Particle Markov Chain Monte Carlo

by

Roman Holenstein

B.Sc. Physics/Computer Science, University of Northern British Columbia, 2002

M.Sc. Electrical Engineering, University of Alberta, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University Of British Columbia

(Vancouver)

March 2009

Abstract

Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC) methods have emerged as the two main tools to sample from high-dimensional probability distributions. Although asymptotic convergence of MCMC algorithms is ensured under weak assumptions, the performance of these latters is unreliable when the proposal distributions used to explore the space are poorly chosen and/or if highly correlated variables are updated independently. In this thesis we propose a new Monte Carlo framework in which we build efficient high-dimensional proposal distributions using SMC methods. This allows us to design effective MCMC algorithms in complex scenarios where standard strategies fail. We demonstrate these algorithms on a number of example problems, including simulated tempering, nonlinear non-Gaussian state-space model, and protein folding.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Glossary

Notation

X	a random variable (upper case)
x	a value (lower case)
$\mathbf{X}, \mathbf{x}$	a vector
$\mathcal{B}(E)$	Borel σ-algebra on E
$\mathbb{P}(A)$	probability of event A
$\pi(d\mathbf{x})$	probability distribution
$\pi(\mathbf{x})$	probability density
$(E, \mathcal{F})$	measurable space (σ-field) on set E and σ-algebra $\mathcal{F}$
$\mathbb{E}_\pi(\cdot)$	expectation with respect to distribution π
$\text{var}_\pi(\cdot)$	variance with respect to distribution π
$\delta_x(dx)$	Dirac delta function
$\mathcal{N}(x; \mu, \sigma^2)$	Normal (Gaussian) distribution with mean μ and variance σ^2
$\mathcal{B}(x; N, p)$	Binomial distribution for N trials with success probability p
$\mathcal{M}(x; N, \mathbf{p})$	Multinomial distribution for N trials with event probabilities $\mathbf{p} = (p_1, ..., p_k)$
$\mathcal{G}a(x; k, \theta)$	Gamma distribution with shape $k > 0$ and scale $\theta > 0$
$\mathcal{IG}a(x; \alpha, \beta)$	Inverse Gamma distribution with shape $\alpha > 0$ and scale $\beta > 0$
$\mathcal{B}e(x; \alpha, \beta)$	Beta distribution with shape parameters $\alpha > 0$ and $\beta > 0$
$\mathcal{TS}(\kappa, \delta, \gamma)$	tempered-stable distribution

Acronyms

AC	auto-correlation
ACF	auto-correlation function
ACO	ant colony optimisation
CBMC	configurational bias Monte Carlo
DP	Dirichlet process
DPERM	dynamic prune-enriched Rosenbluth method
GPU	graphics processing unit
EKF	extended Kalman filter
ESS	effective sample size
FRESS	fragment regrowth via energy-guided sequential sampling [76]
G	Gibbs
HMM	hidden Markov model
LV	Lotka-Volterra, predator-prey model
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
MH	Metropolis Hastings
MJP	Markov jump process
MTM	multiple-try Metropolis
NLSS	non-linear state-space model
NTT	Neal's tempered transitions, MCMC algorithm to sample from multimodal distributions developed by Radford Neal [66]
OU	Ornstein-Uhlenbeck process
PDF	probability distribution function
PERM	prune-enriched Rosenbluth method

PG particle Gibbs

PMCMC particle Markov chain Monte Carlo

PMH particle Metropolis Hastings

PMMH particle marginal Metropolis Hastings

REMC replica exchange Monte Carlo

RG recoil-growth

RJMCMC reversible-jump MCMC

S&P 500 Standard & Poor's 500 index

SA simulated annealing

SAW self-avoiding walk

SDE stochastic differential equation

SIMCMC sequentially interacting Markov chain Monte Carlo

SMC sequential Monte Carlo

SSM state-space model

SV stochastic volatility

UKS unscented Kalman smoother

Acknowledgements

I am extremely grateful to my supervisor Arnaud Doucet. His enthusiasm, his support, his honest critiques, his sound advice, and his deep insight have helped me greatly and this thesis would not have been possible without him. He is an inspiration and I am very honoured to call him my teacher and friend.

I would like to thank Christophe Andrieu who has also been integral to the development of the methodology in this thesis.

Many thanks go to the numerous people who have been my teachers and mentors along my academic journey so far. I would especially like to express my gratitude to: Mark Shegelski (undergraduate research supervisor), Robert Fedosejevs and Ying Tsui (M.Sc. supervisors), Raphael Gottardo (committee member), as well as Nando de Freitas (committee member), who introduced me to the fun of Monte Carlo and machine learning. I am very grateful for their generous support, advise, and help with various projects.

Furthermore I would like to thank my student colleagues and friends for great discussions and for providing a fun and stimulating learning environment. Special thanks go to Aline Tabet, Emtiyaz Khan, François Caron, Frank Hutter, Gareth Peters, Glen Goodvin, Hoyt Koepke, Luke Bornn, Mark Schmidt, Mark Crowley, Matt Hoffman, Mike Chiang, and Mike Whitwick.

The secretaries at the CS department deserve my gratitude for providing assistance and administrative support, with special mention to Joyce Poon, Lara Hall, and Kath Imhiran.

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), the University of British Columbia, the Department of Computer Science, and the Province of British Columbia. I would also like to

acknowledge the Western Canada Research Grid (WestGrid) for providing computational resources.

Last but not least, I am forever grateful to my parents and siblings for their continuous love and support, especially my mother who has always been there for me and helped me grow as a person. Finally, I want to thank my darling Elaine Qing Chang for her unbounded encouragement, support, and love.

Chapter 1

Introduction

Monte Carlo methods have become the standard tool to solve many problems in statistics and scientific computing. Examples are abound, and include instances in Bayesian statistics (posterior estimation), statistical physics (Ising model), particle physics (simulation of high energy particles interacting with a detector), biology (protein folding, Lotka-Volterra model), chemistry (chemical reaction networks), and finance (option valuation), to name a few. The problems are becoming more and more sophisticated and Monte Carlo methods are now expected to deal with high dimensionalities and complex interactions between the model variables.

In this thesis we present a new framework [3] that allows us to expand the class of problems that can routinely be addressed with Monte Carlo methods. It is based on a non-trivial and novel combination of existing sampling strategies and takes advantage of their strengths. We then apply the method to problems in Bayesian statistics and biology and make comparisons with competitive algorithms.

Assume we are given a probability distribution $\pi(d\mathbf{x})$ defined on a measurable space $(E, \mathcal{F})$ and that we are interested in sampling from this distribution, usually to compute analytically intractable expectations of interest with respect to $\pi(d\mathbf{x})$ by invoking the law of large numbers [61, 68]. For ease of notation, we shall further assume that $\pi(d\mathbf{x})$ admits a density $\pi(\mathbf{x})$ with respect to a σ-finite dominating measure denoted $d\mathbf{x}$.

Monte Carlo sampling is generally done by proposing samples from some instrumental distribution. A weighing and/or selection mechanism then corrects for

the bias from the proposal distribution. As the dimension increases, finding a good proposal distribution becomes harder. A good proposal distribution should take into account key features of the target, such as multimodality, and ideally should be as close as possible to the target.

There are two classes of Monte Carlo algorithms that are often used for sampling from high-dimensional distributions: Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC). MCMC relies on sampling a realization of a Markov chain with invariant distribution $\pi(d\mathbf{x})$. SMC is a sequential implementation of importance sampling, employing a population of samples (particles) and a sequence of probability distributions (of increasing dimension), with the final distribution being the target distribution to generate samples from $\pi(d\mathbf{x})$. When the consecutive distributions in the sequence are not too different one can find good proposals to move the particles from one distribution to the next. An introduction to this method is given in section 2.2. For a book-length review see [27]. While initially designed for on-line inference in dynamic models, usually referred to as "particle filters" in that context, they are now also used in static models [15, 32, 65]. SMC samplers have found a wide range of applications in fields as diverse as econometrics [16], [67], ecology [12], and systems biology [48]. It was realised later that the interest of SMC methods is not limited to dynamic models and they are now increasingly used to perform inference for a wide range of models including contingency tables [15], mixtures models [32, 65], graphical models [53], and population genetics [61, Section 4.1.2]. Where traditional importance sampling would try to directly produce weighted samples on E to approximate $\pi(\mathbf{x})$, and most likely fail for the same reason that an independent Metropolis Hastings (MH) algorithm would fail, an SMC algorithm will adopt a more progressive approach which can be beneficial in large dimensional scenarios. The first ingredient of such an algorithm is a sequence of intermediate "bridging" probability distributions of increasing dimensions $\{\pi_n(d\mathbf{x}_n),\ n = 1, \ldots, p-1\}$ with $\mathbf{x}_n = (x_1, x_2, \ldots, x_n) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ and such that $\pi_n(d\mathbf{x}_n) = \pi_n(\mathbf{x}_n)\, d\mathbf{x}_n$ where $d\mathbf{x}_n = d\mathbf{x}_{n-1} \times dx_n$ and $\pi_p(\mathbf{x}_p) = \pi(\mathbf{x})$.

However, sampling sequentially comes at a price. Indeed, if p is too large, then the SMC approximation of the joint density $\pi(\mathbf{x})$ deteriorates as components imputed at any time $n < p$ are not rejuvenated at subsequent time steps. As a

result, when $p - n$ is too large the approximation of the marginal $\pi(\mathbf{x}_n)$ is likely to be rather poor as the successive resampling steps deplete the number of distinct particles. However, despite the potential drawbacks outlined above, SMC methods have had a major impact on inference in dynamic systems in the last fifteen years, and are now showing great promise for more classical "static" inference problems *e.g.* [15], [65], [32], [53] and [61, Section 4.1.2].

This thesis proposes a novel sampling strategy which aims to combine the advantages of MCMC and SMC methods. As we shall see, this opens up the possibility to routinely tackle problems that cannot be satisfactorily addressed using either of these approaches on its own. Several algorithms combining both approaches have already been proposed in the literature. In particular MCMC algorithms have been successfully used as updating mechanisms or proposals within SMC methods [45]; see also [17], [65]. Our approach is entirely different because on the contrary it proposes to use SMC algorithms as a proposal mechanism within MCMC methods. We use MCMC methods to allow us to "break" the complex task of sampling from $\pi(\mathbf{x})$ into a series of simulations from lower-dimensional distributions whereas we use SMC ideas to efficiently update very large sub-blocks of $\mathbf{x}$. Although this idea might seem natural, it is important to note that its realisation is far from obvious. Indeed, a direct implementation of such a strategy is impossible as the marginal distribution of a particle generated by an SMC algorithm is not available in closed-form but would be required for the implementation. To bypass this problem we introduce a non-standard auxiliary target distribution on an extended space which allows us to define a valid MH update and from which inference about $\pi(\mathbf{x})$ can be carried out. We show that the resulting algorithm enjoys attractive properties and can be used as a component of more advanced algorithms.

The rest of this thesis is organised as follows. In Chapter 2 we give a brief review of MCMC and SMC and provide some convergence results to motivate our new method. Note that SMC is presented in a non-standard notation in order to simplify the proofs for the particle Markov chain Monte Carlo (PMCMC) algorithms. Chapter 3 introduces the PMCMC framework, starting with the particle Metropolis Hastings (PMH) sampler, followed by the particle Gibbs (PG) and particle marginal Metropolis Hastings (PMMH) samplers. The PMH sampler (Section 3.1) is a novel Metropolis Hastings sampler which uses SMC algorithms as proposals to sample

from $\pi(\mathbf{x})$. The PG sampler (Section 3.2) extends the PMH algorithm to allow sampling from distributions of the form $\pi(\theta, \mathbf{x})$ (defined on some space $\Theta \times E$), which can be understood as being an approximation of the standard Gibbs sampler (which alternates sampling from the full conditionals $\pi(\theta|\mathbf{x})$ and $\pi(\mathbf{x}|\theta)$), but has the property that it leaves $\pi(\theta, \mathbf{x})$ invariant, and is hence not biased. In cases where θ and $\mathbf{x}$ are strongly correlated, this approach is likely inefficient. This is addressed by the PMMH algorithm (Section 3.3), which is an unbiased approximation of a MH algorithm that works directly on the marginal space Θ. In Chapter 4 we present a variety of applications to which this methodology can be applied and make comparisons with competitive alternative algorithms. In Section 4.1 we apply PMCMC to a non-linear state-space model. Using PMH with simulated annealing we find optimal configurations in a protein folding model in Section 4.2. We perform clustering using Dirichlet mixture models in Section 4.3. We estimate parameters for the Lotka-Volterra (LV) model in Section 4.4. In Section 4.5 we consider a Lévy-driven stochastic volatility (SV) model for the Standard & Poor's 500 (S&P 500) data set. And in Section 4.6 we use PMH with tempering to sample from highly multimodal distributions. Finally Chapter 5 summarises the contributions of this thesis and outlines further extensions.

Chapter 2

Review of Monte Carlo Methods

2.1 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods are a class of algorithms based on constructing a Markov chain which has the desired target distribution as its equilibrium distribution. Two very popular MCMC algorithms are the Metropolis Hastings (MH) and the Gibbs (G) sampler [68]. While originally mainly used in physics, these algorithms soon found widespread use in many other disciplines. Often they are used to solve integration and optimisation problems, for example computing expectations in Bayesian statistics by invoking the *law of large numbers*.

In MCMC, a Markov chain is constructed from a transition kernel $K(\mathbf{x}_n, \mathbf{x}_{n+1})$. This kernel is defined on $(E, \mathcal{B}(E))$ such that $K(\mathbf{x}, \cdot)$ is a probability measure for all $\mathbf{X} \in E$. In the discrete domain the kernel is simply a transition matrix.

In order to establish that the Markov chain converges to the required target distribution we require the following three properties:

- The invariant distribution of the Markov kernel is $\pi(\cdot)$
- The Markov chain is *irreducible*: any subset of $A \subset E$ for which $\pi(A) > 0$ is reachable from any starting point $\mathbf{x}(0) \in E$
- The Markov chain is *aperiodic*

Next we will (very) briefly introduce the MH and Gibbs samplers.

2.1.1 Metropolis-Hastings

The MH sampler uses an instrumental proposal distribution to generate candidates that are accepted or rejected such that the resulting Markov chain has the target $\pi(\mathbf{x}) = \gamma(\mathbf{x})/Z$ as the invariant distribution.

Algorithm 2.1: Metropolis-Hastings

1 Initialise $\mathbf{X}(0)$

2 **For iteration** $i \geq 1$

3 Sample a candidate from a proposal distribution: $\mathbf{X}^* \sim q(\mathbf{X}(i-1), \cdot)$

4 With probability

$$1 \wedge \frac{\gamma(\mathbf{X}^*)}{\gamma(\mathbf{X}(i-1))} \frac{q(\mathbf{X}^*, \mathbf{X}(i-1))}{q(\mathbf{X}(i-1), \mathbf{X}^*)} \tag{2.1}$$

set $\mathbf{X}(i) = \mathbf{X}^*$, otherwise set $\mathbf{X}(i) = \mathbf{X}(i-1)$

The case where the proposal distribution is symmetric,

$$q(\mathbf{x}, \mathbf{y}) = q(\mathbf{y}, \mathbf{x}), \ \forall \mathbf{y}, \mathbf{x} \in E,$$

corresponds to the Metropolis algorithm, also known as *random walk Metropolis*. When the proposal is independent of the current state of the Markov chain we get the *independent MH* sampler. Many more variations and special cases exist. For a thorough survey of these methods see [68]. We may for example use multiple transition kernels and combine them through mixture or composition.

2.1.2 Gibbs Sampler

The Gibbs sampler is in fact a special case of the Metropolis-Hastings algorithm. If we consider multiple kernels, one for each block of variables, and use them in a composition, and additionally these kernels use the conditional distribution of the target as the proposal, then the acceptance rate is 1 and we obtain the Gibbs

sampler.

Algorithm 2.2: Gibbs Sampler

Initialise $\mathbf{X}(0)$
For iteration $i \geq 1$
  For $n = 1, \ldots, p$ **do**
    Sample $X_n(i) \sim \pi(\cdot|\mathbf{X}_{1:n-1}(i), \mathbf{X}_{n+1:p}(i-1))$

2.2 Sequential Monte Carlo

In this section, we briefly review the principle of SMC methods to sample from a given target distribution $\pi(\mathbf{x})$. As explained in the introduction the method requires one to introduce a sequence of bridging probability densities

$$\{\pi_n(\mathbf{x}_n), n = 1, \ldots, p\}$$

of increasing dimension such that $\pi_p(\mathbf{x}_p) = \pi(\mathbf{x})$; see Section 4 for detailed examples. For ease of presentation, we will assume that $\mathcal{X}_i = \mathcal{X}$ for any $i = 1, \ldots, p$, implying that $\pi_n(\mathbf{x}_n)$ is defined on the product space $\mathcal{X}^n$. Each density is assumed known up to a normalising constant, *i.e.* for $n = 1, \ldots, p$

$$\pi_n(\mathbf{x}_n) = Z_n^{-1}\gamma_n(\mathbf{x}_n) \ ,$$

where $\gamma_n : \mathcal{X}^n \to \mathbb{R}^+$ can be evaluated pointwise, but the normalising constant Z_n is unknown. We will use the notation Z for Z_p.

We describe below a generic SMC algorithm which encompasses numerous variants proposed in the literature. Section 2.2.1 below is important and sufficient to understand the description of the new algorithms presented in the paper. It is worth noting for readers already familiar with SMC methods that our description outlined below might appear slightly unorthodox in that it does not exactly match standard computer implementations. It however enjoys the same important theoretical and practical properties and has the advantage of greatly simplifying the mathematical developments required to show the validity of the algorithms presented throughout the paper. Section 2.2.2 focuses on such subtle issues, is only

required to understand the theoretical justifications of our algorithms, and might be skipped on a first reading.

2.2.1 Algorithm and Use of Its Output

An SMC algorithm requires one to specify an importance density $M_1(x_1)$ on $\mathcal{X}$ in order to initialise the recursion at time 1 and a family of transition kernels with associated densities $\{M_n(\mathbf{x}_{n-1}, x_n), n = 2, ..., p\}$ in order to extend $\mathbf{x}_{n-1} \in \mathcal{X}^{n-1}$ by sampling $x_n \in \mathcal{X}$ conditional upon $\mathbf{x}_{n-1}$ at time instants $n = 2, ..., p$. Guidelines on how to best select $\{M_n(\mathbf{x}_{n-1}, x_n)\}$ are well known, and the main recommendation is that an approximation of the conditional density $\pi_n(x_n|\mathbf{x}_{n-1})$ should be used [27], [61]. As mentioned earlier an SMC algorithm also involves a resampling procedure of the N particles, which relies on a family of probability distributions on $\{1, \ldots, N\}^N$, $\{r(\cdot|\mathbf{w}), \mathbf{w} \in [0,1]^N\}$. The resampling step is usually necessary as in most applications the variance of the importance weights would otherwise increase over time. This is an undesirable feature which rapidly manifests itself in practice in that the majority of particles have negligible weights.

