) \right) (x) \frac{l_{T_{\omega} - 1}^{-1}(\omega, x)}{P(d\omega)},$$

(A12)

where $\Xi \in (-\infty, \infty)$ is as in (A10). By the concluding lines of the proof of Proposition A.4 above, we know that $\Upsilon(\tilde{M}_\theta) = \Xi - 2\Lambda$, and by Proposition A.3 and (A12), we have $\Xi - 2\Lambda = \mathbb{E}[\log \xi_{T_{\omega} - 1}]$. Therefore, the first part of Theorem 4.3.1 implies that

$$\Upsilon(\tilde{M}_\theta) = \Xi - 2\Lambda = \mathbb{E}[\log \frac{\xi_{T_{\omega} - 1}}{\lambda^2}] = 0.$$

(A13)

The results (A11), (A12), and (A13) are used extensively throughout the following proof.

Next, define $\gamma_{m_1}(\omega, x) = \frac{h_{m_1}(\omega, x)^2}{l_{m_1}(\omega, x)}$ and $\rho_{m_1} = \text{ess sup}_x \gamma_{m_1}(\omega, x)$ for $x \in E_{m_1}$. From the third part of Proposition A.2 and from (A11) above, we note that

$$\sup_{\omega, \omega', x, x'} \frac{\gamma_{T_{\omega} - 1}(\omega, x)}{\gamma_{T_{\omega'} - 1}(\omega', x')} < \infty \quad \Rightarrow \quad 0 < \rho_{T_{\omega} - 1} < \infty \quad \forall \omega \in \Omega.$$

This bound on $\rho_{T_{\omega} - 1}$ allows us to write the following: by (A11), we have, for $P$–almost all
\[ \omega \text{ and any } x \in E_{T_{\omega}^{-1}}, \]
\[ \xi_{\omega, T_{\omega}^{-1}} = \frac{R_{T_{\omega}^{-1}, T_{\omega}^{-1}} \left( I_{T_{\omega}^{-1}} \left( z_{\omega}, \cdot \right) \right) (x)}{l_{T_{\omega}^{-1}}(\omega, x)} \]
\[ = \int W_{T_{\omega}^{-1}}(\omega, x)^2 \phi_{T_{\omega}^{-1}, T_{\omega}^{-1}}(x, x')^2 \tilde{M}_{T_{\omega}^{-1}, T_{\omega}^{-1}}(\omega, x, dx') I_{T_{\omega}^{-1}}(z_{\omega}, x') \]
\[ \geq \frac{\int W_{T_{\omega}^{-1}}(\omega, x)^2 \phi_{T_{\omega}^{-1}, T_{\omega}^{-1}}(x, x')^2 \tilde{M}_{T_{\omega}^{-1}, T_{\omega}^{-1}}(\omega, x, dx') h_{T_{\omega}^{-1}}(z_{\omega}, x')^2}{l_{T_{\omega}^{-1}}(\omega, x) \rho_{T_{\omega}^{-1}}^{T_{\omega}^{-1}}} \]
\[ \geq \left[ \int Q_{T_{\omega}^{-1}, T_{\omega}^{-1}}(\omega, x, dx') h_{T_{\omega}^{-1}}(z_{\omega}, x') \right]^2 \]
\[ \geq \frac{1}{\rho_{T_{\omega}^{-1}}^{T_{\omega}^{-1}}} \times \]
\[ \text{ess sup} \frac{\int W_{T_{\omega}^{-1}}(\omega, x)^2 \phi_{T_{\omega}^{-1}, T_{\omega}^{-1}}(x, x')^2 \tilde{M}_{T_{\omega}^{-1}, T_{\omega}^{-1}}(\omega, x, dx') h_{T_{\omega}^{-1}}(z_{\omega}, x')^2}{l_{T_{\omega}^{-1}}(\omega, x) \rho_{T_{\omega}^{-1}}^{T_{\omega}^{-1}}} \]
\[ \geq \lambda^2 \rho_{T_{\omega}^{-1}}^{T_{\omega}^{-1}}. \]
These two inequalities, combined with assumption (B1) and equation (A13), allow for

\[ \mathbb{E}[\log \xi, T_{-1}] \geq \int_{\Omega} \log \left[ \frac{1}{\rho_{T_\omega}^{\omega-1}} \text{ess sup}_x \left( \frac{\mathcal{R}_{\theta}^{T_\omega-1, T_{z\omega}-1}(\omega, x, dx') \mathcal{h}^{T_{z\omega}-1}(z\omega, x')}{l^{T_\omega-1}(\omega, x)} \right)^2 \right] \mathbb{P}(d\omega) \]

\[ = \int_{\Omega} \log \left[ \frac{1}{\rho_{T_\omega}^{\omega-1}} \text{ess sup}_x \left( \mathcal{R}_{\theta}^{T_\omega-1, T_{z\omega}-1}(\omega, x, dx') \mathcal{h}^{T_{z\omega}-1}(z\omega, x') \right)^2 \right] \mathbb{P}(d\omega) \]

\[ \geq \int_{\Omega} \log \left[ \frac{1}{\rho_{T_\omega}^{\omega-1}} \text{ess sup}_x \left( \mathcal{R}_{\theta}^{T_\omega-1, T_{z\omega}}(\omega, x) \mathcal{h}^{T_{z\omega}}(x)^2 \right) \right] \mathbb{P}(d\omega) \]

which implies

\[ \mathbb{E}[\log \xi, T_{-1}] = \int_{\Omega} \log \left[ \frac{1}{\rho_{T_\omega}^{\omega-1}} \text{ess sup}_x \left( \mathcal{R}_{\theta}^{T_\omega-1, T_{z\omega}}(\omega, x) \mathcal{h}^{T_{z\omega}}(x)^2 \right) \right] \mathbb{P}(d\omega) \]

\[ = \int_{\Omega} \log \left[ \frac{1}{\rho_{T_\omega}^{\omega-1}} \text{ess sup}_x \left( \mathcal{R}_{\theta}^{T_\omega-1, T_{z\omega}}(\omega, x) \mathcal{h}^{T_{z\omega}}(x)^2 \right) \right] \mathbb{P}(d\omega). \]

(A16)

For any \( e > 0 \) and any \( \omega \in \Omega \), let us introduce a set \( A_{\omega, e} \subseteq \mathbb{E}^{(T_{-1})} \) such that

\[ A_{\omega, e} = \left\{ x : l^{T_{-1}}(\omega, x) < \frac{1 + e}{\rho_{T_\omega}^{\omega-1}} \mathcal{h}^{T_{-1}}(\omega, x)^2 \right\} . \]

For \( \mathbb{P} \)-almost all \( \omega \) and all \( x \in E^{T_{-1}} \), it is then clear that

\[ l^{T_{-1}}(\omega, x) \geq \frac{1 + e}{\rho_{T_\omega}^{\omega-1}} \mathcal{h}^{T_{-1}}(\omega, x)^2 \mathbb{I}_{A_{\omega, e}}(x) . \]

By assumption (B3), the third part of Proposition A.2, and equations (4.1.4) and (A11), we know \( l^{T_{-1}}(\omega, x) \) and \( \mathcal{h}^{T_{-1}}(\omega, x) \) are each greater than zero for any \( (\omega, x) \). Thus, for
Chapter 4. Twisting the Alive Particle Filter 197

\[ x \in E^{T_{\omega}^{-1}} \]

\[ \xi_{\omega,T_{\omega}^{-1}} = \frac{R_{\theta}^{T_{\omega}^{-1}, T_{\omega}^{-1}} \left( T_{\omega}^{-1} (z\omega, \cdot) \right) (x)}{l^{T_{\omega}^{-1}} (\omega, x)} \geq \frac{R_{\theta}^{T_{\omega}^{-1}, T_{\omega}^{-1}} \left( h_{T_{\omega}^{-1}} (z\omega, \cdot) ^2 \right) (x)}{l^{T_{\omega}^{-1}} (\omega, x) \rho_{\omega,T_{\omega}^{-1}}^{T_{\omega}^{-1}}} + e \int_{\mathcal{A}_{\omega,e}} R_{\theta}^{T_{\omega}^{-1}, T_{\omega}^{-1}} (\omega, x, dx)^{T_{\omega}^{-1}} \left( z\omega, x' \right) ^2 \h_{T_{\omega}^{-1}} (z\omega, x') ^2 \left( \sigma_{T_{\omega}^{-1}} \rho_{\omega,T_{\omega}^{-1}}^{T_{\omega}^{-1}} \right) ^{-1} \right] ^{1} \] 

where the measure \( \sigma (\cdot) \) from Lemma A.2 aids in taking the place of \( R_{\theta}^{T_{\omega}^{-1}, T_{\omega}^{-1}} (\omega, x, dx) \) and \( \kappa \) is some strictly positive constant that acts as an upper bound for \( h_{T_{\omega}^{-1}} (z\omega, x') ^2 \), \( l^{T_{\omega}^{-1}} (\omega, x) \), and the weight included within \( R_{\theta}^{T_{\omega}^{-1}, T_{\omega}^{-1}} (\omega, x, dx) \). As this expression holds for all \( x \in E^{T_{\omega}^{-1}} \) and as \( z \) preserves \( \mathbb{P} \) under assumption (B1), \( \mathbb{E}[\log \xi_{\cdot,T_{\omega}^{-1}}] \geq \int_{\Omega} \log \left[ \text{ess sup} \frac{R_{\theta}^{T_{\omega}^{-1}, T_{\omega}^{-1}} \left( h_{T_{\omega}^{-1}} (z\omega, \cdot) ^2 \right) (x)}{l^{T_{\omega}^{-1}} (\omega, x) \rho_{\omega,T_{\omega}^{-1}}^{T_{\omega}^{-1}}} + e \int_{\mathcal{A}_{\omega,e}} R_{\theta}^{T_{\omega}^{-1}, T_{\omega}^{-1}} (\omega, x, dx)^{T_{\omega}^{-1}} \left( z\omega, x' \right) ^2 \h_{T_{\omega}^{-1}} (z\omega, x') ^2 \left( \sigma_{T_{\omega}^{-1}} \rho_{\omega,T_{\omega}^{-1}}^{T_{\omega}^{-1}} \right) ^{-1} \right] \mathbb{P}(\omega). \] 

(A17) The R.H.S. of (A17) and (A16) together imply that, for \( \mathbb{P} \)-almost all \( \omega \), \( \sigma \left( A_{z\omega,e}^{c} \right) = 0 \) — i.e., \( \mathbb{P} \left( \{ \omega: \sigma \left( A_{z\omega,e}^{c} \right) = 0 \} \right) = 1 \). The definition of \( \sigma (\cdot) \) from Lemma A.2 then indicates

\[ 1 = \mathbb{P} \left( \{ \omega: \sigma \left( A_{z\omega,e}^{c} \right) = 0 \} \right) = \mathbb{P} \left( \{ \omega: \nu^{T_{\omega}^{-1}} \left( A_{z\omega,e}^{c} \right) = 0 \} \right) \]

\[ = \mathbb{P} \left( \{ \omega: \nu^{T_{\omega}^{-1}} \left( A_{z\omega,e} \right) = 1 \} \right) = \mathbb{P} \left( \{ \omega: \nu^{T_{\omega}^{-1}} \left( A_{z\omega,e} \right) = 1 \} \right) . \]

The definitions of \( l^{T_{\omega}^{-1}} (\omega, x) \), \( A_{\omega,e} \), and \( \rho_{\omega,T_{\omega}^{-1}}^{T_{\omega}^{-1}} \) imply

\[ \bigcap_{t \geq 1} A_{\omega,1}^{t} = \left\{ x: l^{T_{\omega}^{-1}} (\omega, x) = \frac{1}{\rho_{\omega,T_{\omega}^{-1}}^{T_{\omega}^{-1}}} h_{T_{\omega}^{-1}} (\omega, x)^2 \right\} \]  

(A18)

and

\[ A_{\omega,1}^{t} \subseteq A_{\omega,1}^{t+1}, \quad \forall t \geq 1. \]  

(A19)

We denote (A18) by \( A_{\omega}^{*} \). Given a probability measure \( \nu^{T_{\omega}^{-1}} \), continuity of probability states that if \( A_{\omega,1}^{t} \subseteq E^{\otimes \otimes (T_{\omega}^{-1})} \) is a decreasing family of sets (see (A19)) and if \( \bigcap_{t \geq 1} A_{\omega,1}^{t} \) is
equal to some set $A^*_\omega$ (see (A18)), then \( \lim_{t \to \infty} \nu^{\otimes (T_\omega - 1)} \left( A_{\omega, \frac{1}{t}} \right) = \nu^{\otimes (T_\omega - 1)} (A^*_\omega) \). Thus, if for some $\omega \in \Omega$, \( \nu^{\otimes (T_\omega - 1)} \left( A_{\omega, \frac{1}{t}} \right) = 1 \) for all $t \geq 1$, then the continuity of probability implies that \( \nu^{\otimes (T_\omega - 1)} (A^*_\omega) = 1 \). Additionally, by the definitions of $A^*_\omega$ and $A_{\omega, \frac{1}{t}}$, if \( \nu^{\otimes (T_\omega - 1)} (A^*_\omega) = 1 \), then it must be that \( \nu^{\otimes (T_\omega - 1)} \left( A_{\omega, \frac{1}{t}} \right) = 1 \) for all $t \geq 1$. It is then clear that

\[
\left\{ \omega : \nu^{\otimes (T_\omega - 1)} (A^*_\omega) = 1 \right\} \subseteq \bigcap_{t \geq 1} \left\{ \omega : \nu^{\otimes (T_\omega - 1)} \left( A_{\omega, \frac{1}{t}} \right) = 1 \right\}.
\]

(A20)

Furthermore, (A19) indicates that

\[
\left\{ \omega : \nu^{\otimes (T_\omega - 1)} \left( A_{\omega, \frac{1}{t}} \right) = 1 \right\} \leq \left\{ \omega : \nu^{\otimes (T_\omega - 1)} (A^*_\omega) = 1 \right\}, \quad \forall t \geq 1.
\]

(A21)

Again, by the continuity of probability, equations (A20) and (A21), together with

\[
P \left( \left\{ \omega : \nu^{\otimes (T_\omega - 1)} (A^*_\omega) = 1 \right\} \right) = 1
\]

from above, imply that

\[
P \left( \left\{ \omega : \nu^{\otimes (T_\omega - 1)} (A^*_\omega) = 1 \right\} \right) = 1.
\]

We now know that $A^*_\omega$ is non-empty, and so for any $x \in A^*_\omega$, we can write

\[
\frac{\xi_{\omega, T_\omega - 1}}{\lambda^2_{\omega}} = \frac{R^\omega_{\theta, T_\omega - 1, T_{\omega - 1}} \left( I_{T_{\omega - 1}} \left( z_{\omega, \cdot} \right) \right) (x)}{I_{T_{\omega - 1}} (\omega, x)} \cdot \frac{h^{T_{\omega - 1}} (\omega, x)^2}{Q^\omega_{\theta, T_\omega - 1, T_{\omega - 1}} \left( h^{T_{\omega - 1}} \left( z_{\omega, \cdot} \right) \right) (x)^2}.
\]