The algorithm proceeds as shown in Algorithm 2.3 in order to produce a sequence of samples $\{\mathbf{X}_n^i, i = 1, ..., N\}$ for $n = 1, \ldots, p$. Note that in order to alleviate the notational burden we adopt below the convention that whenever the index i is used we mean "for all $i \in \{1, ..., N\}$." Further on, we also use the standard convention whereby capital letters are used for random variables while lower case letters are used for their values. We have used the notation $\mathbf{W}_n := \left(W_n^1, ..., W_n^N\right)$ and $\mathbf{A}_n := \left(A_n^1, ..., A_n^N\right)$. The variable A_{n-1}^i plays an important rôle in our formulation of SMC methods, and represents the index of the "parent" at time $n-1$ of particle $\mathbf{X}_n^i$ for $n = 2, \ldots, p$ (see Figure 2.1). The vector $\mathbf{A}_n$ is thus a random mapping defined on $\{1, \ldots, N\} \rightarrow \{1, \ldots, N\}^N$, and the standard resampling procedure is hence interpreted here as being the operation by which child particles at time n choose their parent particles at time $n-1$ according to a probability $r(\cdot|\mathbf{W}_{n-1})$ dependent on the parents' weights $\mathbf{W}_{n-1}$, or "fitness."

Let $O_n^i := \sum_{k=1}^N \mathbb{I}\left(A_n^k = i\right)$ be the number of offspring of particle i at time n. A desirable property of a resampling scheme is that it satisfies the following

Algorithm 2.3: Sequential Monte Carlo Algorithm

At $n = 1$

Sample $\mathbf{X}_1^i \sim M_1(\cdot)$

Update and normalise the weights

$$w_1\left(\mathbf{X}_1^i\right) := \frac{\gamma_1(\mathbf{X}_1^i)}{M_1(\mathbf{X}_1^i)},\ W_1^i = \frac{w_1\left(\mathbf{X}_1^i\right)}{\sum_{k=1}^N w_1\left(\mathbf{X}_1^k\right)}\ .$$

For $n = 2, ..., p$ **do**

Sample $\mathbf{A}_{n-1} \sim r\left(\cdot|\mathbf{W}_{n-1}\right)$

Sample $X_n^i \sim M_n(\mathbf{X}_{n-1}^{A_{n-1}^i}, \cdot)$ and set $\mathbf{X}_n^i = (\mathbf{X}_{n-1}^{A_{n-1}^i}, X_n^i)$

Update and normalise the weights

$$w_n\left(\mathbf{X}_n^i\right) := \frac{\gamma_n\left(\mathbf{X}_n^i\right)}{\gamma_{n-1}\left(\mathbf{X}_{n-1}^{A_{n-1}^i}\right) M_n\left(\mathbf{X}_{n-1}^{A_{n-1}^i}, X_n^i\right)}\ , \tag{2.2}$$

$$W_n^i = \frac{w_n\left(\mathbf{X}_n^i\right)}{\sum_{k=1}^N w_n\left(\mathbf{X}_n^k\right)} \tag{2.3}$$

unbiasedness condition

$$\mathbb{E}\left[O_n^i|\mathbf{W}_n\right] = NW_n^i\ . \tag{2.4}$$

The unbiasedness condition (2.4) ensures that future computational effort is concentrated on the most promising particles, while guaranteeing that the SMC algorithm described earlier yields consistent approximations of the distributions

$$\left\{\pi_n\left(d\mathbf{x}_n\right),\ n = 1, \ldots, p\right\}$$

and of their normalising constants $\{Z_n,\ n = 1, \ldots, p\}$. In particular, conditional upon the sampled particles, an approximation of the target distribution $\pi\left(d\mathbf{x}\right)$ is given by

$$\hat{\pi}^N\left(d\mathbf{x}\right) := \sum_{i=1}^N W_p^i \delta_{\mathbf{X}_p^i}\left(d\mathbf{x}\right)\ , \tag{2.5}$$

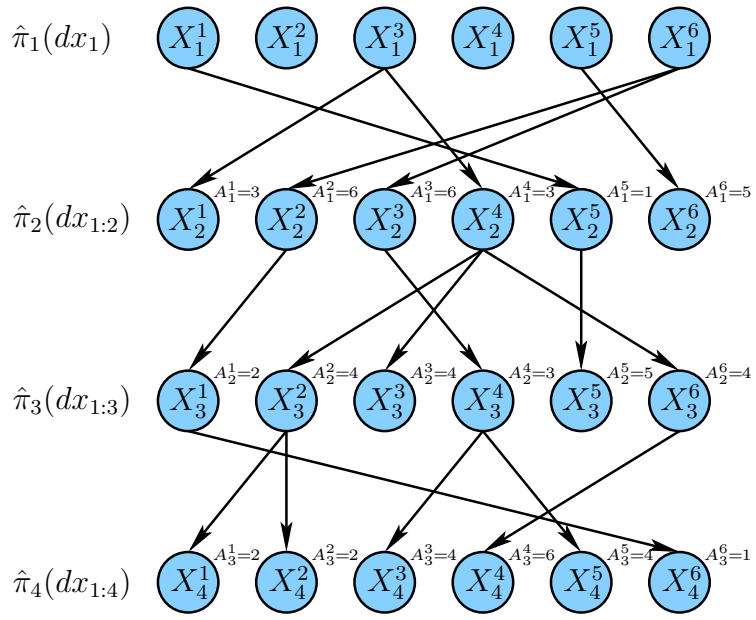


Figure 2.1: Illustration of particle ancestry in SMC.

from which expectations can be easily computed, but also an estimate of its normalising constant Z

$$\hat{Z}^N := \prod_{n=1}^{p} \left[\frac{1}{N} \sum_{i=1}^{N} w_n \left(\mathbf{X}_n^i \right) \right] . \tag{2.6}$$

The derivation for the estimate of the normalising constant is as follows:

$$Z := Z_p = Z_1 \prod_{n=2}^{p} \frac{Z_n}{Z_{n-1}} \tag{2.7}$$

where Z_1 and the ratio of normalising constants Z_n/Z_{n-1} are given by

$$Z_1 = \int \frac{\gamma_1(\mathbf{x}_1)}{M_1(\mathbf{x}_1)} M_1(\mathbf{x}_1)\, d\mathbf{x}_1 = \int w_n\left(\mathbf{x}_1\right) M_1(\mathbf{x}_1)\, d\mathbf{x}_1 \approx \frac{1}{N} \sum_{i=1}^{N} w_1\left(\mathbf{X}_1^i\right)$$

$$\begin{aligned}\frac{Z_n}{Z_{n-1}} &= \frac{1}{Z_{n-1}} \int \gamma_n(\mathbf{x}_n)\, d\mathbf{x}_n = \int \gamma_n(\mathbf{x}_n) \frac{\pi_{n-1}(\mathbf{x}_{n-1})}{\gamma_{n-1}(\mathbf{x}_{n-1})}\, d\mathbf{x}_n \\ &= \int \frac{\gamma_n(\mathbf{x}_n)}{\gamma_{n-1}(\mathbf{x}_{n-1}) M_n(\mathbf{x}_{n-1}, x_n)}\, \pi_{n-1}(\mathbf{x}_{n-1}) M_n(\mathbf{x}_{n-1}, x_n)\, d\mathbf{x}_n \\ &= \int w_n\left(\mathbf{x}_n\right)\, \pi_{n-1}(\mathbf{x}_{n-1}) M_n(\mathbf{x}_{n-1}, x_n)\, d\mathbf{x}_n \approx \frac{1}{N} \sum_{i=1}^{N} w_n\left(\mathbf{X}_n^i\right)\end{aligned}$$

Now substituting the Monte Carlo estimates of these integrals into Eqn. 2.7 yields the estimate of the normalising constant given in Eqn. 2.6.

2.2.2 Implementation Issues

In fact in practice, for computational efficiency, $\mathbf{O}_n$ is drawn first (*i.e.* without explicit reference to $\mathbf{A}_n$) according to a probability distribution $s(\cdot|\mathbf{W}_n)$ such that (2.4) holds and the offspring then matched to their parents. For example, the simplest unbiased resampling algorithm consists of sampling $\mathbf{O}_n$ according to a multinomial distribution of parameter $(N, \mathbf{W}_n)$. More sophisticated schemes such as residual resampling [61], stratified resampling [57] and minimum entropy resampling [23] also satisfy (2.4). Once $\mathbf{O}_n$ has been sampled, this is followed by a deterministic allocation procedure of the child particles to the parents, which defines "computer" indices *e.g.* the O_n^1 first child particles are associated to the parent particle number 1, *i.e.* $A_n^1 = 1, ..., A_n^{O_n^1} = 1$, likewise for the O_n^2 following child particles and the parent particle number 2, *i.e.* $A_n^{O_n^1+1} = 2, ..., A_n^{O_n^1+O_n^2} = 2$ *etc.* Further on, we will impose the slightly stronger unbiasedness condition.

(**A1**) For any $i = 1, \ldots, N$ and $n = 1, \ldots, p$ the resampling scheme satisfies

$$\mathbb{E}\left[O_n^i | \mathbf{W}_n\right] = N W_n^i$$

and

$$r\left(A_n^i = k | \mathbf{W}_n\right) = W_n^k\,. \tag{2.8}$$

Note that even if (2.4) holds then (2.8) is not necessarily satisfied, for example by the standard deterministic allocation procedure, but this property can be easily enforced by the addition of a random permutation of these "computer" indices. As

we shall see our indexing system makes the writing of the probability distributions underpinning our algorithms extremely simple.

2.2.3 Discussion of Convergence Results

The theoretical properties of SMC are now well understood, and we restrict ourselves to some of the simplest assumptions required to ensure the validity of our algorithms; see [64] for a full treatment and more sophisticated assumptions. The following notation will be needed

$$\mathcal{S}_n = \{\mathbf{x}_n \in \mathcal{X}^n : \pi_n(\mathbf{x}_n) > 0\} ,$$
$$\mathcal{Q}_1 = \{\mathbf{x}_1 \in \mathcal{X} : M_1(\mathbf{x}_1) > 0\} ,$$
$$\mathcal{Q}_n = \{\mathbf{x}_n \in \mathcal{X}^n : \pi_{n-1}(\mathbf{x}_{n-1}) M_n(\mathbf{x}_{n-1}, x_n) > 0\} \text{ for } n \geq 2.$$

(A2) For $n = 1, ..., p$, we have $\mathcal{S}_n \subseteq \mathcal{Q}_n$.

(A3) For any $n = 1, ..., p$, there exists $\{B_n < \infty\}$ such that for any $\mathbf{x}_n \in \mathcal{S}_n$

$$w_n(\mathbf{x}_n) \leq B_n . \tag{2.9}$$

(A4) There exist $\mu(\cdot)$ a probability density on $\mathcal{X}$ and $0 < \underline{w}, \overline{w}, \underline{\varepsilon}, \overline{\varepsilon} < \infty$ such that for any $n = 1, ..., p$ and any $\mathbf{x}_n \in \mathcal{S}_n$,

$$\underline{w} \leq w_n(\mathbf{x}_n) \leq \overline{w}$$

and

$$\underline{\varepsilon}\, \mu(x_n) \leq M_n(\mathbf{x}_{n-1}, x_n) \leq \overline{\varepsilon}\, \mu(x_n) .$$

Under assumption (A3) it is possible to establish results such as $\mathbb{L}^p$ bounds, show that $\hat{\pi}^N(d\mathbf{x})$ converges (weakly) almost surely (a.s.) as $N \to \infty$ towards $\pi(d\mathbf{x})$ and that $\hat{Z}^N$ converges a.s. to Z. Under additional mixing assumptions on the Feynman-Kac semi-group associated to $\{\pi_n\}$ and $\{M_n\}$, stronger results can be established. In particular, it can be shown using a proof similar to [18, Theorem 5], [64, Section 7.4.3] that for multinomial and residual resampling $\hat{Z}^N$ satisfies a central limit theorem and that, under (A4), there exists a finite constant

C, depending on $\underline{w}, \overline{w}, \underline{\varepsilon}, \overline{\varepsilon}, \mu$ and the dimension of $\mathcal{X}$ but not p, such that the variance $\text{var}(N)$ of $\hat{Z}^N/Z$ satisfies for any $N \geq 1$

$$\text{var}(N) \leq \frac{C\,p}{N} \tag{2.10}$$

The so-called degeneracy phenomenon for SMC algorithms is well documented, and manifests itself in practice by the fact that the approximation $\hat{\pi}^N(d\mathbf{x}_n)$ of the marginal $\pi(d\mathbf{x}_n)$ of $\pi(d\mathbf{x})$ is concentrated on a very limited number of particles whenever $p-n$ is large for a given number N of particles; this is the consequence of the successive resampling steps between times n and p. As a result the approximation of $\hat{\pi}^N(d\mathbf{x}_n)$ is poor and cannot be used for inferential purposes. This however does not necessarily mean that the distribution of the surviving path(s) is not "close" to $\pi(d\mathbf{x}_n)$. Under (A4), it can in fact be established [64, Section 8.3] that there exists a finite constant D, depending on $\underline{w}, \overline{w}, \underline{\varepsilon}, \overline{\varepsilon}$ but not p, such that the *unconditional* distribution of a sample from $\hat{\pi}^N(d\mathbf{x})$, denoted $q^N(d\mathbf{x})$, satisfies

$$\left\| q^N(\cdot) - \pi(\cdot) \right\| \leq \frac{D\,p}{N}\,, \tag{2.11}$$

where $\|\cdot\|$ is the total variation norm. Note that the constants C and D above typically increase with the dimensionality of $\mathcal{X}$ and decrease as the ergodic properties of the Feynman-Kac semi-group improve.

These results suggest that, under sufficient mixing conditions, the performance of SMC degrades "gracefully" linearly and not exponentially with p as is often the case with other sampling strategies. Note however, that (A4) is a very strong assumption not satisfied in most realistic scenarios and that it is difficult in practice to obtain useful quantitative bounds on C and D that could guide the choice of N. This generally results in a bias of unknown magnitude. Nevertheless, empirical evidence from numerous researchers suggests that SMC can be designed to produce random samples whose distribution is "close" to π even for large p in realistic scenarios where (A4) is not satisfied. This motivates using samples provided by an SMC algorithm as a proposal to feed a MH based algorithm in order to correct for the fact that $q^N(d\mathbf{x}) \neq \pi(d\mathbf{x})$. As we shall see in the next section, this is not straightforward.

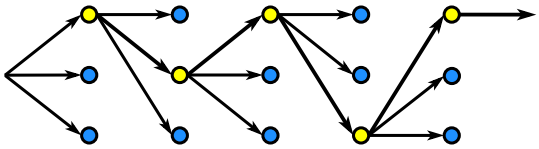


Figure 2.2: Construction of the proposal in CBMC. For each component several candidates are sampled, of which one is selected with probability proportional to it's importance weight.

2.3 Configurational Bias Monte Carlo

We now briefly introduce the configurational bias Monte Carlo (CBMC) algorithm, which is one of the methods used to tackle problems that we are interested in, as mentioned in the introduction, and we will use it to compare against the performance of PMCMC. CBMC is based on the scheme introduced by Rosenbluth and Rosenbluth in 1955 [70] and was developed by J. Siepmann to sample the conformational space of linear chain molecules [73]. In this section we will briefly introduce the basic CBMC algorithm. We will later compare the performance of our new method with this algorithm.

CBMC is a Markov chain Monte Carlo algorithm where the proposal is built up sequentially. In the case of a long molecule – or high dimensional state space – only a subset of the variables may be updated at an iteration. As in SMC, we need to define a sequence of distributions $\pi_n(\cdot)$ $(n = 1, \ldots, p)$ of increasing dimension. In the case of sampling chain molecules, this sequence is naturally given by the Boltzmann distributions obtained when growing the chain.

The proposal configuration is built as follows. For each variable (element in the chain) a population of candidate values are sampled, of which one is selected with probability proportional to its importance weight. This yields a candidate chain $\mathbf{X}^*$ with the importance weight $\hat{Z}^{N,*}$ given in Eqn. 2.12, as illustrated in Figure 2.2. To compute the importance weight $\hat{Z}^N$ for the current configuration, this procedure is repeated while ensuring that one of the candidates matches the current configuration $\mathbf{X}$. The candidate is then accepted with probability $1 \wedge \hat{Z}^{N,*}/\hat{Z}^N$. The details are given in Algorithm 2.4.

This algorithm is quite greedy (as illustrated in Figure 2.2), and hence only

works for moderately large chains. Several other extensions to the Rosenbluth method have been proposed [40], such as PERM [51], DPERM [19], and recoil-growth [20][40, Sec. 13.7]. PERM, like the Rosenbluth method, is a static Monte Carlo method, and addresses the weight degeneracy problem of the Rosenbluth method by pruning conformations with weights below some preset threshold and replicating ones with weights above some threshold (adjusting the weights accordingly). DPERM is the *dynamic* (Markov chain) generalisation of PERM. However, the performance of (D)PERM is quite sensitive to the thresholds [19], and unlike SMC, the particles (conformations) are not interacting. The recoil-growth (RG) method uses a multi-step look-ahead in an effort to overcome the dead-alley problem of CBMC, which potentially spends much of the simulation time exploring dead ends or evaluating importance weights of low probability candidates. Instead of generating N candidates at each step, RG proposes something similar to a "depth-first" search (of finite depth) to find "open" and "closed" trial directions, where "open" denotes directions that are deemed to have non-zero probability of generating a complete chain (at least for the finite depth look ahead), and "closed" indicates that the trial direction will result in a dead-alley, e.g. as in sampling self-avoiding walks on a grid.

2.4 Inference in State-Space Models

We now take a look at how MCMC and SMC are applied to state-space models (SSMs), and motivate the need for PMCMC. SSM is a very popular class of models and has broad application in many disciplines; for a thorough discussion of SSM see [13, 29, 41]. Some examples include target tracking models [50], change-point models [36], population dynamics models [12], stochastic volatility models (Sec. 4.5), partially observed diffusions [37], and models appearing in systems biology (Sec. 4.4).