The definition of $A^*_\omega$ then allows for

\[
= \frac{R^\omega_{\theta, T_\omega - 1, T_{\omega - 1}} \left( h^{T_{\omega - 1}} \left( z_{\omega, \cdot} \right) \right) (x)}{\rho_{T_{\omega - 1}}^{T_{\omega - 1}}} \cdot \rho_{T_{\omega - 1}}^{T_{\omega - 1}} \left[ Q^\omega_{\theta, T_\omega - 1, T_{\omega - 1}} \left( h^{T_{\omega - 1}} \left( z_{\omega, \cdot} \right) \right) (x)^2 \right]^{-1}
\]

\[
= \frac{\rho_{T_{\omega - 1}}^{T_{\omega - 1}}}{\rho_{T_{\omega - 1}}^{T_{\omega - 1}}} \times \frac{W_{T_{\omega - 1}} (\omega, x)^2 \int_{E_{T_{\omega - 1}}} \phi^\omega_{\theta, T_\omega - 1, T_{\omega - 1}} (x, x')^2 M_{\theta}^{T_{\omega - 1}, T_{\omega - 1}} (\omega, x, dx) h^{T_{\omega - 1}} (z_{\omega, x'})^2}{W_{T_{\omega - 1}} (\omega, x)^2 \left[ \int_{E_{T_{\omega - 1}}} \phi^\omega_{\theta, T_\omega - 1, T_{\omega - 1}} (x, x') M_{\theta}^{T_{\omega - 1}, T_{\omega - 1}} (\omega, x, dx') h^{T_{\omega - 1}} (z_{\omega, x'})^2 \right]^{-1}}
\]

\[
\geq \frac{\rho_{T_{\omega - 1}}^{T_{\omega - 1}}}{\rho_{T_{\omega - 1}}^{T_{\omega - 1}}}
\]
by Jensen’s inequality. Additionally, by (A13) and assumption (B1),

\[
\mathbb{E} \left[ \log \frac{\xi_{\omega,T_\omega \omega}^{-1} \cdot \rho_{\omega,T_\omega \omega}^{-1}}{\lambda_\omega^2} \right] = \mathbb{E} \left[ \log \frac{\xi_{\omega,T_\omega \omega}^{-1}}{\lambda_\omega^2} \right] + \mathbb{E} \left[ \log \rho_{\omega,T_\omega \omega}^{-1} \right] - \mathbb{E} \left[ \log \rho_{\omega,T_\omega \omega}^{-1} \right] \\
= 0 \rightarrow \frac{\xi_{\omega,T_\omega \omega}^{-1}}{\lambda_\omega^2} \cdot \rho_{\omega,T_\omega \omega}^{-1} = 1,
\]

for \( \mathbb{P} \)-almost all \( \omega \), and this result implies that equality must hold in the last use of Jensen’s inequality:

\[
\int_{E_{\omega,T_\omega \omega}^{-1}} \phi_\omega^T \omega_{T_\omega \omega}^{-1} \left( x, x' \right)^2 \tilde{M}_{\theta}^{T_\omega \omega \omega}^{-1} \left( \omega, x, dx' \right) h_{T_\omega \omega}^{-1} \left( z_{\omega}, x' \right)^2 = \left[ \int_{E_{\omega,T_\omega \omega}^{-1}} \phi_\omega^T \omega_{T_\omega \omega}^{-1} \left( x, x' \right) \tilde{M}_{\theta}^{T_\omega \omega \omega}^{-1} \left( \omega, x, dx' \right) h_{T_\omega \omega}^{-1} \left( z_{\omega}, x' \right) \right]^2
\]

\[
= \left[ \int_{E_{\omega,T_\omega \omega}^{-1}} \phi_\omega^T \omega_{T_\omega \omega}^{-1} \left( x, x' \right) \tilde{M}_{\theta}^{T_\omega \omega \omega}^{-1} \left( \omega, x, dx' \right) h_{T_\omega \omega}^{-1} \left( z_{\omega}, x' \right) \right]^2
\]

Straight calculation shows that the following satisfies (A22):

\[
\tilde{M}_{\theta}^{T_\omega \omega \omega}^{-1} \left( \omega, x, dx' \right) = \frac{M_{\theta}^{T_\omega \omega \omega}^{-1} \left( \omega, x, dx' \right) h_{T_\omega \omega}^{-1} \left( z_{\omega}, x' \right)}{\int_{E_{\omega,T_\omega \omega}^{-1}} M_{\theta}^{T_\omega \omega \omega}^{-1} \left( \omega, x, du \right) h_{T_\omega \omega}^{-1} \left( z_{\omega}, u \right)}.
\]
is clearly a member of $\mathcal{M}^{T_\omega-1, T_\omega-1}$. Thus, we can use (4.1.2) and Lemma A.3 to write

$$\prod_{k=0}^{n-1} W^{T_k\omega-1}(z^k\omega, x_k) \phi^{z^k\omega, T_k\omega-1, T_{k+1}\omega-1}(x_k, x_{k+1})$$

$$= \prod_{k=0}^{n-1} W^{T_k\omega-1}(z^k\omega, x_k) \times$$

$$\int_{E^{T_{k+1}\omega-1}} M^{T_{k+1}\omega-1, T_{k+1}\omega-1}_{\theta}(z^k\omega, x_k, du_{k+1}) h^{T_{k+1}\omega-1}(z^{k+1}\omega, u_{k+1})$$

$$= \prod_{k=0}^{n-1} Q^{z^k\omega, T_k\omega-1, T_{k+1}\omega-1}_{\theta}(h^{T_{k+1}\omega-1}(z^{k+1}\omega, \cdot))(x_k)$$

$$= \frac{h^{T_{n}\omega-1}(\omega, x_0) \prod_{k=0}^{n-1} Q^{z^k\omega, T_k\omega-1, T_{k+1}\omega-1}_{\theta}(h^{T_{k+1}\omega-1}(z^{k+1}\omega, \cdot))(x_k)}{h^{T_{n}\omega-1}(z^n\omega, x_n)}$$

$$= \frac{h^{T_{n}\omega-1}(\omega, x_0) \prod_{k=0}^{n-1} \lambda^{z^k\omega}(x_k)}{h^{T_{n}\omega-1}(z^n\omega, x_n)}$$

Now consider the following, where the expectation is taken with respect to $\tilde{M}_\theta$:

$$\mu^{(T_\omega-1)}_{\tilde{M}_\theta, n} \begin{pmatrix} \omega, T_\omega-1, T_n\omega-1 \end{pmatrix}$$

$$= \mathbb{E} \left[ \prod_{k=0}^{n-1} W^{T_k\omega-1}(z^k\omega, x_k)^2 \phi^{z^k\omega, T_k\omega-1, T_{k+1}\omega-1}(x_k, x_{k+1})^2 \right]$$

$$= \left( \prod_{k=0}^{n-1} \lambda^{z^k\omega} \right)^2 \mathbb{E} \left[ \frac{h^{T_n\omega-1}(\omega, x_0)^2}{h^{T_n\omega-1}(z^n\omega, x_n)^2} \right].$$

Recall that (A23) is defined such that $\nu^{(T_\omega-1)}(A^\omega_n) = 0$ and $x \in A_\omega$. This fact, combined with the assumption (B1) that $z$ preserves $\mathbb{P}$ yields

$$\leq \left( \prod_{k=0}^{n-1} \lambda^{z^k\omega} \right)^2 \left( \sup_{(\omega, \omega', x, x')} \frac{h(\omega, x)}{h(\omega', x')} \right)^2.$$

Now, due to the fourth part of Proposition A.2, we also have

$$W(\omega, x) M^{\omega}_{\theta}(z(\omega, \cdot))(x) = \lambda_{\omega} h(\omega, x).$$
This result implies the following, where the expectation is taken with respect to $M_0$:

$$(\mu Q_{\theta,n}^\omega(1))^2 = \mathbb{E}\left[\prod_{k=0}^{n-1} W(z^k, x_k) \right]^2$$

$$= \mathbb{E}\left[\prod_{k=0}^{n-1} \lambda_{z^k} h(z^k, x_k) \right]^2 M^{z^k}(h(z^{k+1}, \cdot))(x_k)$$

$$= \left(\prod_{k=0}^{n-1} \lambda_{z^k}\right)^2 \mathbb{E}\left[\frac{h(\omega, x_0)}{h(z^n, x_n)} \prod_{k=1}^{n} \frac{h(z^k, x_k)}{M^{z^k}(h(z^k, \cdot))(x_{k-1})}\right]^2$$

$$\geq \left(\prod_{k=0}^{n-1} \lambda_{z^k}\right)^2 \left(\inf_{(\omega,x,x')}(\frac{h(\omega, x)}{h(\omega', x')})\right)^2.$$ 

Finally, we have, by the third part of Proposition A.2,

$$\sup_{n \geq 1} \frac{\mu^{\otimes(T_\omega-1)} R_{\theta,n}^{\omega,T_\omega-1,T_n\omega-1}(1)}{(\mu Q_{\theta,n}^\omega(1))^2} \leq \left(\sup_{(\omega,x,x')}(\frac{h(\omega, x)}{h(\omega', x')})\right)^4 < \infty. \tag*{\square}$$
B Table of kernel and operator notation

Table B1: Kernel and operator notation used throughout Chapter 4

<table>
<thead>
<tr>
<th>Kernels</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( Q_\theta (\omega, x, dx) = W(\omega, x) M_\theta (\omega, x, dx) )</td>
<td>1. ( M_\sigma^\omega (W)(x) = \int_E M_\theta (\omega, x, dx) W(z\omega, x') )</td>
</tr>
<tr>
<td>2a. ( Q_{\theta,n} (\omega, x, x') = \int Q_{\theta,n-1} (\omega, x, u) Q_\theta (z^{n-1}\omega, du, x') ), ( n \geq 1 )</td>
<td>2a. ( Q_\theta^\omega (\varphi)(x) = \int_E Q_\theta (\omega, x, dx') \varphi(x') ), ( \varphi \in \mathcal{B}_b(E) )</td>
</tr>
<tr>
<td>2b. ( Q_{\theta,n} (\omega, x, x') = Q_{\theta,n-1}^\omega Q_\theta (z^{n-1}\omega, x') ), ( n \geq 1 ), with ( Q_{\theta,0} (\omega, x, x) = I )</td>
<td>2b. ( Q_{\theta,n}^\omega (\varphi)(x) = \int_{E^n} Q_\theta (\omega, x, dx') \varphi(x') ), ( \varphi \in \mathcal{B}_b(E) )</td>
</tr>
<tr>
<td>2c. ( Q_{\theta,n+1} (\omega, x, dx) = Q_{\theta,n}^\omega Q_{\theta,n} (z^n\omega, dx) ) via induction as in [91]</td>
<td>3a. ( \sigma Q_\theta^\omega (\cdot) = \int_E Q_\theta (\omega, x, \cdot) \sigma(dx) ), ( \sigma \in \mathcal{M}(E) ) or ( \sigma \in \mathcal{P}(E) )</td>
</tr>
<tr>
<td>3. ( Q_{\theta}^{m_1,m_2} (\omega, x, dx) = W^{m_1}(\omega, x) M_\theta^{m_1,m_2} (\omega, x, dx) )</td>
<td>3b. ( \sigma Q_\theta^\omega (\cdot) = \int_E Q_\theta (\omega, x, \cdot) \sigma(dx) ), ( \sigma \in \mathcal{M}(E) ) or ( \sigma \in \mathcal{P}(E) )</td>
</tr>
<tr>
<td></td>
<td>3c. ( \sigma Q_\theta^\omega (W^{z^n\omega}) = \int_{E^{n+1}} Q_\theta (\omega, x, x') W(z^n\omega, x') \sigma(dx') = \sigma Q_{\theta,n+1}^\omega (1) )</td>
</tr>
<tr>
<td></td>
<td>4a. ( Q_{\theta}^{m_1,m_2} (f^{m_2}(z\omega, \cdot))(x) = \int_{E^{m_2}} Q_{\theta}^{m_1,m_2} (\omega, x, dx') f^{m_2}(z\omega, x') ), ( f \in \mathcal{B}_b(E) )</td>
</tr>
<tr>
<td></td>
<td>4b. ( \sigma Q_\theta^{m_1,m_2} (\cdot) = \int_{E^{m_1}} Q_{\theta}^{m_1,m_2} (\omega, x, \cdot) \sigma(dx) ), ( \sigma \in \mathcal{M}(E^{m_1}) ) or ( \sigma \in \mathcal{P}(E^{m_1}) )</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Probability measures</td>
<td></td>
</tr>
<tr>
<td>1. ( \Phi_\theta^\omega (\sigma)(\cdot) = \frac{\sigma Q_\theta^\omega (\cdot)}{\sigma Q_\theta^\omega (1)} (\cdot) )</td>
<td></td>
</tr>
<tr>
<td>2. ( \Phi_{\theta,n}^\omega (\sigma)(\cdot) = \left( \Phi_{\theta,n-1}^{n-1} \circ \Phi_{\theta,n}^\omega \right)(\sigma)(\cdot) ), ( n \geq 1 ), with ( \Phi_{\theta,0}^\omega (\sigma)(\cdot) = I )</td>
<td></td>
</tr>
<tr>
<td>3a. ( \Phi_{\theta,n}^\omega (\sigma) = \frac{\sigma Q_{\theta,n}^\omega (\cdot)}{\sigma Q_{\theta,n}^\omega (1)} ) via induction as in [91]</td>
<td></td>
</tr>
<tr>
<td>3b. ( \Phi_{\theta,n}^\omega (\sigma) = \Phi_{\theta,n-1}^{n-1} \circ \Phi_{\theta}^\omega (\sigma) ) via induction as in [91]</td>
<td></td>
</tr>
<tr>
<td>3c. ( \Phi_{\theta,n+p}^\omega (\sigma) = \left( \Phi_{\theta,p}^{n+1} \circ \Phi_{\theta,n}^\omega \right)(\sigma) ) via induction as in [91]</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 5

Inference for Partially Observed Stopped Markov Processes

At this point, it is clear how the PMCMC methodology can be applied to perform Bayesian inference on HMMs. In this chapter, we expand the applicability of PMCMC to a new class of problems (namely, the stopped Markov processes of Section 2.3). We introduce multi-level SMC (see Algorithm 17) into PMMH (see Algorithm 22) to allow one to perform Bayesian parameter inference on stopped Markov processes that are only partially observed. That is, we assume no direct access to the past states of the process \( \{X_n\}_{1 \leq n \leq \tau} \); we may only observe the process through the final state \( y = x_\tau \). The wider literature largely focuses on simulation methodology for stopped Markov processes with known parameters, and the only work on Bayesian parameter inference that we are aware of has been concerned with the scenario where \( \{X_n\}_{1 \leq n \leq \tau} \) is fully observed \([8]\).

Recall that Algorithm 17 uses an arbitrary sequence of \( \mathcal{F} \)-sets (or \( \mathcal{F} \)-sets)

\[ B_0, B_1, \ldots, B_p, \]

which are defined by the practitioner on a case-by-case basis. When all model parameters are known, trial runs can be used to determine a sensible sequence of \( \mathcal{F} \)-sets in a multi-level SMC scheme. However, when one is trying to infer the model parameters, trial runs are less of a feasible option. Thus, we introduce two types of PMMH algorithms that
use multi-level SMC (henceforth called multi-level PMMH algorithms, or ML-PMMH). One algorithm (see Section 5.1) uses a fixed sequence of $\mathcal{F}$—sets, which are again defined by the practitioner. The second algorithm (see Section 5.2) uses an adaptive sequence of $\mathcal{F}$—sets. In Section 5.3, we compare the resulting algorithms via numerical simulation to the standard Algorithm 22. In these examples, we infer model parameters for a partially observed coalescent with mutation and a partially observed coalescent with mutation and migration. Our methods converge significantly faster than Algorithm 22, with the adaptive scheme providing the most accurate output. Section 5.4 concludes the chapter with a summary and discussion.

5.1 Multi-level PMMH: fixed $\mathcal{F}$—sets

We begin by defining the multi-level PMMH algorithm that uses some fixed sequence of $\mathcal{F}$—sets (fixed ML-PMMH). We thereby establish the groundwork to define the adaptive algorithm in the next section.