2.4.1 Model Description

We consider the following SSM, also known as a hidden Markov model (HMM), where $\{X_n\}_{n\geq 1}$ is an unobserved Markov process with initial density $X_1 \sim \mu_\theta(\cdot)$

Algorithm 2.4: Configurational Bias Monte Carlo

Initialise $\mathbf{X}(0)$ (arbitrarily)

For iteration $s \geq 1$

Sample proposal $\mathbf{X}^*$ *and compute importance weight:*

At $n = 1$

For $k = 1, \ldots, N$, sample $X_1^{k,*} \sim M_1(\cdot)$

Compute the weights: $w_1\left(\mathbf{X}_1^{k,*}\right) := \gamma_1\left(\mathbf{X}_1^{k,*}\right)/M_1\left(X_1^{k,*}\right)$, and normalise $W_1^{k,*} = w_1\left(\mathbf{X}_1^{k,*}\right) / \sum_{k=1}^N w_1\left(\mathbf{X}_1^k\right)$

Sample $\mathbf{X}_1^* \sim \hat{\pi}_1^N(d\mathbf{x}) := \sum_{k=1}^N W_1^{k,*} \delta_{\mathbf{X}_1^{k,*}}(d\mathbf{x})$

For $n = 2, \ldots, p$ **do**

For $k = 1, \ldots, N$, sample $X_n^{k,*} \sim M_n(\mathbf{X}_{n-1}^*, \cdot)$ and set $\mathbf{X}_n^{k,*} = (\mathbf{X}_{n-1}, X_n^{k,*})$

Compute the weights $w_n\left(\mathbf{X}_n^{k,*}\right) := \frac{\gamma_n\left(\mathbf{X}_n^{k,*}\right)}{\gamma_{n-1}\left(\mathbf{X}_{n-1}^{k,*}\right) M_n\left(\mathbf{X}_{n-1}, X_n^{k,*}\right)}$ and normalise $W_n^{k,*} = \frac{w_n\left(\mathbf{X}_n^{k,*}\right)}{\sum_{k=1}^N w_n\left(\mathbf{X}_n^{k,*}\right)}$

Sample $X_n^* \sim \hat{\pi}_n^N(d\mathbf{x}) := \sum_{k=1}^N W_n^{k,*} \delta_{\mathbf{X}_n^{k,*}}(d\mathbf{x})$ and set $\mathbf{X}_n^* = (\mathbf{X}_{n-1}^*, X_n^*)$

Compute importance weight:

$$\hat{Z}^{N,*} = \prod_{n=1}^p \left[\frac{1}{N} \sum_{k=1}^N w_n\left(\mathbf{X}_n^{k,*}\right)\right] \tag{2.12}$$

Sample auxiliary variables and compute importance weight:

At $n = 1$

Set $X_1^1 := X_1(s)$ For $k = 2, \ldots, N$, sample $X_1^k \sim M_1(\cdot)$

Compute the weights $w_1\left(\mathbf{X}_1^k\right)$ as above

For $n = 2, \ldots, p$ **do**

Set $X_n^1 := X_n(s)$ For $k = 2, \ldots, N$, sample $X_n^k \sim M_n(\mathbf{X}_{n-1}(s), \cdot)$ and set $\mathbf{X}_n^k = (\mathbf{X}_{n-1}, X_n^k)$

Compute the weights $w_n\left(\mathbf{X}_n^k\right)$ as above

Compute importance weight: $\hat{Z}^N = \prod_{n=1}^p \left[\frac{1}{N} \sum_{k=1}^N w_n\left(\mathbf{X}_n^k\right)\right]$

With probability $1 \wedge \left(\hat{Z}^{N,*}/\hat{Z}^N(i-1)\right)$ set $\mathbf{X}(i) = \mathbf{X}^*$, otherwise set $\mathbf{X}(i) = \mathbf{X}(i-1)$

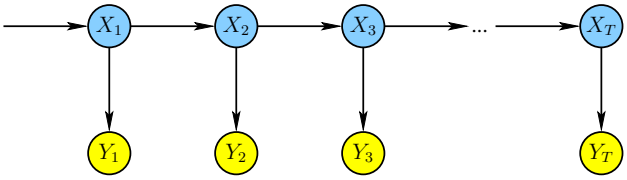


Figure 2.3: State-space model, also known as a hidden Markov model (HMM). The Markov process $\{X_n\}_{n\geq 1}$ is not observed directly but through a conditionally independent observation process $\{Y_n\}_{n\geq 1}$.

and transition probability density (with $n > 1$)

$$X_n | \, X_{n-1} \sim f_\theta \left(\cdot | \, X_{n-1} \right)$$

where θ is a static parameter of the model. The process $\{X_n\}_{n\geq 1}$ cannot be observed directly, but only through observations $\{Y_n\}_{n\geq 1}$, which are assumed independent conditional upon $\{X_n\}_{n\geq 1}$ with

$$Y_n | \, X_n \sim g_\theta \left(\cdot | \, X_n \right) \ .$$

The parameter θ may also be unknown, in which case it follows the prior distribution $p\left(\theta\right)$. The goal is now to perform Bayesian inference in this context.
First consider the case where the static model parameter θ is known. Given some observations $y_{1:T}$ inference relies on the following posterior density

$$p_\theta \left(x_{1:T} | \, y_{1:T} \right) \propto \mu_\theta \left(x_1 \right) g_\theta \left(y_1 | \, x_1 \right) \prod_{n=2}^{T} f_\theta \left(x_n | \, x_{n-1} \right) g_\theta \left(y_n | \, x_n \right) . \tag{2.13}$$

If we do not know θ, we assign a prior density $p(\theta)$ to θ and perform Bayesian inference using the joint posterior

$$p \left(\theta, \, x_{1:T} | \, y_{1:T} \right) \propto p(\theta) \, p_\theta \left(x_{1:T} | \, y_{1:T} \right) \tag{2.14}$$

When the model is non-linear non-Gaussian, the posterior densities $p_\theta \left(x_{1:T} | \, y_{1:T} \right)$ and $p \left(\theta, \, x_{1:T} | \, y_{1:T} \right)$ do not admit standard forms. This makes inference difficult and we need to resort to approximations. Here we consider Monte Carlo methods,

which have shown to be a flexible and efficient tool for inference in these types of models. As SSMs are ubiquitous in many areas of science there are literally thousands of papers published on this in the past decade alone. Instead of a thorough review, we instead highlight the underlying principles of applying MCMC and SMC in this context, as well as point out some of their limitations, which will motivate the use of PMCMC.

2.4.2 Markov Chain Monte Carlo for State-Space Models

In order to perform inference in SSM using Monte Carlo (MC) we need to be able to generate samples from the posteriors $p_\theta\left(x_{1:T}|\, y_{1:T}\right)$ or $p\left(\theta,\, x_{1:T}|\, y_{1:T}\right)$ in the case of fixed (known) or unknown parameter θ, respectively.

It is generally impossible to sample exactly from $p_\theta\left(x_{1:T}|\, y_{1:T}\right)$, except for linear Gaussian models and finite state-space HMMs. As outlined in Section 2.1 the Metropolis Hastings (MH) sampler makes use of a proposal distribution $q(x_{1:T}, x^*_{1:T})$ to generate candidates $x^*_{1:T}$, which are then accepted or rejected with some probability. The high dimensionality of $x_{1:T}$ makes it impossible to design good proposal distributions to update all states jointly, and a popular strategy consists of updating only a subset of components at a time. For example we can divide the T components into adjacent blocks of length K and iteratively update all the blocks, conditioned on the components outside the current block and the observations. A block $x_{n:(n+K-1)}$ is then updated according to an MCMC step of invariant density

$$p_\theta\left(x_{n:(n+K-1)}|y_{1:T}, x_{1:(n-1)}, x_{(n+K+1):T}\right) \propto \prod_{k=n}^{n+K} f_\theta\left(x_k|x_{k-1}\right) \prod_{k=n}^{n+K-1} g_\theta\left(y_k|x_k\right) \tag{2.15}$$

As long as K is not too large, it may be possible to design good proposals to be used in a MH update. For example if the transition density f_θ is linear Gaussian and the observation density g_θ is log-concave, then we may design a proposal density based on a Gaussian approximation of Eqn. 2.15, as illustrated in [71]. However, as K increases a Gaussian approximation may become insufficient. Even seemingly benign differences between the proposal and target distribution in low dimension can accumulate in higher dimensions and result in the acceptance rate dropping to zero, which limits the number of components K that can be updated simultane-

ously. This is a serious limitation, as it can severely constrain and slow down the exploration of the support of $p_\theta\left(x_{1:T}|y_{1:T}\right)$ and introduce dependence between the samples produced by the MCMC algorithm.

If the parameter is unknown we need to be able to sample from the joint density $p\left(\theta,\, x_{1:T}|\, y_{1:T}\right)$. A common approach is to alternately update the state components $x_{1:T}$ conditional on θ and the parameter θ conditional upon $x_{1:T}$. Sampling exactly from $p\left(\theta|\, y_{1:T}, x_{1:T}\right)$ can often be performed efficiently due to its small or moderate size. However, when the parameter θ and states $x_{1:T}$ are strongly correlated, this approach can result in poor mixing and thus yield an inefficient algorithm. In this case we would like to be able to update the parameter and states jointly. However, this would require sampling all the states jointly, which in general we cannot do as outlined above.

2.4.3 Sequential Monte Carlo for State-Space Models

The SMC algorithm decomposes the problem of sampling from $p_\theta\left(x_{1:T}|\, y_{1:T}\right)$ into a series of simpler sub-problems, as illustrated in Section 2.2, where we define the sequence of distributions

$$\left\{\pi_n(\mathbf{x}_n) = p_\theta\left(x_{1:n}|\, y_{1:n}\right), n = 1\ldots, p = T\right\}\,,$$

as well as transition densities

$$\begin{aligned} M_1(x_1) &= q_\theta\left(x_1|\, y_n\right) \quad \text{and} \\ M_n(\mathbf{x}_{n-1}, x_n) &= q_\theta\left(x_n|\, x_{n-1}, y_n\right), \quad \text{for } n = 2\ldots, p = T\,. \end{aligned}$$

The incremental particle weights (Eqn. 2.2) in this case are:

$$w_1\left(\mathbf{x}_1\right) = \frac{\mu_\theta\left(x_1\right)\, g_\theta\left(y_1|\, x_1\right)}{q_\theta\left(x_1|\, y_1\right)} \quad \text{and} \quad w_n\left(\mathbf{x}_n\right) = \frac{f_\theta\left(x_n|\, x_{n-1}\right)\, g_\theta\left(y_n|\, x_n\right)}{q_\theta\left(x_n|\, x_{n-1}, y_n\right)}$$

This allows us to efficiently generate a population of samples (particles) from $p_\theta\left(x_{1:T}|\, y_{1:T}\right)$. However, SMC also suffers from well-known drawbacks. If T is too large, then the SMC approximation of the joint density deteriorates as components at earlier times are not rejuvenated at subsequent time steps. This is due to

the resampling which diminishes the diversity of the samples at earlier times. As a result the approximation of the marginals $p_\theta(x_n|y_{1:T})$ is likely very poor when $T - n$ is large. This is also the reason why it is very difficult, although possible [2, 31, 43], perform inference on static parameters using SMC, i.e. to sample from the joint distribution $p(\theta, x_{1:T}|\, y_{1:T})$.

This naturally suggest to combine MCMC with SMC and leverage the strengths of each by using SMC to build an approximation of the joint posterior distribution which can then be used to generate proposals to be used in MCMC. In the next Chapter we will present the framework to do this and also perform parameter estimation, i.e. to sample from the joint posterior. Chapter 4 demonstrates this novel methodology on several applications, including SSMs.

Chapter 3

Particle Markov Chain Monte Carlo

As discussed previously, a simple independent MH update which leaves $\pi(\mathbf{x})$ invariant requires a proposal density $q(\mathbf{x})$. In order to sample a realisation $\{\mathbf{X}(i)\}$ of the associated Markov chain, the MH update proceeds as follows at iteration i: (a) sample $\mathbf{X}^* \sim q(\cdot)$, (b) set $\mathbf{X}(i) = \mathbf{X}^*$ with probability

$$1 \wedge \frac{\pi(\mathbf{X}^*)}{\pi(\mathbf{X}(i-1))} \frac{q(\mathbf{X}(i-1))}{q(\mathbf{X}^*)},$$

otherwise set $\mathbf{X}(i) = \mathbf{X}(i-1)$. We could suggest using as proposal density $q(\mathbf{x}) = q^N(\mathbf{x})$, *i.e.* the density of a particle generated by an SMC algorithm targeting $\pi(\mathbf{x})$. This is likely to result in an efficient independent MH algorithm following the discussion of the previous section. This would however require being able to evaluate $q^N(\mathbf{x})$ in order to compute the acceptance ratio above. This quantity is unfortunately not available in closed-form. Indeed, for $n = 1, ..., p$ let us denote $\overline{\mathbf{X}}_n := \left(X_n^1, ..., X_n^N\right) \in \mathcal{X}^N$ the set of N simulated $\mathcal{X}$-valued random variables at time n, then it is not difficult to establish that the joint density of all the variables

generated by the SMC algorithm described in Section 2.2 is

$$\begin{aligned}
&\psi\left(\bar{\mathbf{x}}_1, ..., \bar{\mathbf{x}}_p, \mathbf{a}_1, ..., \mathbf{a}_{p-1}\right) \\
&\quad = \psi\left(\bar{\mathbf{x}}_1\right) \prod_{n=2}^{p} \psi\left(\mathbf{a}_{n-1} | \mathbf{x}_{n-1}\right) \times \psi\left(\bar{\mathbf{x}}_n | \bar{\mathbf{x}}_{n-1}, \mathbf{a}_{n-1}\right) \qquad (3.1) \\
&\quad = \left(\prod_{i=1}^{N} M_1\left(x_1^i\right)\right) \prod_{n=2}^{p} \left(r\left(\mathbf{a}_{n-1} | \mathbf{w}_{n-1}\right) \prod_{i=1}^{N} M_n(\mathbf{x}_{n-1}^{a_{n-1}^i}, x_n^i)\right) , \qquad (3.2)
\end{aligned}$$

which is defined on $\mathcal{X}^{pN} \times \{1, \ldots, N\}^{(p-1)N}$. Now, we deduce that the marginal distribution of a particle drawn according to $\hat{\pi}^N(d\mathbf{x})$ given in (2.5) is, denoting $\mathbb{E}_\psi$ the expectation with respect to ψ,

$$q^N\left(d\mathbf{x}\right) = \mathbb{E}_\psi\left(\hat{\pi}^N(d\mathbf{x})\right) = \mathbb{E}_\psi\left(\sum_{i=1}^{N} W_p^i \delta_{\mathbf{X}_p^i}\left(d\mathbf{x}\right)\right) ,$$

which cannot be computed analytically in most situations of interest. In the next section we present a general methodology to circumvent this difficulty.

3.1 Particle Metropolis-Hastings Sampler

In order to illustrate the simplicity of the implementation of our approach we first describe a particular instance of the methodology in order to sample from $\pi(\mathbf{x})$, where $\mathbf{x}$ is updated in one single block. *The particle Metropolis Hastings (PMH) algorithm by itself does not offer any advantage over SMC, but would instead be used as a component of more complex MCMC algorithms.* For example in the protein folding application (Sec. 4.2) we use PMH in simulated annealing to perform block updates. We can show that this particular MCMC update is nothing but an independent MH sampler with an auxiliary target distribution defined on an extended space with the output of an SMC algorithm as a proposal. However, the principle underlying the construction of the auxiliary target distribution is not classical.

3.1.1 Algorithm

In order to sample from $\pi(\mathbf{x})$ the particle Metropolis Hastings (PMH) sampler proceeds as shown in Algorithm 3.1 (with the notation of Section 2.2, in particu-

Algorithm 3.1: Particle Metropolis Hastings Sampler

Initialisation $i = 0$

Run an SMC algorithm targeting $\pi(\mathbf{x})$

sample $\mathbf{X}(0) \sim \hat{\pi}^N(\cdot)$ and compute $\hat{Z}^N(0)$

For iteration $i \geq 1$

Run an SMC algorithm targeting $\pi(\mathbf{x})$, sample $\mathbf{X}^* \sim \hat{\pi}^N(\cdot)$ and compute $\hat{Z}^{N,*}$

With probability

$$1 \wedge \frac{\hat{Z}^{N,*}}{\hat{Z}^N(i-1)}, \tag{3.3}$$

set $\mathbf{X}(i) = \mathbf{X}^*$ and $\hat{Z}^N(i) = \hat{Z}^{N,*}$, otherwise set $\mathbf{X}(i) = \mathbf{X}(i-1)$ and $\hat{Z}^N(i) = \hat{Z}^N(i-1)$

lar (2.5) and (2.6)). Note that the acceptance probability (3.3) is independent of $\mathbf{X}^*$ which makes it possible to sample $\mathbf{X}^*$ only once the move is accepted. The acceptance probability (3.3) also enjoys the attractive property that, under (A3), it converges to 1 as $N \to \infty$ as both $\hat{Z}^{N,*}$ and $\hat{Z}^N(i-1)$ are consistent estimates of Z, the unknown normalising constant of $\pi(d\mathbf{x})$. In addition, under mixing assumptions of the Feynman-Kac semi-group, the variance of this acceptance ratio can be shown to be proportional to p/N.

3.1.2 Proof of Validity

In this section we establish the validity of the PMH algorithm by showing that it is a standard independent MH update with specific target and proposal distributions defined on an extended space (Theorem 1). This results in straightforward convergence properties, see Theorem 2.

At first sight, one might think that the sequence $\{\mathbf{X}(i)\}$ generated by the PMH will have $\pi(d\mathbf{x})$ as the desired equilibrium distribution only when $N \to \infty$. We can show that this is in fact the case for all $N \geq 1$. The key to establish this result is to reformulate the PMH as a standard independent MH sampler defined on an extended state-space with a suitable invariant distribution.

In the following we will need the following notation related to particle ances-

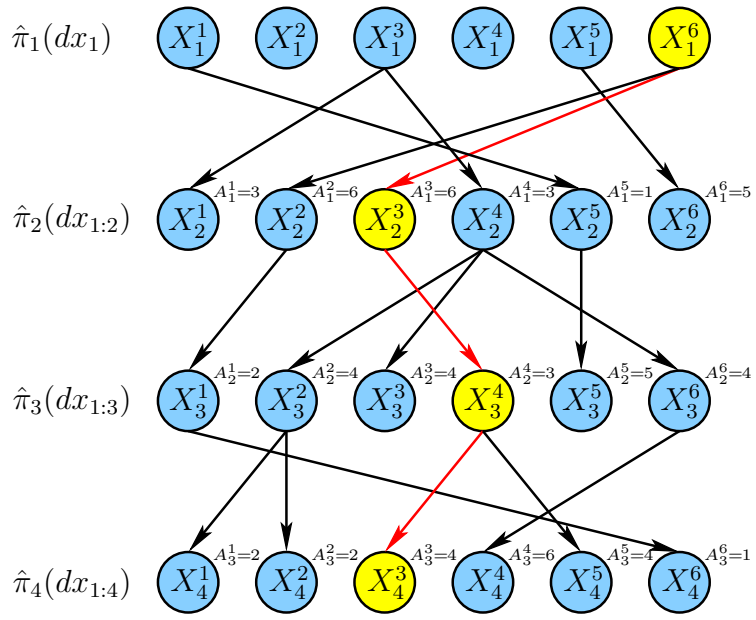


Figure 3.1: Illustration of running an SMC algorithm and selection of a path $\{X_1^6, X_2^3, X_3^4, X_4^3\}$ (yellow) to use as a proposal within PMCMC. The arrows show the resampling and resulting assignment of ancestry variables A_n^i.

tries. Assume that we have selected particle $\mathbf{X}_p^k$ at time p. For $k = 1, \ldots, N$ and $n = 1, \ldots, p$, let i_n^k denote the index of the ancestor particle of $\mathbf{x}_p^k$ at generation n. More formally we define $i_p^k := k$, $i_{p-1}^k := a_{p-1}^k$ (with the notation of Section 2.2) and for $n = p-1, \ldots, 1$ we have the backward recursive relation $i_n^k := a_n^{i_{n+1}^k}$. As a result we can rewrite $\mathbf{x}_p^k = (x_1^{i_1^k}, x_2^{i_2^k}, \ldots, x_{p-1}^{i_{p-1}^k}, x_p^{i_p^k})$. The matrix $\boldsymbol{a} = \{a_n^k\}$ is sampled row by row at the resampling steps of the SMC algorithm and allows us to represent the ancestries of all X_n^i (e.g. see arrows in Figure 3.1 which point from parent to offspring), while $\mathbf{i}^k$ lets us easily index the path of particle k. For example in Figure 3.1 the highlighted path is $i_4^3 = 3, i_3^3 = 4, i_2^3 = 3, i_1^3 = 6$.