Recall the definition of the backward-in-time stopped Markov process $\{X_j\}_{1 \leq j \leq \tau}$, and recall that it can be redefined as $\{X_n\}_{1 \leq n \leq p}$ using the fixed $\mathcal{F}$—sets $B_0, B_1, \ldots, B_p = A$ (see Section 2.11.1). For some known parameter $\theta$, the joint density $\pi_\theta(X_{1:p})$ as defined by (2.11.5) can be targeted using Algorithm 17 where one employs the backward SIS scheme of Section 2.11.1. Similar to (2.18.1), when one excludes the very final resampling step, the joint density of the variables simulated by Algorithm 17 will be

$$
\psi_\theta(X_{1:p}, a_{1:p-1}) = \left[ \prod_{i=1}^{N} \frac{1}{q_i (X_i)} \right] \left[ \prod_{n=2}^{p} r(a_{n-1} | w_{n-1}) \prod_{i=1}^{N} q_n (X^i_n | X_{1:n-1}) \right].
$$

(5.1.1)

When $\theta$ is unknown with prior density $\pi(\theta)$, we will want to sample from

$$
\pi(\theta, X_{1:p}) = \frac{\tilde{\gamma}_\theta(X_{1:p}) \pi(\theta)}{Z_{\theta,1:p}} = \frac{\gamma_\theta(x_{1:\tau}, \tau, y) \pi(\theta)}{Z_{\theta,1:p}}.
$$

(5.1.2)
We can use (5.1.1) to extend this density and obtain a target for a PMMH algorithm:

\[
\pi^N (k, \theta, X_{1:p}, a_{1:p-1}) = \frac{\pi(\theta, X_{1:p}^k)}{N_p} \cdot \frac{\psi(\mathcal{X}_{1:p}^k, a_{1:p-1})}{\bar{q}_1 (X_{1:p}^{k}) \left( \prod_{n=2}^{p} w_{n-1}^{n(k)} q_n (\mathcal{X}_n^{n(k)} | \mathcal{X}_{1:n-1}^{n(k)}) \right)}. \tag{5.1.3}
\]

Assuming again that \(q (\theta | \zeta)\) is the arbitrary density that proposes a new value \(\theta \in \Theta\) conditional on a current accepted value \(\zeta \in \Theta\), we can define a fixed ML-PMMH algorithm with proposal density

\[
q^N (k, \theta, X_{1:p}, a_{1:p-1}) = q (\theta | \zeta) \psi(\mathcal{X}_{1:p}^k, a_{1:p-1}) w_p^k
\]

as in Algorithm 35. [50] present convergence results for this algorithm and prove that using multi-level SMC within PMMH is valid.

We now have a multi-level PMMH algorithm, yet we still need to determine how best to structure the \(\mathcal{F}\)–sets \(B_0, B_1, \ldots, B_p\) so that they efficiently guide the sampled paths from \(B_0\) to the terminal set \(A\) with as few trajectories as possible leading to some absorbing set in between. Furthermore, if the \(\mathcal{F}\)–sets are too far apart (i.e., Algorithm 17 does not resample enough), then one can expect weight degeneracy. If the \(\mathcal{F}\)–sets are too close to one another (i.e., Algorithm 17 resamples too often), then one can expect extreme path degeneracy. In the next section, we develop an ML-PMMH algorithm that adapts itself to changing values of \(\theta\) so as to automatically pick the best \(\mathcal{F}\)–sets structure.

### 5.2 Multi-level PMMH: adaptive \(\mathcal{F}\)–sets

We begin by introducing the arbitrary auxiliary parameter \(v\) defined on the abstract state-space \((U, \mathcal{X})\) with \(V \sim \Lambda_\theta (\cdot)\). This variable will induce, for every \(\theta \in \Theta\), a random number of sets \(p(v) \in \mathcal{J} \subset \mathbb{Z}^+\) and a sequence of \(\mathcal{F}\)–sets \(\{B_n(v)\}_{1 \leq n \leq p(v)}\), with \(B_{p(v)}(v) = A\) (the terminal set). Similar to before, the sequence \(\{\mathcal{X}_n\}_{1 \leq n \leq p(v)}\) still forms a valid Markov chain taking values in

\[
\mathcal{E}_n = \bigcup_{\tau_n \in \mathcal{I}_n} \{\tau_n\} \times \mathcal{E}_{\tau_n}.
\]
5.2 Multi-level PMMH: adaptive $\mathcal{F}$-sets

**Algorithm 35** Fixed multi-level particle marginal Metropolis-Hastings

- **Step 0:** Draw $\theta$ from its prior. Fix the $\mathcal{F}$-sets $B_0, B_1, \ldots, B_p$ appropriately. All remaining random variables can be sampled from their full conditionals defined by the target (5.1.3):
  - Sample $X_{1:p}, a_{1:p-1} | \cdots \sim \psi_\theta(\cdot)$ via Algorithm 17, excluding the final resampling step. Use the backward methodology of Section 2.11.1.
  - Choose $k \propto W_k^p$.

Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta,1:p}$, via (2.11.1).

- **Step 1:** Sample $\theta^* \sim q(\cdot | \theta)$. All remaining random variables can be sampled from their full conditionals defined by the target (5.1.3):
  - Sample $X_{1:p}^*, a_{1:p-1}^* | \cdots \sim \psi_\theta(\cdot)$ via Algorithm 17, excluding the final resampling step. Use the backward methodology of Section 2.11.1.
  - Choose $k^* \propto W_p^{k^*}$.

Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta^*,1:p}$, via (2.11.1).

- **Step 2:** With acceptance probability

$$1 \land \frac{\pi(\theta^*) q(\theta^* | \theta^*) \hat{Z}_{\theta^*,1:p}}{\pi(\theta) q(\theta^* | \theta) \hat{Z}_{\theta,1:p}}$$

set $k=k^*, \theta=\theta^*, X_{1:p}=X_{1:p}^*,$ and $a_{1:p-1}=a_{1:p-1}^*$.

Return to the beginning of Step 1.
where each $\mathcal{I}_n$ is a collection of positive integer values related to possible stopping times. Thus, for every $\theta \in \Theta$, we have

$$Z_\theta = \sum_{\tau_p(v) \in \mathcal{I}_{p(v)}} \int_{E_{p(v)}} \gamma_\theta \left( x_{1: \tau_p(v)}, \tau_p(v), y \right) dx_{1: \tau_p(v)} = \sum_{\tau \in \mathcal{I}} \int_{E_{\tau}} \gamma_\theta \left( x_{1: \tau}, \tau, y \right) dx_{1: \tau},$$

which implies that our estimate of (5.1.2) will be unaffected by the addition of the auxiliary variable $V$.

We still want to ultimately sample from (5.1.2). So, by adding $V$ to the state space of the target (5.1.3), we obtain a target probability density for an adaptive ML-PMMH as

$$\pi^N \left( k, \theta, \mathcal{X}_{1:p(v)}, \omega_{1:p(v)-1}, v \right) = \frac{\pi(\theta, \mathcal{X}^k_{1:p(v)})}{N_{p(v)}} \times \frac{\psi_\theta(\mathcal{X}_{1:p(v)}, \omega_{1:p(v)-1}) \Lambda_\theta(v)}{q_1 \left( \mathcal{X}^k \right) \left( \prod_{n=2}^{p(v)} w_{n-1} \right) \Lambda_n \left( \mathcal{X}_n \mid \mathcal{X}^{a(k)}_{1:n-1} \right)},$$

Note that the auxiliary process associated with $V$ will be conditionally independent of $k$, $\mathcal{X}_{1:p}$, and $\omega_{1:p-1}$ given $\theta$. An adaptive ML-PMMH algorithm with proposal density

$$q^N \left( k, \theta, \mathcal{X}_{1:p(v)}, \omega_{1:p(v)-1}, v \right) = q \left( \theta \mid \zeta \right) \Lambda_\theta(v) \psi_\theta(\mathcal{X}_{1:p(v)}, \omega_{1:p(v)-1}) w_p^{k(v)}$$

is given as Algorithm 36. [50] present convergence results for this algorithm and prove its validity.

Note that in computational statistics the word “adaptive” is typically used to label algorithms whose transition kernels change based on the past history of the chain. Our adaptive ML-PMMH is not adaptive in this sense.

### 5.3 Implementation of the multi-level PMMH algorithms

We consider two examples from population genetics to compare different PMMH implementations. The first example is a coalescent model with mutation that has a low dimensional observed dataset. The second example is a more challenging coalescent model with mutation and migration that has a larger dataset. Both models were introduced in Section
Algorithm 36 Adaptive multi-level particle marginal Metropolis-Hastings

- Step 0: Draw $\theta$ from its prior. All remaining random variables can be sampled from their full conditionals defined by the target (5.2.1):
  - Sample $v \sim \Lambda_\theta(\cdot)$. Based on $v$, set the sequence of $\mathcal{F}$-sets $\{B_n(v)\}_{1 \leq n \leq p(v)}$.
  - Sample $X_{1:p(v)}^{\theta}, a_{1:p(v) - 1}^{\theta} | \cdots \sim \psi_\theta(\cdot)$ via Algorithm 17, excluding the final resampling step. Use the backward methodology of Section 2.11.1.
  - Choose $k \propto W_{k}^{\theta} p(v)$.

Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta, 1:p(v)}$, via (2.11.1).

- Step 1: Sample $\theta^* \sim q(\cdot | \theta)$. All remaining random variables can be sampled from their full conditionals defined by the target (5.2.1):
  - Sample $v^* \sim \Lambda_\theta(\cdot)$. Based on $v^*$, set the sequence of $\mathcal{F}$-sets $\{B_n(v^*)\}_{1 \leq n \leq p(v^*)}$.
  - Sample $X_{1:p(v^*)}^{\theta^*}, a_{1:p(v^*) - 1}^{\theta^*} | \cdots \sim \psi_\theta(\cdot)$ via Algorithm 17, excluding the final resampling step. Use the backward methodology of Section 2.11.1.
  - Choose $k^* \propto W_{k}^{\theta^*} p(v^*)$.

Finally, calculate the marginal likelihood estimate, $\hat{Z}_{\theta^*, 1:p(v^*)}$, via (2.11.1).

- Step 2: With acceptance probability

$$1 \wedge \frac{\pi(\theta^*) q(\theta | \theta^*)}{\pi(\theta) q(\theta^* | \theta)} \frac{\hat{Z}_{\theta^*, 1:p(v^*)}}{\hat{Z}_{\theta, 1:p(v)}},$$

set $v = v^*, k = k^*, \theta = \theta^*, X_{1:p(v)} = X_{1:p(v^*)}^{\theta^*}$, and $a_{1:p(v) - 1} = a_{1:p(v^*) - 1}^{\theta^*}$.

Return to the beginning of Step 1.
2.3.1. We illustrate the performance of the following PMMH algorithms on the coalescent with mutation:

- Fixed ML-PMMH: see Algorithm 35.
- Adaptive ML-PMMH: see Algorithm 36.
- Generic PMMH: see Algorithm 22. To clarify, we code a version of the PMMH sampler that uses an SMC algorithm (similar to Algorithm 3) that does not rely on any $\mathcal{F}$—sets but still uses the backward sampling scheme of Section 2.11.1. Essentially, this algorithm resamples after every genetic event (i.e., at every time-point of the SMC algorithm).

For the coalescent with mutation and migration, we again compare the performance of the adaptive ML-PMMH to the generic PMMH, but we do not test the fixed ML-PMMH algorithm.

In each example, we aim to compare the convergence and the mixing of the ML-PMMH algorithms to the generic PMMH. In the first example, we also illustrate the advantages of adaptive over fixed ML-PMMH, in regard to mixing and convergence rates. To track such quantities, we present the autocorrelation functions and the trace plots for several components of $\theta$. We also plot the estimated probability density functions of several components of $\theta$ given the data to track each algorithm’s exploration of its target’s state space.

5.3.1 Coalescent with mutation

We set the model as presented in Section 2.3.1, using a known stochastic matrix $R$ with all entries equal to $1/d$. In this example, $d = 4$ and the dataset is $y = (10, 5, 9, 5)$ (note that this is a small toy example and the next numerical example uses a larger dataset; this first example serves more as a proof of principle). We attempt to infer $\mu \in \Theta = [0, 1.5]$, where $\mu$ has prior $U(0, 1.5)$. All simulations employ the backward sampling scheme of Section 2.11.1, and the SMC algorithms use the optimal proposal distributions of [85]. When appropriate, the $\mathcal{F}$—sets take a form similar to (2.11.6). Also, for all of the algorithms
5.3 Implementation of the multi-level PMMH algorithms

described in this section, the PMMH proposal is a log normal random walk: \( \log(\theta^*) = \log(\theta) + 0.4\epsilon, \epsilon \sim N(0,1) \).

Several versions of the fixed ML-PMMH are implemented, with the number of \( \mathcal{F} \) – sets denoted by \( p \) and the number of particles in the SMC denoted by \( N \). For \( N = 50, 100, \) and 200, \( p \) is set to equal eight, 14, 21, and then 28, for a total of twelve implementations. For each implementation, the number of iterations of the ML-PMMH is \( M = 100000 \). Also, each simulation is repeated twenty times. The \( \mathcal{F} \) – sets are always roughly equally spaced apart for a given value of \( p \).

Several versions of the adaptive ML-PMMH (where \( p \) is allowed to vary) are also implemented. For \( N = 50, 100, \) and 200, we allow \( p \in \{8, 14, 21, 28\} \) and \( p \in \{8, 9, \ldots, 28\} \), for a total of six implementations. \( M = 100000 \) for each implementation, and each simulation is repeated twenty times. We always place the \( \mathcal{F} \) – sets almost equally spaced apart for a given value of \( p \), and we sample \( p \) directly using a multinomial distribution defined on \( p \)’s range with probabilities proportional to \( \mu^p \).

Only one version of the generic PMMH is implemented for each \( N \) value of 50, 100, and 200, with \( M = 100000 \). We only run each simulation once, for reasons which will become clear in the discussion of the results below.

For \( N = 50 \), all algorithms require approximately two to four hours to complete, while for \( N = 100 \) and 200, the algorithms require approximately four to eight hours and eight to 16 hours, respectively. Those algorithms that resample the least fall on the shorter end of the spectrum. The algorithms are implemented in MATLAB 7.14 R2012a and run on a Windows desktop using an Intel Core i7-2600 CPU at 3.40 GHz.

Results

- Fixed ML-PMMH: see Figures 5.3.1, 5.3.2, 5.3.3
- Adaptive ML-PMMH: see Figures 5.3.1, 5.3.2, 5.3.3
- Generic PMMH: results not pictured

The two ML-PMMH algorithms demonstrate fast mixing, as is illustrated by the ACFs (see Figure 5.3.2). The adaptive ML-PMMH exhibits faster mixing when \( p \) is chosen from a
Chapter 5. Inference for Partially Observed Stopped Markov Processes 211

smaller range. This is expected because expanding the range of $p$ also increases the size of the state space of the extended target (5.2.1). Furthermore, the trace plots (see Figure 5.3.3) show both multi-level algorithms are non-sticky. Increasing values of $N$ do yield slightly faster convergence rates in the fixed ML-PMMH, but the adaptive ML-PMMH’s rate of mixing is approximately constant given $N$.

The two ML-PMMH algorithms do yield significantly different estimates of the pdf of $\mu$ (see Figure 5.3.1). The fixed ML-PMMH give a consistent estimate of the likelihood for all fixed values of $p$, regardless of the value of $N$. In all cases, the density appears to have a single mode close to zero. However, the adaptive ML-PMMH reveals a different mode when $p$ is allowed to be chosen from the range $\{8, 14, 21, 28\}$. In that case, the density appears to have a single mode close to one. When we allow $p$ to be chosen from the larger range $\{8, 9, \ldots, 28\}$, the adaptive ML-PMMH is able to find both modes simultaneously. Thus, it would seem a finer distribution on $p$ (i.e., the case $p \in \{8, 9, \ldots, 28\}$) facilitates the movement between these aforementioned modes. These results are consistent across the many repetitions.