Theorem 1 *Assume (A1)-(A2) then for any $N \geq 1$ the PMH sampler is an inde-*

pendent MH sampler defined on the extended space defined on

$$\mathcal{X}^{pN} \times \{1, \ldots, N\}^{(p-1)N+1},$$

with target density

$$\tilde{\pi}^N\left(k, \bar{\mathbf{x}}_1, ..., \bar{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}\right) = \tag{3.4}$$
$$\frac{1}{N^p} \frac{\pi\left(\mathbf{x}_p^k\right)}{M_1(x_1^{i_1^k}) \prod_{n=2}^{p} r(i_{n-1}^k|\mathbf{w}_{n-1}) M_n(\mathbf{x}_{n-1}^{i_{n-1}^k}, x_n^{i_n^k})} \psi\left(\bar{\mathbf{x}}_1, ..., \bar{\mathbf{x}}_p, \mathbf{a}_1, ..., \mathbf{a}_{p-1}\right),$$

with ψ defined in (3.1), and proposal density

$$q^N(k, \bar{\mathbf{x}}_1, ..., \bar{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}) := w_p^k \times \psi\left(\bar{\mathbf{x}}_1, ..., \bar{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}\right), \tag{3.5}$$

where w_p^k is a realization of the normalised importance weight defined in (2.3).

The proof is given below. Here $\mathbf{x}_p^k$ is distributed according to $\pi(d\mathbf{x})$. As a result the sequence $\{\mathbf{X}(i)\} = \{\mathbf{X}_p^{K(i)}(i)\}$ is of interest, because from standard MH theory it will leave $\pi(d\mathbf{x})$ invariant.

Proof of Theorem 1. We can easily check that (3.4) and (3.5) are both positive and sum to one; the factor $1/N^p$ corresponds to the uniform distribution on the set $\{1, ..., N\}^p$ for the random variables $K, A_1^{I_2^K}, \ldots, A_{p-1}^{I_p^K}$, i.e. there are N^p equivalent orderings of the indices in the particle path. Now the acceptance ratio of an independent MH algorithm is known to depend on the following "importance weight"

$$\frac{\tilde{\pi}^N\left(k, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}\right)}{q^N(k, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1})}$$
$$= \frac{1}{N^p} \frac{\pi\left(\mathbf{x}_p^k\right)}{w_p^k \times M_1(x_1^{i_1^k}) \prod_{n=2}^{p} r(i_{n-1}^k|\mathbf{w}_{n-1}) M_n(\mathbf{x}_{n-1}^{i_{n-1}^k}, x_n^{i_n^k})}$$

First we make use of (A1), i.e. $r(i_n^k|\mathbf{w}_n) = w_n^{i_n^k} = w_n(\mathbf{x}_n^{i_n^k}) / \sum_i^N w_n(\mathbf{x}_n^i)$, with

some abuse of notation. This yields:

$$\begin{aligned}\frac{\tilde{\pi}^N(\cdot)}{q^N(\cdot)} &= \frac{1}{N^p}\frac{\pi\left(\mathbf{x}_p^k\right)}{M_1(x_1^{i_1^k})\prod_{n=2}^{p} M_n(\mathbf{x}_{n-1}^{i_{n-1}^k}, x_n^{i_n^k})\prod_{n=1}^{p} w_n^{i_n^k}} \\ &= \frac{\frac{\gamma\left(\mathbf{x}_p^k\right)}{Z}\frac{1}{N^p}\prod_{n=1}^{p}\sum_{i=1}^{N} w_n\left(\mathbf{x}_n^i\right)}{M_1(x_1^{i_1^k})\prod_{n=2}^{p} M_n(\mathbf{x}_{n-1,n}^{i_{n-1}^k}, x_n^{i_n^k})\prod_{n=1}^{p} w_n(\mathbf{x}_n^{i_n^k})}\end{aligned}$$

In the above manipulations we have also substituted $\gamma\left(\mathbf{x}_p^k\right)/Z$ for $\pi\left(\mathbf{x}_p^k\right)$.

$$\begin{aligned}\frac{\tilde{\pi}^N(\cdot)}{q^N(\cdot)} &= \frac{\frac{\gamma\left(\mathbf{x}_p^k\right)}{Z}\frac{1}{N^p}\prod_{n=1}^{p}\sum_{i=1}^{N} w_n\left(\mathbf{x}_n^i\right)}{M_1(x_1^{i_1^k})\prod_{n=2}^{p} M_n(\mathbf{x}_{n-1,n}^{i_{n-1}^k}, x_n^{i_n^k})\frac{\gamma_1(\mathbf{x}_1^{i_1^k})}{M_1(\mathbf{x}_1^{i_1^k})}\prod_{n=2}^{p}\frac{\gamma_n(\mathbf{x}_n^{i_n^k})}{\gamma_{n-1}(\mathbf{x}_{n-1}^{i_{n-1}^k})M_n(\mathbf{x}_{n-1}^{i_{n-1}^k}, x_n^{i_n^k})}} \\ &= \frac{\hat{Z}^N}{Z}\end{aligned}$$

The final result is obtained thanks to the definitions of the incremental weights (2.2)

$$w_1\left(\mathbf{x}_1^i\right) := \frac{\gamma_1(\mathbf{x}_1^i)}{M_1(\mathbf{x}_1^i)} \quad \text{and} \quad w_n\left(\mathbf{x}_n^i\right) := \frac{\gamma_n\left(\mathbf{x}_n^i\right)}{\gamma_{n-1}\left(\mathbf{x}_{n-1}^{i_{n-1}^k}\right) M_n\left(\mathbf{x}_{n-1}^{i_{n-1}^k}, x_n^i\right)},$$

and of the normalising constant estimate (2.6)

$$\hat{Z}^N := \prod_{n=1}^{p}\left[\frac{1}{N}\sum_{i=1}^{N} w_n\left(\mathbf{x}_n^i\right)\right].$$

It should now be clear that the PMH algorithm described above corresponds to sampling particles according to q^N defined in (3.5) and that the acceptance probability (3.3) corresponds to that of an independent MH algorithm with target distribution $\tilde{\pi}^N$ given by (3.4). ■

As a result, Theorem 1 is powerful since by showing that our PMH algorithm is a MH algorithm in disguise, it allows us to use standard results concerning the

convergence properties of the independent MH algorithm to prove the following theorem. Let $\mathcal{L}^N(\mathbf{X}(i) \in \cdot)$ denote the distribution of $\mathbf{X}(i)$ generated by the PMH algorithm with $N \geq 1$ particles.

Theorem 2 *Assume (A1)-(A2) then for any $N \geq 1$ the PMH sampler generates a sequence of probability distributions $\left\{\mathcal{L}^N(\mathbf{X}(i) \in \cdot)\right\}$ such that*

$$\left\|\mathcal{L}^N(\mathbf{X}(i) \in \cdot) - \pi(\cdot)\right\| \to 0 \quad as \quad i \to \infty .$$

In addition, under (A3), there exists $\rho \in [0, 1)$ such that for any $i \geq 1$ and $N \geq 1$,

$$\left\|\mathcal{L}^N(\mathbf{X}(i) \in \cdot) - \pi(\cdot)\right\| \leq \rho^i .$$

The proof can be found in Appendix A. Note that the second result might appear negative since from the proof we do not observe an improvement on the rate of convergence of the algorithm as N increases. However, as a particular case of [1, Theorem 1] it is possible to show that for any $\epsilon, \eta > 0$ there exists N_0 such that for any $N \geq N_0$ and any $i \geq 1$

$$\left\|\mathcal{L}_*^N(\mathbf{X}(i) \in \cdot) - \pi(\cdot)\right\| \leq \epsilon$$

with $\psi-$probability larger than $1-\eta$, where $\mathcal{L}_*^N(\mathbf{X}(i) \in \cdot)$ denotes the conditional distribution of $\mathbf{X}(i)$ conditional upon the random variables generated at iteration 0 by the SMC algorithm.

3.1.3 Extensions and Variations

As pointed out earlier, Theorem 1 is fundamental since it highlights the standard MH nature of the PMH algorithm and allows us to easily establish its validity. In this section we show how this property can also suggest further improvements.

Using All the Particles

One criticism of PMH is that a lot of work seems wasted by only selecting one candidate from the SMC proposal, especially considering that the importance weight (estimate of the normalisation constant) is independent of the selected path. We

propose to use all the particles in order to carry out inference using a strategy which shares some characteristics with [39].

The 'standard' estimate of $\mathbb{E}_\pi(f)$ for L MCMC iterations is $\frac{1}{L}\sum_{i=1}^{L} f(\mathbf{X}(i))$. We will show in this thesis that the following estimator, which utilises all the particles, converges towards $\mathbb{E}_\pi(f)$ as $L \to \infty$ for any $N \geq 1$:

Theorem 3 *Assume (A1)-(A2) and $\mathbb{E}_\pi(|f|) < \infty$. Then for any $N \geq 1$,*

1. *the estimate*
$$\frac{1}{L}\sum_{i=1}^{L}\left(\sum_{k=1}^{N} W_p^k(i)\, f(X_{1:p}^k(i))\right) \tag{3.6}$$
converges almost surely towards $\mathbb{E}_\pi(f)$ as $L \to \infty$ where $\{W_p^k(i), X_{1:p}^k(i)\}$ corresponds to the set of normalised weights and particles used to compute $\hat{Z}(i)$,

2. *denoting $\{W_p^{*k}(i), X_{1:p}^{*k}(i), k = 1, \ldots, N\}$ the set of proposed particles at iteration i (*i.e. *before deciding whether or not to accept this population)*
$$\frac{1}{L}\sum_{i=1}^{L}\left(1 \wedge \frac{\hat{Z}^{N,*}(i)}{\hat{Z}^{N}(i-1)}\right)\left(\sum_{k=1}^{N} W_p^{*k}(i)\, f(X_{1:p}^{*k}(i))\right) \tag{3.7}$$
$$+\left(1 - 1 \wedge \frac{\hat{Z}^{N,*}(i)}{\hat{Z}^{N}(i-1)}\right)\left(\sum_{k=1}^{N} W_p^{k}(i-1)\, f(X_{1:p}^{k}(i-1))\right) \tag{3.8}$$
converges almost surely towards $\mathbb{E}_\pi(f)$ as $L \to \infty$.

The proof is given in Appendix A. The estimate (3.6) is an average of importance sampling estimates. Each of these estimates is biased but the MH mechanism allows us to obtain asymptotically consistent estimates by rejecting some of the populations of particles proposed. Following [39] is also possible to propose an estimate which recycles the candidate populations of particles rejected by the PMH.

Subblock Updates of π

We have focused on the case where all the components of $\mathbf{x} = (x_1, ..., x_p)$ are updated simultaneously. Again, if p is too large compared to the number N of par-

ticles, then the particle approximation of π produced by the SMC algorithm might be poor, resulting in an inefficient MH algorithm. In such situations we can suggest the use of mixtures/compositions of PMH transition probabilities that update subblocks of the form $\mathbf{x}_{a:b}$ for $1 \leq a < b \leq p$, effectively targeting conditional distributions of the type $\pi\left(x_{a:b}|\mathbf{x}_{-a:b}\right)$. The theoretical results of Section 2.2 suggest that under favourable conditions the number of particles required to approximate $\pi\left(x_{a:b}|\mathbf{x}_{-a:b}\right)$ for a given precision decreases linearly with the size $b-a+1$ of the subblock. For such a strategy there is a tradeoff between the accuracy of the proposal distribution for the conditional update, and the dependence structure on the complementary block which might result in long correlation time in the sequence $\{\mathbf{X}(i)\}$.

Partial Particle Updates

Although the structure of the target distribution $\tilde{\pi}^N$ naturally lends itself to full updates of all the particles using SMC algorithms we might consider rejuvenating some of the particles conditional upon the others for computational purposes. For example, given $\mathbf{x}_p^k$ and its "ancestral lineage" $(i_1^k, i_2^k, \ldots, i_p^k)$ which is such that $\mathbf{x}_p^k = (x_1^{i_1^k}, x_2^{i_2^k}, \ldots, x_{p-1}^{i_{p-1}^k}, x_p^{i_p^k})$ then we could propose to update the $N-1$ remaining particles and their "ancestral lineages" according to a Gibbs step under $\tilde{\pi}^N$; this "conditional SMC sampling" strategy is detailed in Section 3.2. Any classical MCMC updates that leave $\tilde{\pi}^N$ invariant can also obviously be used. Another possible suggestion in order to fight the aforementioned degeneracy for large p's is to use possibly standard MCMC transition probabilities in order to update subblocks $\mathbf{x}_{a:b}$ of $\mathbf{x}$ for $1 \leq a < b \leq p$ such that $\mathbf{x}_{a:b}$ corresponds to a region of $\mathbf{x}$ where particle depletion is severe.

3.2 Particle Gibbs Sampler

Assume now that we are interested in sampling from a distribution

$$\pi\left(\theta, \mathbf{x}\right) = \frac{\gamma\left(\theta, \mathbf{x}\right)}{Z} \tag{3.9}$$

where $\gamma : \Theta \times E \to \mathbb{R}^+$ is known pointwise. We show here again how SMC algorithms can be used as a component of an MCMC algorithm, namely a Gibbs sampler, to sample from distributions of the type $\pi(\theta, \mathbf{x})$ in situations where sampling from $\pi(\mathbf{x}|\theta)$ is difficult without resorting to SMC methods.

A standard strategy to sample from $\pi(\theta, \mathbf{x})$ consists of using the Gibbs sampler; that is sampling iteratively from the full conditionals $\pi(\theta|\mathbf{x})$ and $\pi(\mathbf{x}|\theta)$. In numerous situations it is possible to sample exactly from $\pi(\theta|\mathbf{x})$, and we will assume here that this is the case. Otherwise an MH update of invariant density $\pi(\theta|\mathbf{x})$ might be used. For models of practical interest $\mathbf{x}$ can be high dimensional (*e.g.* a vector of latent variables of the size of a large dataset) and the conditional distribution $\pi(\mathbf{x}|\theta)$ non-standard, precluding practical exact sampling. We have

$$\pi(\mathbf{x}|\theta) = \frac{\gamma(\theta, \mathbf{x})}{\gamma(\theta)}, \tag{3.10}$$

where $\gamma(\theta) := \int_E \gamma(\theta, \mathbf{x})\, d\mathbf{x}$ is assumed unknown.

It is therefore natural to suggest the use of an SMC algorithm in order to propose approximate samples from this conditional distribution. Hence we consider a family of bridging distributions $\{\pi_n(\mathbf{x}_n|\theta); n = 1, \ldots, p\}$ where

$$\pi_n(\mathbf{x}_n|\theta) = \frac{\gamma_n(\theta, \mathbf{x}_n)}{Z_n^\theta}, \tag{3.11}$$

such that $\pi_p(\mathbf{x}_p|\theta) = \pi(\mathbf{x}|\theta)$ and a family of transition probability densities $\{M_n^\theta(\mathbf{x}_{n-1}, x_n)\}$ that defines sampling of $x_n \in \mathcal{X}$ conditional upon $\mathbf{x}_{n-1} \in \mathcal{X}^{n-1}$; note that $\gamma_p(\theta, \mathbf{x}_p) = \gamma(\theta, \mathbf{x})$ and $Z_p^\theta = \gamma(\theta)$.

3.2.1 Algorithm

The particle Gibbs (PG) sampler is an approximation of the "ideal" Gibbs sampler where we approximately sample from $\pi(\mathbf{x}|\theta)$ using an SMC algorithm. Contrary to the PMH algorithm, sampling from the approximation $\hat{\pi}^N(d\mathbf{x}|\theta)$ of $\pi(\mathbf{x}|\theta)$ using the PG algorithm requires us to keep track of the "ancestral lineage" $I := (I_1, I_2, \ldots, I_p)$ of the random variable $\mathbf{X}$ which, we point out again, is such that

$\mathbf{X} = (X_1^{I_1}, ..., X_p^{I_p})$. The PG sampler is shown in Algorithm 3.2.

Algorithm 3.2: Particle Gibbs Sampler

Initialisation $i = 0$
  Set randomly $\theta(0)$
  Run an SMC algorithm targeting $\pi(\mathbf{x}|\theta(0))$
  Sample $\mathbf{X}(0) \sim \hat{\pi}^N(\cdot|\theta(0))$ and denote $I(0)$ its ancestral lineage.
For iteration $i \geq 1$
  Sample $\theta(i) \sim \pi(\cdot|\mathbf{X}(i-1))$
  Run a conditional SMC algorithm for $\theta(i)$ consistent with $\mathbf{X}(i-1), I(i-1)$
  Sample $\mathbf{X}(i) \sim \hat{\pi}^N(\cdot|\theta(i))$ and denote $I(i)$ its ancestral lineage

The conditional SMC step is the non-standard step of the algorithm which, given $(\theta, I, \mathbf{X} = \mathbf{X}_p^{I_p})$, yields an SMC approximation of $\pi(\mathbf{x}|\theta)$ by using the "frozen" particle $\mathbf{X}$ and sampling $N-1$ "free" particles consistent with this particle and its ancestral lineage. This is shown in Algorithm 3.3, where we have used the notation $\mathbf{A}_{n-1}^{-I_n} = \mathbf{A}_{n-1} \backslash \{A_{n-1}^{I_n}\}$.

To sample from the discrete-valued conditional distribution $r(\mathbf{a}_{n-1}^{-I_n}|\mathbf{w}_{n-1}, a_{n-1}^{I_n})$, we can use the following procedure.

1. Sample the number of offspring $\mathbf{O}_{n-1} \sim s\left(\cdot|\,\mathbf{W}_{n-1}, O_{n-1}^{I_n} \geq 1\right)$.
2. Sample the indices of the $N-1$ "free" offspring uniformly on the set $\{1, ..., N\} \setminus \{I_n\}$.

When sampling from $s\left(\cdot|\,\mathbf{W}_{n-1}, O_{n-1}^{I_n} \geq 1\right)$ cannot be performed in closed-form, we can use a rejection sampling procedure by sampling $\mathbf{O}_{n-1} \sim s(\cdot|\,\mathbf{W}_{n-1})$ until $O_{n-1}^{I_n} \geq 1$. Note however that it is possible to sample directly from $s(\cdot|\mathbf{W}_{n-1}, O_{n-1}^{I_n} \geq 1)$ in important cases. We consider multinomial and stratified resampling below.