The output illustrates, for this example, that adapting $p$ allows ML-PMMH to better traverse the state space compared to when a fixed number of $\mathcal{F}$—sets is used. To obtain further confidence that a non-adaptive algorithm is insufficient for this first example, we implement seven more versions of the fixed ML-PMMH with $p$ fixed at various other values in the range of eight to 28 (results not shown). Each algorithm is only run once with $N = 50$. In all instances, the same single mode estimate of the likelihood is obtained.

Furthermore, the size of the range of $p$ has a clear impact on the performance of the adaptive ML-PMMH. The two implementations of the adaptive algorithm seem to suggest that larger ranges for $p$ yield more efficient PMMH, possibly because it is easier to tailor SMC to changing values of $\theta$ through a more refined range for $p$.

Generic PMMH fails in this example for $N = 50, 100,$ and 200. After 100000 iterations, generic PMMH does not come close to converging. Plots of the ACF with a lag of 1000 (not shown) never drop below 0.8, and the average acceptance rates are on the order of 0.02% to 0.4%.
Figure 5.3.1: We present the output for $N = 50$ (red), 100 (blue), and 200 (green). Left: fixing $p = 8$. Middle: allowing $p \in \{8, 14, 21, 28\}$. Right: allowing $p \in \{8, 9, \ldots, 28\}$. In the top row, we select one run from each algorithm for each value of $N$ and compare the estimated pdfs of $\mu$. In the latter rows, we plot the results of all 20 repetitions per value of $N$, with black lines representing the average across the repetitions.
Figure 5.3.2: We present the output for $N = 50$ (red), 100 (blue), and 200 (green). Left: fixing $p = 8$. Middle: allowing $p \in \{8, 14, 21, 28\}$. Right: allowing $p \in \{8, 9, \ldots, 28\}$. Each plot shows the ACFs for 20 repetitions.
5.3 Implementation of the multi-level PMMH algorithms

Figure 5.3.3: We present the output for $N = 50$ (red), 100 (blue), and 200 (green). Left: fixing $p = 8$. Middle: allowing $p \in \{8, 14, 21, 28\}$. Right: allowing $p \in \{8, 9, \ldots, 28\}$. For $p = 8$, the average acceptance ratio is on the order of 0.07, 0.09, and 0.10, for $N = 50, 100, \text{and} 200$, respectively. For $p \in \{8, 14, 21, 28\}$, the average acceptance ratio is always about 0.07, regardless of $N$. For $p \in \{8, 9, \ldots, 28\}$, the average acceptance ratio is 0.09, 0.11, and 0.12, for $N = 50, 100, \text{and} 200$, respectively.

5.3.2 Coalescent with mutation and migration

We set the model as presented in Section 2.3.1, using data generated with $m = 100$, $d = 64$, and $g = 3$ (see Figure 5.3.4). We set the mutation matrix $R$ to be known and uniform. Such an $R$ is realistic if one assumes a Jukes-Cantor-type model of evolution [37, Chapter 11]. We concentrate on inferring $\theta = (\mu, G)$, with true values of $\mu = 0.5$, $g_{12} = 0.5$, \ldots
$g_{23} = 0.25$ and $g_{13} = 0.25$. Each of the components of $\theta$ has a $\mathcal{G}_0(1, 1)$ prior. In all sampling strategies, time is reversed and the backward sampling scheme of Section 2.11.1 is employed. When appropriate, the $\mathcal{F}$—sets take a form similar to (2.11.7). More specific details of the implementation can be found in [22].

Figure 5.3.4: Dataset for the coalescent with mutation and migration. The types $(g \times d)$ run along the horizontal axis, and the height of each bar represents the count of each genetic type at time $\tau$.

We implement several versions of the adaptive ML-PMMH. For $N = 10, 50, 100,$ and 200, we allow either

\[ p \in \{10, 20, 33\}, \]
\[ p \in \{10, 16, 21, 27, 33\}, \text{ or} \]
\[ p \in \{10, 13, 16, 19, 21, 24, 27, 30, 33\}, \]

for a total of twelve implementations. For each implementation, $M = 100000$ and each simulation is repeated ten times. We always place the $\mathcal{F}$—sets almost equally spaced apart for a given value of $p$, and $p$ is chosen with probability proportional to $p^{\log(\mu + \sum_{i>j} G_{ij} + 1)}$. 
The proposals for $\mu$ and the elements of $G$ are log normal random walks: $\log(\theta^*) = \log(\theta) + \epsilon$, $\epsilon \sim N(0, 1)$.

Similar to the experiments in Section 5.3.1, only one version of the generic PMMH is run for $N = 10, 50, 100, \text{and} 200$. One batch of simulations sets $M = 100000$, and the other batch uses $M = 500000$. We only run each simulation once.

The simulations run on a Linux workstation that uses an Intel Core 2 Quad Q9550 CPU at 2.83 GHz. When using multi-level SMC with $p \in \{10, 20, 33\}$, the running time for each algorithm is approximately 36, 180, 360, and 720 minutes for $N = 10, 50, 100, \text{and} 200$, respectively. When we allow $p \in \{10, 16, 21, 27, 33\}$, the running times increase by about 50\% in each case, as opposed to a 100\% increase when $p \in \{10, 13, 16, 19, 21, 24, 27, 30, 33\}$.

Results

- Adaptive ML-PMMH: see Figures 5.3.5, 5.3.6, 5.3.7
- Generic PMMH: results not pictured

The adaptive ML-PMMH algorithms give consistent output regardless of the way in which we choose $p$, and so we only present the output for the case in which $p \in \{10, 16, 21, 27, 33\}$. The estimated density of $\theta$ appears to be accurate across the algorithms (see Figure 5.3.5), and each version of ML-PMMH mixes quickly and efficiently (see the ACF and trace plots of Figures 5.3.6 and 5.3.7). The algorithm converges a bit more slowly as we increase the size of the range of $p$, much like in the example in Section 5.3.1. Given the algorithms’ ability to mix so well, we find the output to be consistent for $N \in \{10, 50, 100, 200\}$ and to be reproducible amongst the ten repeated simulations.

PMMH using generic SMC performs quite poorly again. We find that after 100000 iterations, and even after 500000 iterations, PMMH using generic SMC completely fails to converge. Plots of the ACF with a lag of 200 (not shown) never drop below 0.7, and the average acceptance rate is on the order of 0.01\% to 0.1\% (the average acceptance rate is on the order of 15\% to 20\% for ML-PMMH).
Choosing \( p \in \{10, 16, 21, 27, 33\} \)

![Graphs showing estimated pdfs for different values of N: 10 (black), 50 (red), 100 (blue), and 200 (green).](image)

Figure 5.3.5: Results for \( N = 10 \) (black), 50 (red), 100 (blue), and 200 (green). Estimated pdfs for \( \mu \), \( g_{12} \), \( g_{23} \), and \( g_{13} \) (clockwise from top left corner). Only one of the ten repetitions of each simulation is shown.
Choosing $p \in \{10, 16, 21, 27, 33\}$

Figure 5.3.6: Results for $N = 10$ (black), 50 (red), 100 (blue), and 200 (green). ACF plots for $\mu$, $g_{12}$, $g_{23}$, and $g_{13}$ (clockwise from top left corner). Only one of the ten repetitions of each simulation is shown.
Choosing \( p \in \{10, 16, 21, 27, 33\} \)

Figure 5.3.7: Results for \( N = 100 \). Trace plots for \( \mu, g_{12}, g_{23}, \) and \( g_{13} \) (clockwise from top left corner). Only one of the ten repetitions of each simulation is shown.

### 5.4 Discussion

In this chapter, we used multi-level SMC within PMCMC to develop fixed and adaptive ML-PMMH algorithms that are tailored for inference on partially observed stochastic processes that start from an initial set and are stopped at the first hitting time of a target set. The algorithms were implemented on two examples from population genetics: the simple coalescent with mutation and the more advanced coalescent with mutation and migration. Our multi-level algorithms performed well in numerical testing. We demonstrated that ML-PMMH can greatly outperform generic PMMH when one is sampling from a stopped Markov process. The tests also illustrated the significant increase in precision that the
adaptive scheme is able to provide over the fixed scheme. These results are encouraging, as computational Bayesian inference has not been attempted for this class of problems before.