Remark 1 *Note that the conditional SMC algorithm does not correspond to sampling from a conditional distribution of the distribution of the SMC algorithm ψ^θ (given in (3.16)). However this becomes true as $N \to \infty$.*

Algorithm 3.3: Conditional Sequential Monte Carlo Algorithm

At $n = 1$

For $i \neq I_1$, sample $\mathbf{X}_1^i \sim M_1^\theta(\cdot)$

Update and normalise the weights

$$w_1\left(\mathbf{X}_1^i\right) = \frac{\gamma_1(\theta, \mathbf{X}_1^i)}{M_1^\theta(\mathbf{X}_1^i)},\ W_1^i = \frac{w_1\left(\mathbf{X}_1^i\right)}{\sum_{k=1}^N w_1\left(\mathbf{X}_1^k\right)}\ .$$

For $n = 2, ..., p$ **do**

Sample $\mathbf{A}_{n-1}^{-I_n} \sim r(\cdot|\mathbf{W}_{n-1}, A_{n-1}^{I_n})$

For $i \neq I_n$, sample $X_n^i \sim M_n^\theta(\mathbf{X}_{n-1}^{A_{n-1}^i}, \cdot)$ and set $\mathbf{X}_n^i = (\mathbf{X}_{n-1}^{A_{n-1}^i}, X_n^i)$

Update and normalise the weights

$$w_n\left(\mathbf{X}_n^i\right) = \frac{\gamma_n\left(\theta, \mathbf{X}_n^i\right)}{\gamma_{n-1}\left(\theta, \mathbf{X}_{n-1}^{A_{n-1}^i}\right) M_n^\theta\left(\mathbf{X}_{n-1}^{A_{n-1}^i}, X_n^i\right)}\ , \tag{3.12}$$

$$W_n^i = \frac{w_n\left(\mathbf{X}_n^i\right)}{\sum_{k=1}^N w_n\left(\mathbf{X}_n^k\right)}\ . \tag{3.13}$$

Multinomial Resampling

For multinomial resampling, we can factorise the probability of the number of offsprings as

$$\mathbb{P}(\mathbf{O}_{n-1}|\mathbf{W}_{n-1}, O_{n-1}^{I_n} \geq 1) = \mathbb{P}(O_{n-1}^{I_n}|\mathbf{W}_{n-1}, O_{n-1}^{I_n} \geq 1)\, \mathbb{P}(\mathbf{O}_{n-1}^{-I_n}|\mathbf{W}_{n-1}, O_{n-1}^{I_n})$$

We then have the following procedure to perform multinomial resampling:

Algorithm 3.4: Conditional Multinomial Resampling

Sample $O_{n-1}^{I_n} \sim \mathcal{B}^+\left(N, W_{n-1}^{I_n}\right)$

Compute $\overline{W}_{n-1}^i \propto W_{n-1}^i$ with $\sum_{i \neq I_n} \overline{W}_{n-1}^i = 1$ and set $\overline{\mathbf{W}}_{n-1} = \left(\overline{W}_{n-1}^1, \ldots, \overline{W}_{n-1}^{I_n - 1}, \overline{W}_{n-1}^{I_n + 1}, \ldots, \overline{W}_{n-1}^N\right)$

Sample $\mathbf{O}_{n-1}^{-I_n} \sim \mathcal{M}\left(N, \overline{\mathbf{W}}_{n-1}\right)$

where $\mathcal{B}^{+}\left(N, W_{n-1}^{I_n}\right)$ is the Binomial distribution of parameters $(N, W_{n-1}^{I_n})$ restricted to the support $\{1, 2, \ldots\}$.

Stratified Resampling

Algorithm 3.5: Conditional Stratified Resampling

Compute adjusted weights $W_{n-1}^{i,*}$ as given in Eqn.3.14
if $W_{n-1}^{I_n,*} \geq 1/N$ **then**
  sample $\mathbf{O}_{n-1} \sim s\left(\cdot \middle| \mathbf{W}_{n-1}^{*}\right)$
else
  sample U_1 uniformly on $\left(\sum_{k=1}^{I_n-1} W_{n-1}^{k,*}, \sum_{k=1}^{I_n} W_{n-1}^{k,*}\right)$
  compute $U_j = U_1 + \frac{j-1}{N}$ for $j \in \mathbb{Z}$ and set

$$O_{n-1}^{i} = \#\left\{U_j : \sum_{k=1}^{i-1} W_{n-1}^{k,*} \leq U_j \leq \sum_{k=1}^{i} W_{n-1}^{k,*}\right\}.$$

For the popular stratified resampling algorithm [57], we can use the method given in Algorithm 3.5. The condition of having at least one offspring of particle I_n changes the resample distribution and results in the following adjusted weights:

$$W_{n-1}^{I_n,*} = \frac{W_{n-1}^{I_n}}{1-(1-W_{n-1}^{I_n})^N}, \qquad W_{n-1}^{i\neq I_n,*} = W_{n-1}^{i}\left(\frac{1-W_{n-1}^{I_n,*}}{1-W_{n-1}^{I_n}}\right) \tag{3.14}$$

3.2.2 Proof of Validity

In this section we establish the validity of the PG algorithm under mild assumptions. The following notation will be needed, for any $\theta \in \Theta$, let

$$\mathcal{S}_n^{\theta} = \left\{\mathbf{x}_n \in \mathcal{X}^n : \pi_n\left(\mathbf{x}_n \middle| \theta\right) > 0\right\},$$
$$\mathcal{Q}_1^{\theta} = \left\{\mathbf{x}_1 \in \mathcal{X} : M_1^{\theta}\left(\mathbf{x}_1\right) > 0\right\},$$
$$\mathcal{Q}_n^{\theta} = \left\{\mathbf{x}_n \in \mathcal{X}^n : \pi_{n-1}\left(\mathbf{x}_{n-1} \middle| \theta\right) M_n^{\theta}\left(\mathbf{x}_{n-1}, x_n\right) > 0\right\} \text{ for } n \geq 2,$$

and

$$\mathcal{S} = \{\theta \in \Theta : \pi(\theta) > 0\} \ .$$

(**A5**) For any $\theta \in \mathcal{S}$, we have $\mathcal{S}_n^\theta \subseteq \mathcal{Q}_n^\theta$ for $n = 1, ..., p$.

(**A6**) The 'ideal' Gibbs (G hereafter) sampler defined by the conditionals $\pi(\theta|\mathbf{x})$ and $\pi(\mathbf{x}|\theta)$ is irreducible and aperiodic (and hence converges for π-almost all starting points).

We have the following result.

Theorem 4 *Assume (A1)-(A5), then for any $N \geq 2$ the PG sampler defines an MCMC kernel on the extended space $\Theta \times \mathcal{X}^{pN} \times \{1, \ldots, N\}^{(p-1)N+1}$ with target density*

$$\tilde{\pi}^N(\theta, k, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}) := \tag{3.15}$$
$$\frac{1}{N^p} \frac{\pi\left(\theta, \mathbf{x}_p^k\right)}{M_1^\theta(x_1^{i_1^k}) \prod_{n=2}^p r(i_{n-1}^k|\mathbf{w}_{n-1}) M_n^\theta(\mathbf{x}_{n-1}^{i_{n-1}^k}, x_n^{i_n^k})} \psi^\theta(\overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, ..., \mathbf{a}_{p-1}) ,$$

where

$$\psi^\theta(\overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, ..., \mathbf{a}_{p-1}) := \tag{3.16}$$
$$\left(\prod_{i=1}^N M_1^\theta(x_1^i)\right) \prod_{n=2}^p \left(r(\mathbf{a}_{n-1}|\mathbf{w}_{n-1}) \prod_{i=1}^N M_n^\theta\left(\mathbf{x}_{n-1}^{a_{n-1}^i}, x_n^i\right)\right) .$$

Additionally assume (A6) holds then for any $N \geq 2$ the PG sampler generates a sequence $\{\theta(i), \mathbf{X}(i)\}$ whose marginal distributions $\{\mathcal{L}^N((\theta(i), \mathbf{X}(i)) \in \cdot)\}$ satisfy

$$\left\|\mathcal{L}^N((\theta(i), \mathbf{X}(i)) \in \cdot) - \pi(\cdot)\right\| \to 0 \text{ as } i \to \infty .$$

The proof can be found in Appendix A.

3.2.3 Extensions

Alternative Moves

The PG algorithm is an appealing algorithm which has the property that it converges to the standard Gibbs sampler as $N \to \infty$. However, given p and for a finite N for which the degeneracy problem is severe, the fact that the PG sampler forces the "frozen" path $I, \mathbf{X} = \mathbf{X}_p^{I_p}$ to survive during the conditional PMH step can have detrimental effects and result in a highly dependent Markov chain. Indeed, in such situations most of the particles at time p generated by the conditional SMC algorithm coalesce with the ancestral lineage of the "frozen" path $I, \mathbf{X} = \mathbf{X}_p^{I_p}$. To limit this problem, various alternatives to update of $\mathbf{X}(i)$ proposed above can be suggested. For example, we can suggest to update $\mathbf{X}(i)$ using the PMH algorithm, which yields Algorithm 3.6.

Algorithm 3.6: Alternate Move for Particle Gibbs Sampler

For iteration $i \geq 1$

- Sample $\theta(i) \sim \pi(\cdot|\mathbf{X}(i-1))$
- Run a conditional SMC algorithm for $\theta(i)$ consistent with $\mathbf{X}(i-1), I(i-1)$ and set $\widehat{\gamma}^N(\theta(i)) = \widehat{Z}^N$
- Run an SMC algorithm targeting $\pi(\mathbf{x}|\theta(i))$, sample $\mathbf{X}^* \sim \hat{\pi}^N(\cdot|\theta(i))$ and set $\widehat{\gamma}^{N,*}(\theta(i)) = \widehat{Z}^N$
- With probability

$$1 \wedge \frac{\widehat{\gamma}^{N,*}(\theta(i))}{\widehat{\gamma}^N(\theta(i))}$$

set $\mathbf{X}(i) = \mathbf{X}^*$ and $I(i) = I^*$, otherwise set $\mathbf{X}(i) = \mathbf{X}(i-1)$ and $I(i) = I(i-1)$

Following the developments concerning the PG sampler, we can easily show that under mild assumptions (A1-A5) and for any $N \geq 1$ this algorithm has the desired invariant distribution.

Optimisation

In some situations we might be interested in maximising $\pi(\theta)$. As $\pi(\theta)$ is typically unknown, even up to a normalising constant, it is not possible to use an algorithm such as the simulated annealing algorithm. However, we can introduce the following distribution, defined on $\Theta \times E^m$

$$\pi_m\left(\theta, \mathbf{x}^1, ..., \mathbf{x}^m\right) \propto \prod_{i=1}^m \pi\left(\theta, \mathbf{x}^i\right) \tag{3.17}$$

whose marginal distribution $\pi_m(\theta)$ is proportional to $[\pi(\theta)]^m$ [28]. This distribution concentrates itself on the set of global maxima of $\pi(\theta)$ as m increases. It is straightforward to apply a modified version of the PG sampler to sample from $\pi_m\left(\theta, \mathbf{x}^1, ..., \mathbf{x}^m\right)$ by sampling m independent conditional SMC algorithms. At iteration i, it proceeds as follows:

Algorithm 3.7: Marginal Maximisation using Particle Gibbs

1 Sample $\theta(i) \sim \pi_m\left(\cdot|\mathbf{X}^1(i-1), ..., \mathbf{X}^m(i-1)\right)$
2 **For** $k = 1, ..., m$
3 Run a conditional SMC algorithm for $\theta(i)$ consistent with $\mathbf{X}^k(i-1), I^k(i-1)$
4 Sample $\mathbf{X}^k(i) \sim \hat{\pi}^N(\cdot|\theta(i))$ and denote $I^k(i)$ its ancestral lineage

It can be shown under mild assumptions (A1-A5) and for any $N \geq 2$ that the Markov kernel associated to this algorithm admits $\pi_m\left(\theta, \mathbf{x}^1, ..., \mathbf{x}^m\right)$ as invariant distribution.

An alternative consists of considering the following extended distribution

$$\begin{aligned}
&\tilde{\pi}^N\left(\theta, k_1, ..., k_m, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}\right) \\
&\quad \propto \prod_{i=1}^m \pi\left(\theta, \mathbf{x}_p^{k_i}\right) \prod_{i=1, i\neq i_1^{k_1:k_m}}^N M_1^\theta\left(x_1^i\right) \\
&\qquad \times \prod_{n=2}^p \left(r\left(\mathbf{a}_{n-1}^{-i_1^{k_1:k_m}} \middle| \mathbf{w}_{n-1}, a_{n-1}^{i_n^{k_1:k_m}}\right) \prod_{i=1, i\neq i_n^{k_1:k_m}}^N M_n^\theta\left(\mathbf{x}_{n-1}^{a_{n-1}^i}, x_n^i\right) \right)
\end{aligned} \tag{3.18}$$

where $k_a \neq k_b$ for $a \neq b$ and $i_n^{k_1:k_m} = \left(i_n^{k_1}, ..., i_n^{k_m}\right)$. Note that even if $k_a \neq k_b$, these paths might have a common ancestor; that is there exists n such that $i_n^{k_a} = i_n^{k_b}$. To sample from (3.18) and thus from $\pi_m\left(\theta, \mathbf{x}^1, ..., \mathbf{x}^m\right)$, we could propose the following iterative algorithm. At each iteration, sample $\theta \sim \pi_m\left(\cdot | \, \mathbf{x}_p^{k_1}, ..., \mathbf{x}_p^{k_m}\right)$ then run a conditional SMC algorithm for θ consistent with $\mathbf{x}_p^{k_1}, ..., \mathbf{x}_p^{k_m}$ and their ancestral lineages. Finally, sample $(k_1, ..., k_m)$ from

$$\tilde{\pi}^N\left(k_1, ..., k_m | \, \theta, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}\right).$$

However, in this scenario the full conditional distribution of $(k_1, ..., k_m)$ does not take a simple form because of the potential coalescences of the paths and an MH step is required.

3.3 Particle Marginal Metropolis-Hastings Sampler

Assume we are interested in sampling from $\pi(\theta, \mathbf{x})$ defined in (3.9). In cases where $\mathbf{x}$ and θ are highly correlated, the Gibbs sampling strategy described in the previous section might be inefficient. In this case, we might want to sample directly from the marginal $\pi(\theta)$ using say an MH algorithm. However, if $\pi(\theta)$ is unknown even up to a normalising constant, this is impossible. We present here a particle approximation of this "ideal" marginal MH algorithm.

Our algorithm is based on the following remark. Consider a MH algorithm of target density $\pi(\theta, \mathbf{x})$ and proposal density

$$q\left((\theta, \mathbf{x}), (\theta^*, \mathbf{x}^*)\right) = q(\theta, \theta^*)\, \pi\left(\mathbf{x}^* | \, \theta^*\right) \ .$$

Then the acceptance ratio is given by

$$1 \wedge \frac{\pi(\theta^*, \mathbf{x}^*)}{\pi(\theta, \mathbf{x})} \frac{q\left((\theta^*, \mathbf{x}^*), (\theta, \mathbf{x})\right)}{q\left((\theta, \mathbf{x}), (\theta^*, \mathbf{x}^*)\right)} = 1 \wedge \frac{\pi(\theta^*)}{\pi(\theta)} \frac{q(\theta^*, \theta)}{q(\theta, \theta^*)}.$$

Consequently, provided that it is possible to sample the component $\mathbf{x}$ from a good approximation of $\pi\left(\mathbf{x} | \, \theta\right)$ then the resulting MH algorithm will essentially approximate the "marginal" MH algorithm.

To build a proposal approximating $\pi(\mathbf{x}|\theta)$, we use an SMC method and introduce as in the PG sampler a sequence of distributions $\{\pi_n(\mathbf{x}_n|\theta)\}$ defined in (3.11) and some transition kernels $\{M_n^\theta(\mathbf{x}_{n-1}, x_n)\}$. The key point is that the normalising constant estimate obtained at time p is an estimate of $\gamma(\theta)$, the unnormalised version of $\pi(\theta)$, i.e. recall that $\gamma(\theta, \mathbf{x}) = \pi(\mathbf{x}|\theta)\gamma(\theta)$ from (3.10).

3.3.1 Algorithm

Algorithm 3.8: Particle Marginal Metropolis-Hastings Sampler

Initialisation $i = 0$

Set randomly $\theta(0)$

Run an SMC algorithm targeting $\pi(\mathbf{x}|\theta(0))$

Sample $\mathbf{X}(0) \sim \widehat{\pi}^N(\cdot|\theta(0))$ and set $\widehat{\gamma}^N(\theta(0)) = \widehat{Z}^N$

For iteration $i \geq 1$

Sample $\theta^* \sim q(\theta(i-1), \cdot)$

Run an SMC algorithm targeting $\pi(\mathbf{x}|\theta^*)$ sample $\mathbf{X}^* \sim \widehat{\pi}^N(\cdot|\theta^*)$ and set $\widehat{\gamma}^N(\theta^*) = \widehat{Z}^N$

With probability

$$1 \wedge \frac{\widehat{\gamma}^N(\theta^*)}{\widehat{\gamma}^N(\theta(i-1))} \frac{q(\theta^*, \theta(i-1))}{q(\theta(i-1), \theta^*)} \tag{3.19}$$

set $\theta(i) = \theta^*$, $\mathbf{X}(i) = \mathbf{X}^*$, $\widehat{\gamma}^N(\theta(i)) = \widehat{\gamma}^N(\theta^*)$,
otherwise set $\theta(i) = \theta(i-1)$, $\mathbf{X}(i) = \mathbf{X}(i-1)$,
$\widehat{\gamma}^N(\theta(i)) = \widehat{\gamma}^N(\theta(i-1))$

The particle marginal Metropolis Hastings (PMMH) sampler is given in Algorithm 3.8. Note that the acceptance probability (3.19) is independent of $\mathbf{X}^*$ so that it is possible to sample $\mathbf{X}^*$ only once the move is accepted and, if we are only interested in estimating $\pi(\theta)$, then this simulation step can always be omitted. The computational complexity of each iteration is of order $O(N)$. Under mild assumptions the acceptance probability of this sampler converges to the acceptance

probability of the "ideal" marginal MH algorithm as

$$\frac{\widehat{\gamma}^N(\theta^*)}{\widehat{\gamma}^N(\theta(i-1))} \to \frac{\gamma(\theta^*)}{\gamma(\theta(i-1))} = \frac{\pi(\theta^*)}{\pi(\theta(i-1))}$$

when $N \to \infty$. Moreover, under mixing assumptions, the variance of the acceptance ratio is proportional to p/N. We can also show under mild assumptions (A1-A5,A7) that for any $N \geq 1$ the PMMH sampler generates a sequence $\{\theta(i), \mathbf{X}(i)\}$ whose sequence of distributions $\{\mathcal{L}^N((\theta(i), \mathbf{X}(i)) \in \cdot)\}$ satisfies

$$\left\|\mathcal{L}^N((\theta(i), \mathbf{X}(i)) \in \cdot) - \pi(\cdot)\right\| \to 0 \text{ as } i \to \infty .$$

3.3.2 Proof of Validity

We show here that the PMMH is a standard MCMC algorithm on an extended space and that for all $N \geq 1$ it generates a sequence converging towards the target distribution of interest. The following assumption will guarantee convergence.

(**A7**) The "ideal" MH sampler of target density $\pi(\theta)$ and proposal density $q(\theta, \theta^*)$ is irreducible and aperiodic (and hence converges for π-almost all starting points).

Theorem 5 *Assume (A1)-(A5), then for any $N \geq 1$ the PMMH sampler is an MH sampler defined on the extended space $\Theta \times \{1, \ldots, N\} \times \mathcal{X}^{pN} \times \{1, \ldots, N\}^{(p-1)N+1}$ with target density (3.15) and proposal density*

$$q(\theta, \theta^*) \times w_p^{*k} \times \psi^{\theta^*}\left(\overline{\mathbf{x}}_1^*, ..., \overline{\mathbf{x}}_p^*, \mathbf{a}_1^*, ..., \mathbf{a}_{p-1}^*\right)$$

where ψ^{θ^*} is defined in (3.16) and $\{w_p^{*k}\}$ consists of the normalised importance weights associated to the proposed population of particles as defined in (2.3).

Assume in addition that (A7) holds. Then for any $N \geq 1$ the PMMH sampler generates a sequence $\{\theta(i), \mathbf{X}(i)\}$ whose sequence of distributions $\{\mathcal{L}^N((\theta(i), \mathbf{X}(i)) \in \cdot)\}$ satisfies

$$\left\|\mathcal{L}^N((\theta(i), \mathbf{X}(i)) \in \cdot) - \pi(\cdot)\right\| \to 0 \text{ as } i \to \infty .$$

The proof can be found in Appendix A.

3.3.3 Extensions

Alternative Proposals

It can be difficult to select the proposal distribution for θ. However, our framework is flexible enough that it allows us to design proposal distributions for θ relying on the set of current particles. For example, if $\pi(\theta|\mathbf{x})$ is a standard distribution dependent on a set of sufficient statistics of $\mathbf{x}$ then we could use an independent proposal of the same functional form with sufficient statistics computed using the set of current particles with possibly an inflated variance, as long as the dimension of θ is not too large.

Optimisation

To sample from (3.17), we can use straightforwardly the PMMH by sampling m independent SMC algorithms at each iteration i. This is shown in Algorithm 3.9.

Algorithm 3.9: PMMH with m Independent SMC Algorithms

For iteration $i \geq 1$

Sample $\theta^* \sim q(\theta(i-1), \cdot)$

Run m independent SMC targeting $\pi(\mathbf{x}|\theta^*)$ and let

$$\widehat{\gamma}_m^N(\theta^*) = \prod_{i=1}^{m} \widehat{Z}_i^N$$

where $\widehat{Z}_i^N$ is the normalising constant estimate provided by the ith SMC algorithm.

With probability

$$1 \wedge \frac{\widehat{\gamma}_m^N(\theta^*)}{\widehat{\gamma}_m^N(\theta(i-1))} \frac{q(\theta^*, \theta(i-1))}{q(\theta(i-1), \theta^*)}$$

set $\theta(i) = \theta^*$, $\widehat{\gamma}_m^N(\theta(i)) = \widehat{\gamma}_m^N(\theta^*)$,
otherwise set $\theta(i) = \theta(i-1)$, $\widehat{\gamma}_m^N(\theta(i)) = \widehat{\gamma}_m^N(\theta(i-1))$

Under (A1)-(A5), the invariant distribution of the Markov kernel associated to this algorithm is $\pi_m(\theta) \propto \pi^m(\theta)$ for any $N \geq 1$.

3.4 Extensions

In this section, we present various extensions of the PMCMC methodology and discuss connections with earlier work. The content of this section is not required to understand the examples in Section 4, where applications of the PMCMC methodology are presented.

3.4.1 Dynamic and Optimal Resampling

In many applications of SMC, the resampling step is only performed when the accuracy of the estimator is poor. Practically, this is assessed by looking at the variability of the weights using the so-called effective sample size (ESS) criterion [61, pp. 35-36] given at time n by

$$\text{ESS} = \left(\sum_{i=1}^{N} \left(W_n^i \right)^2 \right)^{-1}.$$

Its interpretation is that inference based on the N weighted samples is approximately equivalent to inference based on ESS perfect samples from the target. The ESS takes values between 1 and N and we resample only when it is below a threshold N_T.

All the strategies presented in the previous sections can also be applied in this context. The PMH and PMMH can be implemented in the dynamic resampling context without any modification. However, the PG is more difficult to implement as the conditional SMC step requires simulating a set of $N-1$ particles not only consistent with a 'frozen' path but also consistent with the resampling times of the SMC method used to generate the 'frozen' path. For example assume that the 'frozen' path has been generated by an SMC algorithm resampling at time $n_1, n_2, ..., n_m$. Then the conditional SMC method has to generate a set of $N-1$ particles which are such that when the ESS is computed with this $N-1$ 'free' particles plus the frozen one, then it is below the threshold N_T at times $n_1, n_2, ..., n_m$

and only at these time indices. This implies that the $N-1$ particles cannot be simulated independently anymore. The easiest way to implement the conditional SMC consists of using a rejection method. However the acceptance rate of this rejection step is expected to be only reasonable when N is large - as the variance of ESS is inversely proportional to N - and if p is moderate.

The SMC algorithm proposed in [32], [34] for discrete-valued variables is not of the form described in Section 2.2 as the 'sampling' step is deterministic and the proposed so-called optimal resampling step does not attribute equal weights to all particles. It can be shown though that we can use it directly within the PMH and PMMH algorithms. However the peculiar structure of the optimal resampling step seems to prevent the development of a simple PG-type algorithm.

3.4.2 Arbitrary Target Distribution

For the time being, we have considered it possible to devise a sequence of distributions of increasing dimensions which progressively evolves towards the target distribution of interest. However, there are many problems where such a decomposition might not be obvious to define. In this case, it remains possible to use SMC to sample from any target distribution using the general methodology presented in [65]; see also [17], [45], [56].

Assume now that the target $\tilde{\pi}(x)$ is defined on $\mathcal{X}$. To sample from $\tilde{\pi}(x)$ using SMC-type ideas, we introduce a sequence of $p-1$ intermediate distributions $\{\tilde{\pi}_n(x)\}$ on $\mathcal{X}$ moving smoothly from $\tilde{\pi}_1(x)$ – an easy to sample distribution – to $\tilde{\pi}_p(x) = \tilde{\pi}(x)$, where $\tilde{\pi}_n(x) = Z_n^{-1}\tilde{\gamma}_n(x_n)$. Here $\tilde{\gamma}_n : \mathcal{X} \to \mathbb{R}^+$ is known pointwise but Z_n might be unknown. For example, we could have $\tilde{\pi}_n(x) \propto [\tilde{\pi}(x)]^{\eta_n}$ where $0 < \eta_1 < \eta_2 < \cdots < \eta_p = 1$ or, in a Bayesian context, $\tilde{\pi}_n(x)$ could be the posterior distribution of X given observations until time n [17]. To move from $\tilde{\pi}_{n-1}(x_{n-1})$ to $\tilde{\pi}_n(x_n)$ we use a sequence of Markov kernels $M_n(\mathbf{x}_{n-1}, x_n) = M_n(x_{n-1}, x_n)$. In most applications, M_n is an MCMC kernel of invariant distribution $\tilde{\pi}_n$ although this is not necessary. We then build the distributions $\pi_n(\mathbf{x}_n)$ on $\mathcal{X}^n$ for $n = 2, ..., p$ using

$$\gamma_n(\mathbf{x}_n) = \tilde{\gamma}_n(x_n) \prod_{k=1}^{n-1} L_k(x_{k+1}, x_k)$$

where $\{L_n\}$ is a sequence of auxiliary (backward) Markov kernels giving the probability density to move from x_{n+1} to x_n. By construction, we have

$$\tilde{\pi}(x_p) = \int \pi_p(\mathbf{x}_p)\, d\mathbf{x}_{p-1} .$$

Once $\{\tilde{\pi}_n\}$, $\{M_n\}$ and $\{L_n\}$ have been selected (see [65] for guidelines), then we can straightforwardly implement the algorithms proposed in Section 3.1 to sample from $\pi_p(\mathbf{x}_p)$ and hence from its marginal $\pi_p(x_p) = \widetilde{\pi}(x_p)$.

We can also directly adapt the PMMH presented in Section 3.3 to sample from $\tilde{\pi}(\theta, x) = Z^{-1}\tilde{\gamma}(\theta, x)$ on some space $\Theta \times \mathcal{X}$ by defining

$$\pi_n(\mathbf{x}_n|\,\theta) = \frac{\gamma_n(\theta, \mathbf{x}_n)}{Z_n^\theta}$$

where

$$\gamma_n(\theta, \mathbf{x}_n) = \tilde{\gamma}_n(\theta, x_n) \prod_{k=1}^{n-1} L_k^\theta(x_{k+1}, x_k)$$

and $\tilde{\gamma}_p(\theta, x_p) = \tilde{\gamma}(\theta, x_p)$. However, even if we can sample from $\tilde{\pi}(\theta|\, x)$, the PG sampler proposed in Section 3.2 cannot be applied as it does not require sampling from $\tilde{\pi}(\theta|\, x_p)$ but from $\pi_p(\theta|\, \mathbf{x}_p)$ which is usually intractable. To bypass the simulation step from $\pi_p(\theta|\, \mathbf{x}_p)$, we can use a collapsed Gibbs sampler strategy where we first sample θ from $\tilde{\pi}(\theta|\, x_p)$ then sample $X_{p-1}, ..., X_1$ from

$$\prod_{k=1}^{n-1} L_k^\theta(x_{k+1}, x_k) .$$

3.5 Discussion

The PMH algorithm presented in Section 3.1 is related to the configurational bias Monte Carlo (CBMC) method which is a very popular method in molecular simulation used to sample long proteins [40, 74]. However, on the contrary to the PMH algorithm, the CBMC algorithm does not propagate N particles in parallel. Indeed, at each time step n, the CBMC algorithm samples N particles but the resampling step is such that a single particle survives, to which a new set of N offsprings is then

attached. Using our notation, this means that the CBMC algorithm corresponds to the case where $A_n^i = A_n^j$ for all $i, j = 1, \ldots, N$ and $A_n^1 \sim r(\cdot|\mathbf{W}_n)$ *i.e.* at any time n, all the children share the same and unique parent particle. The problem with this approach is that it is somewhat too greedy and that if a "wrong" decision is taken too prematurely then the proposal will be most likely rejected. It can be shown that the acceptance probability of the CBMC algorithm does not converge to 1 for $p > 1$ as $N \to \infty$ contrary to that of the PMH algorithm. It has been more recently proposed in [19] to improve the CBMC algorithm by propagating forward several particles simultaneously in the spirit of the PMH algorithm. However, contrary to us, the authors in [19] propose to kill or multiply particles by comparing their unnormalised weights $w_n\left(\mathbf{X}_n^i\right)$ with respect to some pre-specified lower and upper thresholds; *i.e.* the particles are not interacting and their number is a random variable. In simulations, they found that the performance of this algorithm was very sensitive to the values of these thresholds. Our approach has the great advantage of bypassing the delicate choice of such thresholds.

In statistics, a variation of the CBMC algorithm known as the multiple-try Metropolis (MTM) has been introduced in the specific case where $p = 1$ in [60]. The key of our methodology is to build efficient proposals using sequential and interacting mechanisms for cases where $p \gg 1$: the sequential structure might be natural for some models but can also be induced in other scenarios in order to take advantage of the potential improvement brought in by the interacting mechanism. In this respect, both methods do not apply to the same class of problems.

The idea of approximating the marginal MH algorithm which samples directly from $\pi(\theta)$, by approximately integrating out the latent variables $\mathbf{x}$, was proposed in [6] then generalised and studied theoretically in [1]. The present work is a simple mechanism which opens up the possibility to make this approach viable in large dimensional setups. Indeed the PMMH approach is expected to lead to approximations of $\pi(\theta)$ (up to a normalising constant) with a variance which for high-dimensional problems is likely to scale favourably compared to estimates based, for example, on standard non-sequential and non-interacting importance sampling. The results in [1] suggest that this is a property of paramount importance in order to design efficient marginal MCMC algorithms.

Chapter 4

Applications

4.1 Nonlinear State-Space Model

Consider the following non-linear non-Gaussian dynamic model where $\{X_n\}_{n\geq 1}$ is an unobserved Markov process defined by $X_1 \sim \mu_\theta(\cdot)$ and for $n > 1$

$$X_n | X_{n-1} \sim f_\theta(\cdot | X_{n-1})$$

where the observations $\{Y_n\}_{n\geq 1}$ are assumed independent conditional upon $\{X_n\}_{n\geq 1}$ with

$$Y_n | X_n \sim g_\theta(\cdot | X_n) \ .$$

The parameter θ is also unknown and follows the prior distribution $p(\theta)$. Having observed $y_{1:T}$, we are interested in sampling from the target distribution

$$p(\theta, x_{1:T} | y_{1:T}) \propto p(\theta)\, \mu_\theta(x_1)\, g_\theta(y_1 | x_1) \prod_{n=2}^{T} f_\theta(x_n | x_{n-1})\, g_\theta(y_n | x_n) \ .$$

Although sampling from $p(\theta | y_{1:T}, x_{1:T})$ can typically be performed using standard techniques, sampling from $p(x_{1:T} | y_{1:T}, \theta)$ is impossible except for linear Gaussian models and finite state-space Markov chains [14], [41]. The standard approach consists of sampling alternatively from the full conditional $p(x_n | y_n, x_{n-1}, x_{n+1}, \theta)$ directly or using a MH step but this strategy can be ex-

tremely inefficient due to the strong dependence between x_n and (x_{n-1}, x_{n+1}). A more efficient strategy would consists of sampling blocks of variables say $p(x_{n:n+L-1}|\, y_{n:n+L-1}, x_{n-1}, x_{n+L}, \theta)$. When this distribution is log-concave, it is possible to design efficient proposal distributions [71]. However for more complex scenarios, this can be very challenging. We propose here to use the PMH algorithm to sample from $p(x_{1:T}|\, y_{1:T}, \theta)$ or from subblocks $p(x_{k:k+L-1}|\, y_{k:k+L-1}, x_{k-1}, x_{k+L}, \theta)$ whenever T is too large. In this case, the "natural" sequence of bridging distributions to consider is $\pi_n(\mathbf{x}_n) = p(\mathbf{x}_n|\, y_{1:n}, \theta)$.

We illustrate our methodology on the following example

$$X_n = \frac{X_{n-1}}{2} + 25\frac{X_{n-1}}{1+X_{n-1}^2} + 8\cos(1.2n) + V_n \tag{4.1}$$

$$Y_n = \frac{X_n^2}{20} + W_n \tag{4.2}$$

where $X_1 \sim \mathcal{N}(0, 5)$, $V_n \overset{\text{iid}}{\sim} \mathcal{N}\left(0, \sigma_V^2\right)$, $W_n \overset{\text{iid}}{\sim} \mathcal{N}\left(0, \sigma_W^2\right)$; $\mathcal{N}\left(m, \sigma^2\right)$ denotes the Gaussian distribution of mean m and variance σ^2. We set $\theta = (\sigma_V, \sigma_W)$. This example is often used in the literature in order to assess the performance of particle methods; *e.g.* [27], [47], [57]. The posterior distribution $p(\mathbf{x}_T|\, y_{1:T}, \theta)$ for this non-linear state-space (NLSS) model is highly multimodal as there is uncertainty about the sign of the state X_n which is only observed through its square.

We generated a set of observations $y_{1:200}$ according to the model (4.1)-(4.2) for $\theta = (\sigma_V, \sigma_W) = \left(\sqrt{10}, 1\right)$. We first compared the acceptance rates for the PMH and CBMC algorithm [40] when sampling from $p(\mathbf{x}_T|\, y_{1:T}, \theta)$ for various lengths T of the observations and a varying number of particles N.

To sample from $p(\mathbf{x}_T|\, y_{1:T})$ we used for the proposal $M_n^\theta(\mathbf{x}_{n-1}, x_n)$ an approximation of the conditional distribution

$$p(x_n|\, y_n, x_{n-1}, \theta) \propto g_\theta(y_n|\, x_n)\, f_\theta(x_n|\, x_{n-1})$$

given by the extended Kalman filter (local-linearisation); see [26] for details. We also used the most basic resampling scheme; *i.e.* multinomial resampling. The results are displayed in Figure 4.1. The PMH algorithm performs consistently

better than the CBMC algorithm, and is able to successfully sample large blocks of variables for a reasonable number of particles. As expected, the PMH algorithm can achieve acceptance rates close to 1 for a sufficiently large number of particles.

We then compared the acceptance rates for proposal from prior (P) versus proposal using local-linearisation (LL). Here we used $\theta = (\sqrt{15}, \sqrt{2})$. The simulations with the P proposal used multinomial resampling at every time step (bootstrap) while the simulations with the LL proposal used dynamic and stratified resampling. The result is shown in Figure 4.2. We found that for low number of particles the LL proposal does slightly better, but as the number of particles increases proposing from the prior does as well or better than LL. This can be explained by the fact that the LL proposal doesn't handle multimodalities well. At low particles it is able to steer more particles into regions of higher posterior density than the much more diffuse prior. However, this advantage becomes less important and eventually the property of being a unimodal Gaussian proposal for a multimodal posterior becomes a handicap for LL as the number of particles increases and the prior is able to propose sufficient number of particles into high probability regions.

We then set the following prior on $\theta = (\theta_1, \theta_2) = (\sigma_V, \sigma_W)$

$$\theta_1^2 \sim \mathcal{IG}a\left(a, b\right), \quad \theta_2^2 \sim \mathcal{IG}a\left(a, b\right),$$

where $\mathcal{IG}a$ is the Inverse-Gamma distribution with $a = b = 0.01$.

To sample from $p\left(\mathbf{x}_T, \theta \middle| y_{1:T}\right)$, we used the PG sampler described in Section 3.2 with for $M_n^\theta\left(\mathbf{x}_{n-1}, x_n\right)$ an approximation of the conditional distribution

$$p\left(x_n \middle| y_n, x_{n-1}, \theta\right) \propto g_\theta\left(y_n \middle| x_n\right) f_\theta\left(x_n \middle| x_{n-1}\right)$$

given by the extended Kalman filter as before (LL proposal). The full conditional distribution $p\left(\theta \middle| y_{1:T}, \mathbf{x}_T\right)$ is of a standard form. We compared the PG sampler to a standard algorithm where one updates the states X_n one-at-a-time using a MH step of invariant distribution $p\left(x_n \middle| y_n, x_{n-1}, x_{n+1}, \theta\right)$ and proposal distribution $M_n^\theta\left(\mathbf{x}_{n-1}, x_n\right)$. In the one-at-a-time algorithm, we updated N times the state variables $\mathbf{X}_T$ at each iteration before updating θ. Hence the PG sampler and the

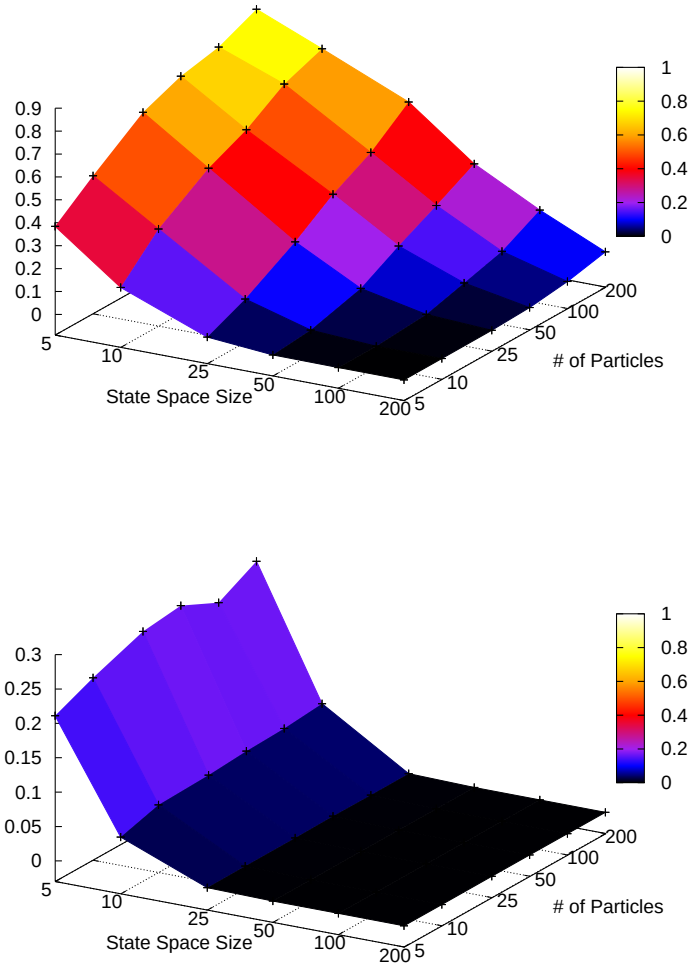


Figure 4.1: Acceptance probabilities for PMH (top) and CBMC (bottom) as a function of T and N. Each point was computed over an average of 250,000 iterations.