Finally, we note that while this chapter focuses on PMMH, extensions using PG (recall Algorithm 24) might prove valuable for other specific applications. In the case where one uses fixed $F$—sets, we have already defined the extended PMCMC target (5.1.3) and a conditional version of Algorithm 17 is straightforward. Thus, following the lines of the transition from PMMH to PG in Section 2.18, one can easily formulate a fixed multi-level PG. In the case where one uses adaptive $F$—sets, a little more fine tuning is required in exactly how the the conditional version of Algorithm 17 would be structured. We address this point in the concluding chapter of the thesis.
Chapter 6

Summary and Future Work

The main aim of this thesis was to make contributions to PMCMC algorithms which were specifically designed for sampling from HMMs and partially observed, discrete time stopped Markov processes. The basic strategy followed in each of three original works was to identify an SMC algorithm that performed with low variance for the particular model at hand (when the parameters for the model are known) and then embed that SMC technique in a PMMH algorithm. We also followed the same strategy to develop some PG samplers. In each work, it was necessary to clearly state how one could use SMC to compute unbiased estimates of the model’s likelihood, for otherwise a PMMH algorithm would not be obtainable.

We close with just a few remarks on how each research chapter was in line with the main aim of the thesis, and we also discuss some possible extensions of our work in each case.

6.1 Review of Chapter 3

The first research project presented a new unbiased estimate of the HMM’s normalising constant and illustrated how it could be calculated using the $O(N)$ two-filter smoother of [36]. We then used this estimate to develop new $O(N)$ PMMH and PG algorithms by placing the two-filter smoother of [36] within PMCMC. Numerical examples showed that our PMCMC algorithms performed best when the pseudo-priors $\{\xi_{n,\theta}\}_{n \geq 1}$ of the smoother’s
backward filter took the form \( \{ \pi(x_n | y_{1:n-1}) \}_{n \geq 1} \). As the sequence \( \{ \pi(x_n | y_{1:n-1}) \}_{n \geq 1} \) is only analytically available for a subset of HMMs, we developed three alternative smoothing algorithms (one biased and two unbiased) that could be used to approximate these ideal settings. Numerical tests showed that none of those alternative methods were able to mimic the ideal settings very well.

We briefly mentioned at the end of Chapter 3 that it may be worthwhile to make other attempts at approximating the ideal backward SMC smoother of Section 3.3, as our two alternative unbiased smoothing algorithms seemed to have a high variance by construction. Furthermore, a future work might consider properly comparing Algorithms 31 and 32 to the standard unbiased FFBS algorithms and the generalized two-filter smoothers for the purposes of testing whether or not Algorithms 31 and 32 are superior smoothing strategies. Such a comparison was not conducted in this work because it runs too far off topic from the thesis’s main goal of making contributions to the PMCMC methodology. Even if a comparison of Algorithms 31 and 32 to other smoothers told us that our algorithms were superior smoothing strategies, we already explained at the end of Section 3.7.1 that Algorithms 31 and 32 are not suited to be used within PMCMC.

### 6.2 Review of Chapter 4

Chapter 4 continued the focus on HMMs, only now we considered HMMs whose observations have unknown or intractable likelihood densities. These types of models can arise in finance when one wants to place a skewed or heavy tailed stable distribution on a time series of returns and the particular choice of stable distribution does not have a known density function. Standard SMC and PMCMC methods cannot be applied to such models, and so one would normally resort to ABC approximations. In this chapter, we combined the works of [51] and [91] to develop lower variance ABC approximations that draw from these models. Basically, we introduced a twisted version of Algorithm 15 in order to develop a twisted version of the alive PMMH of [51]. Adopting similar assumptions as in [91], we demonstrated that the same theoretical arguments of [91] can be used to determine the unique, optimal twisting function for our new ABC algorithms. This optimal choice also happens to be the same unique choice discovered in [91].
When the true variances of the HMM are less than or equal to one and when one has
access to an accurate approximation of the optimal twisting function, our alive twisted par-
ticle filter produced lower variance estimates of the normalising constant of a linear Gauss-
ian HMM when compared to the non-twisted Algorithm 15. Similarly, our alive twisted
PMMH converged slightly faster than the non-twisted PMMH appearing in [51] (again,
when an accurate approximation of the optimal twisting function was used). The greatest
insight from the numerical simulations is perhaps that when the change of measure on the
alive algorithms is not a close approximation of the ideal change in measure, twisting may
not be worthwhile. The alive twisted algorithms have the potential to perform well when a
good approximation of the ideal change in measure is at hand, but such an approximation
is only available for a limited group of HMMs.

We also briefly mention that, using the assumptions adopted in Chapter 4, one can
follow the same proof presented in that chapter’s Section A to prove the same theoretical
results for a twisted version of Algorithm 14. Such an algorithm (and the accompanying
PMMH) might be of interest in a situation where the stochastic running time of Algorithm
15 is undesirable. Further investigation would be required, but it may be the case that
twisting Algorithm 14 helps mitigate its chances of dying out (for it would run with a
reduced variance).

6.3 Review of Chapter 5

The final research chapter switched the focus from HMMs to discrete time stopped Markov
processes that are only partially observed. We developed PMMH algorithms to sample
from these processes, thereby expanding the applicability of PMCMC to a new class of
problems and attempting Bayesian inference on a class of problems for which it had not
been attempted before.

Embedding multi-level SMC within PMCMC, we formalised fixed and adaptive ML-
PMMH algorithms that are tailored for inference on partially observed stochastic processes.
When implemented on two examples from population genetics (i.e., the coalescent with
mutation and the more advanced coalescent with mutation and migration), our multi-level
algorithms performed well. ML-PMMH can greatly outperform generic PMMH when sam-
...
pling from either stopped Markov process, and the tests also illustrated the significant increase in precision that the adaptive scheme is able to provide over the fixed scheme.

Chapter 5 did not consider multi-level PG samplers. While structuring a fixed scheme multi-level PG is straightforward, the adaptive version requires more investigation. Consider the following. Algorithm 36 uses a random number of level-sets from iteration to iteration. Actually, the level-sets used from iteration to iteration could have very different structures as well. A PG analogue to Algorithm 36 could also use different level-sets from iteration to iteration, and so the accompanying conditional version of Algorithm 17 must account for this. The conditional version of Algorithm 17 would have to allow for the case where the sampled paths have different level-sets than the path on which the algorithm is conditioning.

Now, the process \( \{X_n\}_{1 \leq n \leq p} \) is just a construction that we use to run multi-level SMC. The true process of interest is \( \{X_j\}_{1 \leq j \leq \tau} \). Thus, it is samples of \( \{X_j\}_{1 \leq j \leq \tau} \) that the multi-level SMC is producing, and it is a sample of \( \{X_j\}_{1 \leq j \leq \tau} \) on which the multi-level SMC is conditioning. In the numerical examples of Section 5.3, any slight shifting of the level-sets 2.11.6 and 2.11.7 could always fit any sample of \( \{X_j\}_{1 \leq j \leq \tau} \). So, in the case of those numerical examples, one could run conditional multi-level SMC where, when necessary, the level-sets of a sample on which the algorithm conditions are adjusted to match the structure of the new samples. In the general case, this scheme will not work, as there is nothing to say that a previously sampled path can fit any scheme of interpolating sets. A future work might address this issue, with the ultimate intention of forming an adaptive multi-level PG sampler that works in the general case.

We close by pointing out that a non-adaptive PG sampler that can be used to perform inference on Kingman’s coalescent is now available in the wider literature. In [18], the authors adopt a representation of the coalescent in which the state space considers the coalescent events and their times but ignores the mutations and their times. This is the same representation as in, for example, [61]. The PG sampler of [18] runs by iteratively sampling the genealogy structure of the coalescent tree and the times at which coalescent events occur. Recall that in this thesis, we adopted a competing representation of the coalescent. The state spaces of our models considered the coalescent and mutation events but ignored the times (as in [44] and [85]).
References


REFERENCES


REFERENCES