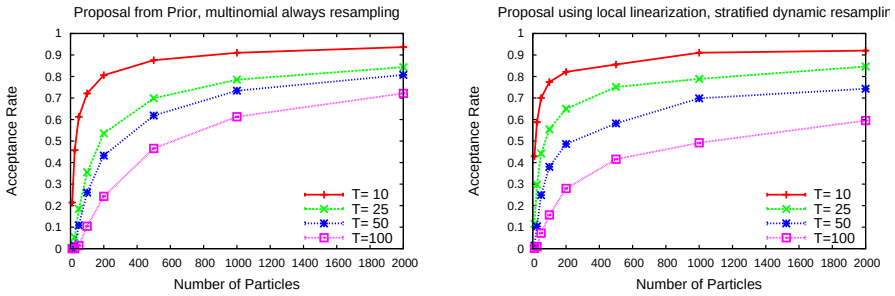


Figure 4.2: PMH acceptance rates for sampling from prior (left) and proposal using local-linearisation (right) for varying number L of observations and number of particles.

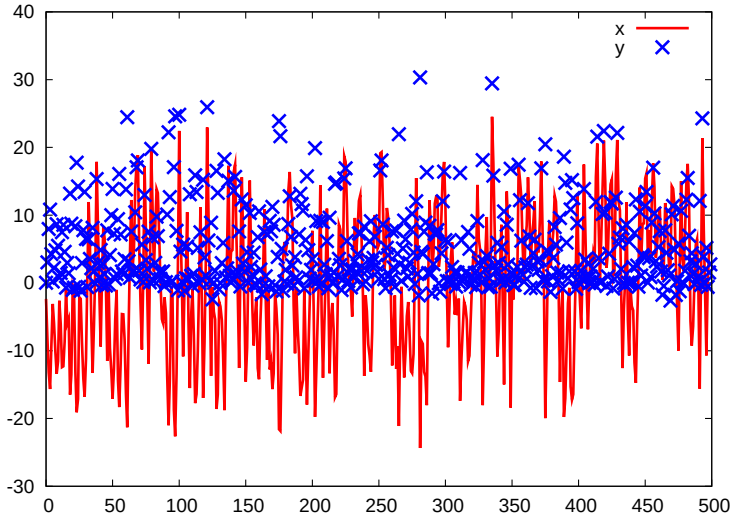


Figure 4.3: Hidden states (x) and observations (y).

standard algorithm have approximately the same computational complexity. Note that in this case, it is difficult to design efficient block proposal distributions to sample from $p\left(x_{k:k+L-1} | \, y_{k:k+L-1}, x_{k-1}, x_{k+L}, \theta\right)$ for $L > 1$ as this distribution can be multimodal.

We used $T = 500$ and $N = 1000$. For θ fixed to its real value, the average ac-

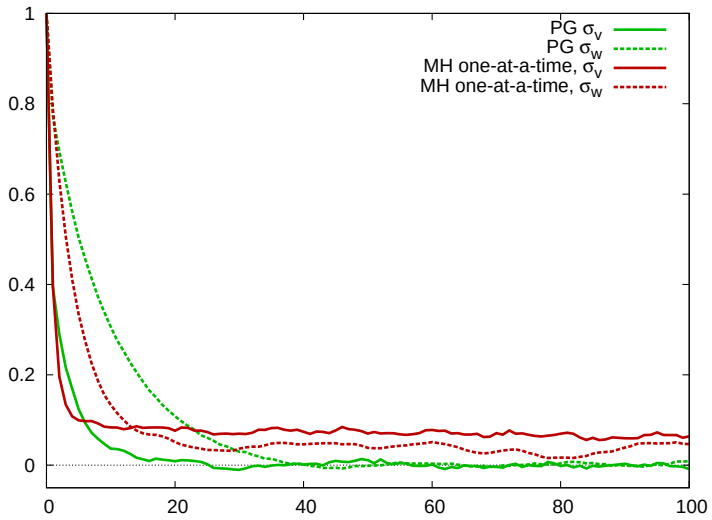


Figure 4.4: ACF of the output of the PG and MH algorithms.

ceptance rate of our PMH algorithm of Section 3.1 using the 'improved' proposal and a multinomial resampling scheme was equal to 0.43. This is quite remarkable as this effectively corresponds to updating 500 real-valued dependent variables simultaneously. We then ran the PG algorithm and the standard one-at-a-time sampler for 50,000 iterations with a burn-in of 10,000 iterations. In Figure 4.4, we display the auto-correlation functions (ACFs) of the parameters (σ_V, σ_W) for the PG and the MH one-at-a-time algorithms obtained using $N = 1000$. The autocorrelation sequences for the various parameters consistently vanish faster for the PG sampler.

In Figure 4.5, we present the estimates of the marginal distributions of the parameters for both algorithms. Contrary to the PG sampler, the standard algorithm clearly misses the main mode of the posterior distribution and over-estimates the parameter σ_V.

The block proposal for the states $\mathbf{X}_T$ of the PG sampler allows us to avoid getting trapped in this secondary mode. Out of 100 independent runs of both algorithms on the same dataset, the PG algorithm never experienced being trapped in such a secondary mode of the target distribution and produced consistent results

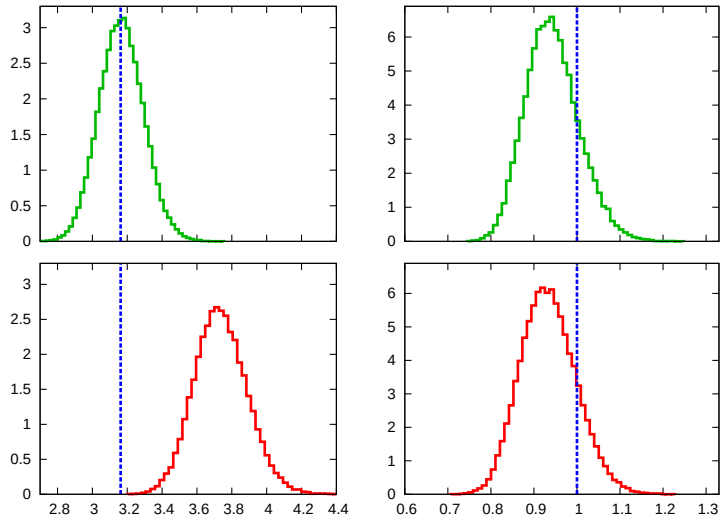
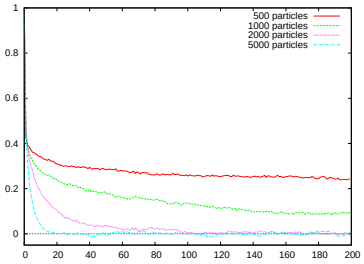


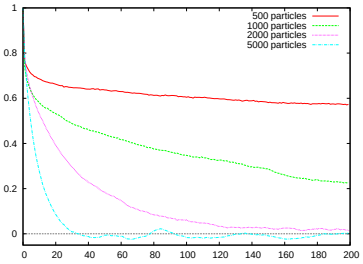
Figure 4.5: Estimates of $p\left(\sigma_V \middle| y_{1:T}\right)$ (left) and $p\left(\sigma_W \middle| y_{1:T}\right)$ (right) for the PG (top) and the MH one-at-a-time (bottom). The vertical lines correspond to the true values.

whereas the standard algorithm, initialised at $\theta = (10, 10)$ for data with ground truth $(\sqrt{10}, 1)$, was trapped 67 times. In practice, we can obviously combine both strategies by only occasionally updating the state variables with the PG sampler to avoid such traps while using more standard and cheaper updates for large proportion of the computational time. In simulations not presented here, we also applied our algorithm to the stochastic volatility model presented in [67], [71]. This model is somewhat 'easier' than the toy example presented here and the particle MCMC algorithms perform very well in this case.

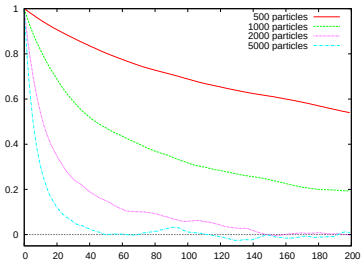
We also compared the performance of the PG sampler with the PMMH sampler. In this case we used the prior as a proposal in the SMC with multinomial resampling. The PMMH used a normal random walk proposal with a diagonal covariance matrix. The standard deviation was equal to 0.15 for σ_V and 0.08 for σ_W. We present in Figure 4.6 the auto-correlation function for (V, W) for the PG and PMMH samplers and various number of particles N. Clearly the performance improves as N increases, and while dependent on the test function f considered



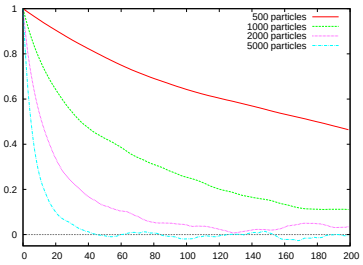
(a) ACF for σ_V – PG sampler



(b) ACF for σ_W – PG sampler



(a) ACF for σ_V – PMMH sampler



(b) ACF for σ_W – PMMH sampler

Figure 4.6: ACF of parameters σ_V and σ_W for the PG sampler and the PMMH sampler.

(as in Eqn. 3.6), it is possible to choose N in order to optimise the ratio integrated autocorrelation function/particle number N. In this scenario, it appears necessary to use at least 2000 particles to get the ACF to drop sharply, whereas increasing N beyond 5000 does not improve performance. That is for $N > 5000$ we observe that the ACFs (not presented here) are very similar to $N = 5000$ and probably very close to that of the corresponding "ideal" MMH algorithm.

4.2 Protein Folding

One of the most important and challenging scientific problems is simulating molecular structures. Protein folding is one of the more well known open problems in structural biology and biophysics. Proteins are the final products of most genes and perform a variety of roles: they serve as structural units, control metabolic pathways, participate in the catalysis or inhibition of various chemical reactions, and serve many other functions. Proteins are chain-like polymers, composed of small subunits (amino acids). There are 20 different types of these amino acids, with a few less common modified versions thereof, which can be loosely grouped into classes based on their chemical properties, such as hydrophobic, hydrophilic (or polar), and charged [8].

The protein folding problem consists of predicting the structure of the protein given its sequence of amino acids. This may include the prediction of the folding pathways to arrive at the "final" shape. The protein structure is dictated by the free energy landscape, lower energy structures being preferred. The native state of a protein is usually at the minimum free energy, and we will consider proteins where this is the case. Exceptions to this can occur when the folding takes place under kinetic control [22]. The function of a protein is primarily determined by its structure, and the sequence of amino acids dictates the structure (Central Dogma of genomics) [22].

There are several different approaches that have been taken to investigate these types of problems, including SMC, CBMC, prune-enriched Rosenbluth method (PERM), recoil-growth (and variations thereof) [40, 61, 74], replica exchange Monte Carlo (parallel tempering) [75], ant colony optimisation [72], and model based search [11].

Next we apply the PMCMC method to this class of problems. We will focus only on the lattice approximations, in particular the HP model, and leave the off-lattice case for future studies.

4.2.1 Lattice Model

Instead of modelling the full protein with all the interacting forces, it is often possible to imitate the protein folding using a simple 2-D or 3-D lattice-bead model.

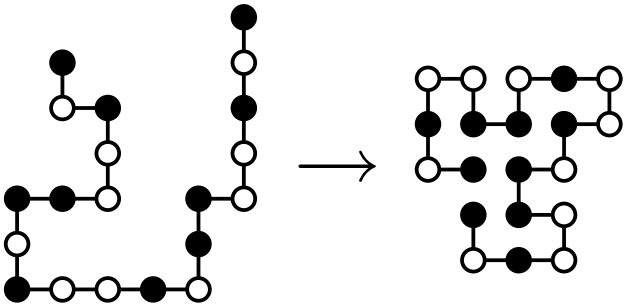


Figure 4.7: Illustration of protein folding in HP model. The left configuration is in a high energy state ($U = 0$), while on the right the protein sequence is folded into into the lowest energy configuration ($U = -9$).

The positions of the amino acids (now beads) in the protein sequence are thus restricted to a lattice and we only need to account for interactions of non-covalent bonds and ensure a self-avoiding walk (SAW), i.e. no two beads can occupy the same position and neighbouring elements in the sequence must be neighbours on the lattice. One of the simplest models used is an Ising model and has only two types of beads: white and black, corresponding to hydrophilic and hydrophobic amino acids, respectively [61]. The energy function for this model is

$$U_n(\mathbf{x}_n) = -\sum_{|i-j|>1} c(x_i, x_j) \tag{4.3}$$

where $c(x_i, x_j) = 1$ if x_i and x_j are non-bonding neighbours and both beads are black (hydrophobic), and $c(x_i, x_j) = 0$ otherwise.

4.2.2 Implementation

We used PMH with sub-block updating and simulated annealing [68, Chapter 5] to optimise the protein configuration for the lowest energy. The inverse temperature for the annealing was increased linearly from $\beta_{\text{min}} = 0.1$ to $\beta_{\text{max}} = 5$ with increments $\triangle\beta$, using $50 \times L$ iterations at each temperature, where L is the length of the protein sequence. The subblocks were chosen at random positions with sizes B sampled from $[2, 20]$, either uniformly or "reverse-linearly", that is, with a distribution $\mathbb{P}_{\text{rl}}(B) \propto 21 - B$. We also used an adaptive number of particles which scaled

linearly with the current block size. This might break detailed balance, however, since this is an optimisation problem, we can afford this approximation and benefit from a more efficient algorithm. To further improve the efficiency of the sampler we implemented the resampling scheme suggested by Fearnhead & Clifford [35]. The method works well for discrete state models, especially when the number of possible states is small.

4.2.3 Results

In order to do comparison with other methods, we ran our method on the data sets given in [75] and [76]. The protein sequences and corresponding lowest (known) energies are given in Tables 4.1 and 4.2, for folding in 2D and 3D, respectively. The results are summarised in Table 4.3. The time is given in seconds. k_r specifies the scaling factor used for adapting the number of particles to the block size (i.e. $N = 1 + k_r \times B$). $\triangle\beta$ is the increment of the inverse-temperature for the annealing. "bsd" is the block size distribution, where "u" indicates uniform and "rl" is reverse-linear. We show both the best performance as well as the median performance over 15 runs at the given settings. The performance of our method is currently not as good as some of the competing methods, such as replica exchange Monte Carlo (REMC) [75], fragment regrowth via energy-guided sequential sampling (FRESS) [76], or specialised versions of PERM (nPERMis [55] and PERM$_{t_{\text{Exp}}}$ [55, 75]). However, the results are encouraging. We are able to find the lowest known energy for most of the protein sequences in a reasonable amount of time. However, we do struggle with the 3D sequences from the data set of [76]. Tables 4.4 and 4.5 compare our results to the other methods for the data sets from [75] and [76], respectively. Further tuning of the method, plus adding some specialised moves could make this a competitive approach.

ID	Length	E_{min}	Protein Sequence
S1-1	20	-9	$(HP)_2PH(HP)_2(PH)_2HP(PH)_2$
S1-2	24	-9	$H_2(P_2H)_7H$
S1-3	25	-8	$(P_2H)_2HP_4H_2P_4H_2P_4H_2$
S1-4	36	-14	$P_3(H_2P_2)_2P_3H_5(H_2P_2)_2P_2H(HP_2)_2$
S1-5	48	-23	$P_2HP_2(H_2P_2)_2P_3H_{10}P_6(H_2P_2)_2HP_2H_5$
S1-6	50	-21	$H(HP)_4H_4PH(P_3H)_2P(P_3H)_3PH_3(HP)_4H_2$
S1-7	60	-36	$P_2H_3PH_8P_3H_9(HP)_2P_2H_{12}P_4H_4(H_2P)_2HP$
S1-8	64	-42	$H_{11}(HP)_3P(H_2P_2)_2HP_2(H_2P_2)_2HP_2(H_2P_2)_2(HP)_2H_{12}$
S1-9	85	-53	$H_4P_4H_{12}P_6H_{12}P_3H_{12}P_3H_{12}P_3HP_2(H_2P_2)_2HPH$
S1-10	100	-50	$P(P_2H_2)_2H_2P_2H(H_2P)_3H_4P_8H_6P_2H_6P_9H(PH_2)_2H_9P_2H(H_2P)_2HP(PH)_2H_2P_6H_3$
S1-11	100	-48	$P_5(PH)_2HP_5H_3PH_3(H_2P)_2P(P_2H_2)_2PH_5PH_8(H_2P)_2H_7P_{11}H_7P(PH)_2H_2P_5(PH)_2H$
2D50	50	-21	$H(HP)_4H_4PH(P_3H)_2P(P_3H)_3PH_3(HP)_4H_2$
2D60	60	-36	$P_2H_3PH_8P_3H_9(HP)_2P_2H_{12}P_4H_4(H_2P)_2HP$
2D64	64	-42	$H_{11}(HP)_3P(H_2P_2)_2HP_2(H_2P_2)_2HP_2(H_2P_2)_2(HP)_2H_{12}$
2D85	85	-53	$H_4P_4H_{12}P_6H_{12}P_3H_{12}P_3H_{12}P_3HP_2(H_2P_2)_2HPH$
2D100a	100	-48	$P_5(PH)_2HP_5H_3PH_3(H_2P)_2P(P_2H_2)_2PH_5PH_8(H_2P)_2H_7P_{11}H_7P(PH)_2H_2P_5(PH)_2H$
2D100b	100	-50	$P(P_2H_2)_2H_2P_2H(H_2P)_3H_4P_8H_6P_2H_6P_9H(PH_2)_2H_9P_2H(H_2P)_2HP(PH)_2H_2P_6H_3$

Table 4.1: Benchmark collection of protein sequences for 2D HP model. H_i and P_i indicate strings of i consecutive H's and P's and $(s)_i$ denotes i repetitions of string s. E_{min} is the lowest known energy.

ID	Length	E_{min}	Protein Sequence
S2-1	48	-32	$\text{HPH}_2\text{P}_2\text{H}(\text{H}_3\text{P})_2\,\text{PH}_2\text{P}(\text{PH})_2\,\text{H}(\text{HP})_2\,(\text{H}_2\text{P}_2)_2$ PHP_8H_2
S2-2	48	-34	$\text{H}_2\,(\text{H}_2\text{P})_2\,\text{H}_5\,(\text{P}_2\text{H})_2\,(\text{HP}_2)_2\,\text{P}_2\,(\text{P}_2\text{H})_2\,\text{P}_3\text{H}\,(\text{P}_2\text{H}_2)_2$ HPH
S2-3	48	-34	$\text{PH}\,(\text{PH}_2)_2\,\text{H}_4\text{P}_2\,(\text{HP})_2\,(\text{PH})_2\,(\text{HP})_3\,\text{P}_2\text{HP}_2\,(\text{H}_2\text{P}_2)_2$ $(\text{HP})_2\,\text{PHP}$
S2-4	48	-33	$(\text{PH})_2\,\text{HP}(\text{PH})_2\,\text{H}_2\text{P}_2\,(\text{H}_2\text{P})_2\,\text{P}_2\text{H}_5\text{P}(\text{PH})_2\,(\text{HP})_3\,\text{P}$ $(\text{P}_2\text{H})_2\,\text{PHP}$
S2-5	48	-32	$\text{P}_2\text{HP}_2\,(\text{PH})_2\,\text{H}_3\text{P}_2\text{H}_2\,(\text{H}_2\text{P})_2\,\text{H}_3\text{P}_2\,(\text{HP})_2\,(\text{HP}_2)_2\,\text{P}_4$ $(\text{H}_2\text{P})_2\,\text{H}$
S2-6	48	-32	$\text{H}_3\text{P}_3\text{H}\,(\text{HP})_2\,(\text{H}_2\text{P})_3\,\text{HP}_7\,(\text{HP})_2\,\text{PHP}\,(\text{P}_2\text{H})_2\,\text{H}_5\text{PH}$
S2-7	48	-32	$\text{PHP}_3\,(\text{PH})_2\,\text{H}\,(\text{HP})_2\,\text{H}_2\,(\text{H}_2\text{P})_3\,\text{P}_2\,(\text{HP})_2\,\text{P}_2\text{H}\,(\text{H}_2\text{P}_2)_3$ PH
S2-8	48	-31	$(\text{PH}_2)_2\,\text{HPH}\,(\text{H}_3\text{P}_2)_2\,\text{P}_3\,(\text{PH})_2\,\text{HP}_2\text{H}\,(\text{HP})_2\,\text{P}_2\text{H}\,(\text{HP})_3$ H_2P_3
S2-9	48	-34	$\text{P}\,(\text{HP})_2\,\text{P}_3\,(\text{HP})_3\,(\text{PH})_2\,\text{H}_5\text{P}_2\text{H}_2\,(\text{HP})_2\,(\text{PH})_2\,\text{HP}$ $(\text{PH})_2\,\text{H}_2\text{P}_4\text{H}$
S2-10	48	-33	$\text{PH}_2\text{P}_3\,(\text{P}_3\text{H}_2)_2\,(\text{HP})_2\,(\text{PH})_2\,\text{H}\,(\text{P}_2\text{H})_2\,(\text{P}_2\text{H}_2)_2$ $\text{H}_5\text{P}_2\text{H}_2$
3D58	58	-44	$\text{PHP}\,(\text{H}_3\text{P})_2\,\text{PH}_2\text{PH}\,(\text{PH}_2)_2\,(\text{HP})_3\,\text{H}_2\text{P}_2\text{H}_3\text{P}_2\,(\text{HP})_2\,\text{P}$ $(\text{P}_2\text{H})_3\,\text{H}\,(\text{P}_2\text{H})_2$
3D64	64	-56	$\text{P}\,(\text{H}_2\text{P})_2\,\text{H}_3\text{P}_2\,(\text{HP})_2\,\text{P}\,(\text{HP})_2\,\text{PH}\,(\text{H}_2\text{P})_3\,\text{P}\,(\text{H}_2\text{P})_2$ $\text{H}_3\text{P}_2\,(\text{HP})_2\,\text{P}\,(\text{HP})_2\,\text{PH}\,(\text{H}_2\text{P})_3$
3D67	67	-56	$\text{PHP}\,(\text{H}_2\text{P})_2\,\text{HP}_2\text{H}_3\text{P}_3\text{HP}\,(\text{H}_2\text{P})_2\,\text{HP}_2\text{H}_3\text{P}_3\text{HP}\,(\text{H}_2\text{P})_2$ $\text{HP}_2\text{H}_3\text{P}_3\text{HP}\,(\text{H}_2\text{P})_2\,\text{HP}_2\text{H}_3\text{P}$
3D88	88	-72	$\text{PHP}\,(\text{H}_2\text{P})_2\,\text{HP}_2\text{H}_2\,(\text{P}_2\text{H})_6\,\text{HP}_2\,(\text{H}_3\text{P}_2)_4\,\text{HP}\,(\text{H}_2\text{P})_2\,\text{H}$ $(\text{P}_2\text{H})_3\,\text{H}\,(\text{P}_2\text{H})_3\,\text{HP}_2\text{HP}$
3D103	103	-57	$\text{P}_2\text{H}_2\text{P}_3\,(\text{P}_2\text{H}_2)_2\,\text{P}\,(\text{HP}_2)_2\,\text{P}_2\,(\text{P}_3\text{H})_2\,\text{HPH}_2\text{P}_4\,(\text{P}_2\text{H})_2\,\text{P}$ $(\text{HP}_2)_2\,\text{P}_3\text{H}_3\text{P}_4\,(\text{H}_2\text{P})_2\,\text{P}_4\text{H}_2\text{P}_4\text{H}_3\,(\text{HP})_2\,\text{P}_7\text{H}_4\,(\text{HP}_2)_2$
3D124	124	-75	$\text{P}_3\text{H}_2\,(\text{HP})_2\,\text{P}_3\text{HP}_5\text{H}_2\text{P}_2\,(\text{P}_2\text{H}_2)_2\,(\text{P}_4\text{H})_2\,(\text{P}_2\text{H})_2$ $\text{HP}_3\text{H}\,(\text{HP})_2\,\text{H}_3\text{P}_4\text{H}_3\text{P}_6\text{H}_2\,(\text{P}_2\text{H})_2\,\text{P}\,(\text{HP}_2)_2\,\text{P}_3\,(\text{P}_2\text{H})_2$ $\text{H}_2\text{P}\,(\text{P}_3\text{H})_2\,\text{H}_4\text{P}_4\text{H}\,(\text{HP})_4\,\text{H}$
3D136	136	-83	$\text{HP}\,(\text{P}_4\text{H})_2\,\text{P}\,(\text{H}_2\text{P})_2\,\text{P}_2\,(\text{PH})_2\,\text{H}_2\text{P}_4\,(\text{HP})_2\,\text{H}_4\text{P}_9\,(\text{P}_2\text{H})_2$ $\text{P}_2\,(\text{PH})_2\,\text{HP}_3\text{H}_2\,(\text{P}_2\text{H})_2\,\text{P}\,(\text{HP})_2\,\text{P}_4\,(\text{P}_3\text{H})_2\,\text{H}_5\text{P}$ $(\text{P}_2\text{H}_2)_2\,\text{HP}_2\,(\text{PH}_2)_2\,\text{H}_3\text{P}_5\,(\text{P}_4\text{H})_2\,\text{PHP}_4$

Table 4.2: Benchmark collection of protein sequences for 3D HP model. H_i and P_i indicate strings of i consecutive H's and P's and $(s)_i$ denotes i repetitions of string s. E_{min} is the lowest known energy.

		Best performance			*Median performance*			*Setting*		
Protein	$E_{\min}$	E	Time	#iter	E	Time	#iter	k_r	$\triangle\beta$	bsd
S1-1	-9	-9	0.0	0 K	-9	0.1	0 K	2	.100	u
S1-2	-9	-9	0.1	1 K	-8	0.2	2 K	1	.100	rl
S1-3	-8	-8	0.1	1 K	-7	0.3	2 K	1	.100	u
S1-4	-14	-14	0.4	1 K	-12	0.1	1 K	2	.100	u
S1-5	-23	-23	0.4	4 K	-20	0.8	6 K	1	.100	rl
S1-6	-21	-21	0.4	2 K	-17	0.2	1 K	2	.100	rl
S1-7	-36	-36	5.8	56 K	-35	2.4	24 K	1	.010	rl
S1-8	-42	-42	22.5	80 K	-38	9.8	58 K	2	.010	u
S1-9	-53	-53	242.9	1535 K	-51	71.2	725 K	1	.001	u
S1-10	-50	-49	182.0	696 K	-47	189.1	440 K	3	.001	rl
S1-11	-48	-47	4.2	43 K	-45	2.5	26 K	1	.010	rl
2D50	-21	-21	0.5	2 K	-19	1.0	4 K	2	.100	u
2D60	-36	-36	536.4	1170 K	-35	151.9	571 K	3	.001	u
2D64	-42	-42	6.6	72 K	-40	8.5	59 K	1	.010	rl
2D85	-53	-53	108.5	366 K	-52	77.0	409 K	2	.001	u
2D100a	-48	-48	112.3	737 K	-46	67.7	664 K	1	.001	u
2D100b	-50	-49	52.5	303 K	-46	79.1	283 K	2	.001	rl
S2-1	-32	-32	45.5	216 K	-31	60.1	181 K	2	.001	rl
S2-2	-34	-34	7.1	35 K	-31	4.9	16 K	2	.010	rl
S2-3	-34	-34	8.0	44 K	-31	2.6	23 K	1	.010	u
S2-4	-33	-33	45.5	240 K	-31	18.0	154 K	1	.001	u
S2-5	-32	-32	2.1	9 K	-29	1.3	4 K	2	.100	rl
S2-6	-32	-32	3.7	36 K	-30	4.9	46 K	1	.010	rl
S2-7	-32	-32	117.0	978 K	-31	80.2	454 K	1	.001	rl
S2-8	-31	-31	4.8	41 K	-29	4.3	24 K	1	.010	rl
S2-9	-34	-33	12.8	25 K	-31	22.2	40 K	3	.010	u
S2-10	-33	-33	362.3	1179 K	-33	362.3	1179 K	3	.001	rl
3D58	-44	-43	8.1	24 K	-39	2.7	14 K	2	.010	u
3D64	-56	-52	260.0	835 K	-51	631.8	1200 K	3	.001	rl
3D67	-56	-50	128.8	1130 K	-49	177.8	995 K	1	.001	rl
3D88	-72	-67	607.1	1708 K	-64	223.4	1053 K	2	.001	u
3D103	-57	-55	5.9	36 K	-48	4.9	46 K	1	.010	u
3D124	-75	-70	150.8	515 K	-68	237.2	498 K	3	.001	rl
3D136	-83	-78	66.7	613 K	-74	97.8	884 K	1	.001	rl

Table 4.3: Protein folding results for HP model using PMH with subblock updating and tempering. The time is given in seconds.

Protein	E_{min}	PMCMC		PERM$_{t_{\text{Exp}}}$		ACO-HPPFP-3		REMC$_m$	
		E	$Time$	E	$Time$	E	$Time$	E	$Time$
S1-1	-9	-9	0.03	-9	<1	-9	<1	-9	<1
S1-2	-9	-9	0.07	-9	<1	-9	<1	-9	<1
S1-3	-8	-8	0.14	-8	2	-8	2	-8	<1
S1-4	-14	-14	0.43	-14	<1	-14	4	-14	<1
S1-5	-23	-23	0.37	-23	2	-23	60	-23	<1
S1-6	-21	-21	0.38	-21	3	-21	15	-21	<1
S1-7	-36	-36	5.79	-36	4	-36	1200	-36	13
S1-8	-42	-42	22.50	-42	280800	-42	5400	-42	6
S1-9	-53	-53	242.94	-53	60	-53	1day	-53	38
S1-10	-50	-49	182.00	-50	1200	-49	43200	-50	480
S1-11	-48	-47	4.23	-48	480	-47	36000	-48	72
S2-1	-32	-32	45.54	-32	6	-32	1800	-32	6
S2-2	-34	-34	7.11	-34	18	-34	25200	-34	12
S2-3	-34	-34	8.03	-34	6	-34	7200	-34	6
S2-4	-33	-33	45.47	-33	120	-33	18000	-33	6
S2-5	-32	-32	2.08	-32	30	-32	900	-32	6
S2-6	-32	-32	3.70	-32	6	-32	43200	-32	6
S2-7	-32	-32	116.97	-32	30	-32	43200	-32	18
S2-8	-31	-31	4.85	-31	18	-31	7200	-31	6
S2-9	-34	-33	12.76	-34	300	-34	27000	-34	54
S2-10	-33	-33	362.34	-33	0.6	-33	3600	-33	6

Table 4.4: Comparison of protein folding performance for the data set from [75]. The time is given in seconds.

		PMCMC		FRESS		nPERMis	
Protein	E_{min}	E	$Time$	E	$Time$	E	$Time$
2D50	-21	-21	0.53	-21	–	–	–
2D60	-36	-36	536.41	-36	–	-36	–
2D64	-42	-42	6.62	-42	–	-42	–
2D85	-53	-53	108.51	-53	–	-52	–
2D100a	-48	-48	112.34	-48	–	-48	–
2D100b	-50	-49	52.54	-50	–	-50	–
3D58	-44	-43	8.07	-44	5.4	-44	11.4
3D64	-56	-52	259.98	-56	32.4	-56	27
3D67	-56	-50	128.81	-56	84.6	-56	66
3D88	-72	-67	607.10	-72	301.8	-69	
3D103	-57	-55	5.94	-57	268.2	-55	187.2
3D124	-75	-70	150.82	-75	–	-71	738
3D136	-83	-78	66.65	-83	–	-80	6600

Table 4.5: Comparison of protein folding performance for the data set from [76]. The time is given in seconds.

4.3 Dirichlet Mixture Model

We consider a Dirichlet process (DP) mixture model for the observations $Y_{1:T}$. We have the following hierarchical model

$$\mathbb{G} \sim DP\left(\alpha, \mathbb{G}_0\right),$$

$$U_n | \, \mathbb{G} \overset{\text{i.i.d.}}{\sim} \mathbb{G},$$

$$Y_n | \, U_n \sim g_{U_n}\left(\cdot\right),$$

where $DP\left(\alpha, \mathbb{G}_0\right)$ is a Dirichlet process of base measure $\mathbb{G}_0$ and scale parameter α. By integrating out $\mathbb{G}$ and using the Polya urn representation of the predictive distribution of U_n given $U_{1:n-1}$, we can equivalently reformulate the model by introducing a vector of cluster labels $X_{1:n}$ which satisfy

$$\mathbb{P}\left(X_n = j | \, x_{1:n-1}\right) = \begin{cases} m_{n-1}^j / \left(n - 1 + \alpha\right) & \text{for } j = 1, ..., k_{n-1} \\ \alpha / \left(n - 1 + \alpha\right) & j = k_{n-1} + 1 \end{cases} \tag{4.4}$$

where k_{n-1} is the number of clusters in the assignment $x_{1:n-1}$ and m_{n-1}^j is the number of observations that $x_{1:n-1}$ assigns to cluster j. The cluster locations are such that

$$\theta_k \overset{\text{i.i.d.}}{\sim} \mathbb{G}_0$$

and

$$Y_n | \, \theta_{X_n} \sim g_{\theta_{X_n}}\left(\cdot\right).$$

Assuming α is also unknown with a suitable prior $\pi\left(\alpha\right)$, the posterior of interest is given by $\pi\left(\mathbf{x}_T, \theta_{1:k_T}, \alpha | \, y_{1:T}\right)$. We will consider here the case where the base measure $\mathbb{G}_0$ and the likelihood g_θ are conjugate so that it is possible to compute the marginal $\pi\left(\mathbf{x}_T, \alpha | \, y_{1:T}\right)$ up to a normalising constant. In such situations, several SMC methods have already been proposed to sample from $\pi\left(\mathbf{x}_T | \, y_{1:T}, \alpha\right)$ [32], [61, Section 4.4.2]. We focus here on the following Gaussian mixture model for real-valued observations where $\theta = \left(\mu, \sigma^2\right)$ and

$$\mathbb{G}_0\left(\mu, \sigma^2\right) = \mathcal{IG}\left(\sigma^2; a, b\right) \mathcal{N}\left(\mu; \eta, \tau\sigma^2\right).$$

For the scale parameter, we use

$$\alpha \sim \mathcal{G}a\left(c,d\right). \tag{4.5}$$

Using standard calculations [32], we can show that

$$\pi\left(y_{1:n}|\, x_{1:n},\alpha\right) =$$
$$\prod_{j=1}^{k_n}\left\{\frac{b^a\Gamma\left(a+m_n^j/2\right)}{\Gamma\left(a\right)\sqrt{m_n^j\tau+1}}\left(b+\frac{m_n^j}{2}\left[\left(\widehat{\sigma}_n^j\right)^2+\frac{\left(\overline{y}_n^j-\eta\right)^2}{1+m_n^j\tau}\right]\right)^{-\left(a+m_n^j/2\right)}\right\}$$

where

$$\overline{y}_n^j = \frac{1}{m_n^j}\sum_{k=1}^{n}\mathbb{I}\left(x_k=j\right)y_k,$$
$$\left(\widehat{\sigma}_n^j\right)^2 = \frac{1}{m_n^j}\sum_{k=1}^{n}\mathbb{I}\left(x_k=j\right)\left(y_k-\overline{y}_n^j\right)^2.$$

We cannot sample from $\pi\left(\alpha|\, y_{1:T},\mathbf{x}_T\right)$ directly. However, following [30], we can introduce an auxiliary variable $\upsilon\in(0,1)$ such that

$$\alpha|\, y_{1:T},\mathbf{x}_T,\upsilon \;\sim\; \gamma_\upsilon\,\mathcal{G}a\left(c+k_T,d-\log\upsilon\right)+\left(1-\gamma_\upsilon\right)\mathcal{G}a\left(c+k_T-1,d-\log\upsilon\right)$$

with $\gamma_\upsilon/\left(1-\gamma_\upsilon\right)=\left(c+k_T-1\right)/\left(T\left(d-\log\upsilon\right)\right)$ and

$$\upsilon|\, y_{1:T},\mathbf{x}_T,\alpha \;\sim\; \mathcal{B}e\left(\alpha+1,T\right)$$

where $\mathcal{B}e$ is the Beta distribution.

We first applied our algorithm to the Galaxy dataset [68, p. 426] and used the version of the data in [32] with $T=82$. We used $a=b=1$, $\eta=20$, $\tau=15^2$ [32] and $c=1$, $d=0.5$. To sample from $\pi\left(\mathbf{x}_T,\alpha|\, y_{1:T}\right)$, we ran the PG algorithm of Section 3.2 with $\pi\left(\mathbf{x}_n|\,\alpha\right)=\pi\left(\mathbf{x}_n|\, y_{1:n},\alpha\right)$ and the stratified resampling scheme at each iteration. We also used the PMMH algorithm of Section 3.3 with the same sequence $\pi\left(\mathbf{x}_n|\,\alpha\right)$ and using both stratified and dynamic resampling

		scale parameter		number of clusters	
	S	$\mathbb{E}(\alpha \mid y_{1:T})$	$\mathbb{V}(\alpha \mid y_{1:T})$	$\mathbb{E}(k_T \mid y_{1:T})$	$\mathbb{V}(k_T \mid y_{1:T})$
PG	9000	0.911	0.267	5.342	2.686
PMMH	9000	0.916	0.273	5.416	2.683
PMH	49000	fixed to $\alpha = 1$		5.754	1.843
SMC [32]	50,000	fixed to $\alpha = 1$		5.75	–
Gibbs [32]	50,000	fixed to $\alpha = 1$		5.75	–
Gibbs [30]	10,000	$\sim$1.0	$\sim$0.2	7.01	3.41

Table 4.6: Results for Galaxy data set. S is the number of samples used in the Monte Carlo estimate. Both PG and PMMH used 100 particles. The Gibbs sampler in Escobar & West [30] used the prior $p(\alpha) = \mathcal{G}a(2, 0.25)$.

schemes; resampling being performed when ESS $< N/2$. The proposal for α is a Gaussian random walk proposal of standard deviation 0.5. We ran the algorithms for $10,000$ iterations and a burn-in of $1,000$. For $N = 50$, we found that the PG and the PMMH algorithms provided similar results in terms of posterior mean and posterior variance of α and k_T; the results for the galaxy data set are summarised in Table 4.6, and the histogram, trace, and ACF for the PG and PMMH samplers are shown in Figure 4.8. Increasing the number of particles did not significantly improve the performance of the algorithms. For example for $N = 50$ the average acceptance rate of the PMMH algorithm was 0.62 while for $N \geq 100$ it stabilised at 0.71. This suggests that in this scenario $N = 100$ is sufficient to approximate the marginal MH algorithm. Fearnhead [32] performed inference on the number of clusters using Gibbs sampling and SMC. The reported mean was 5.75 and agrees with our results, however we should note that in [32] the scaling parameter was fixed at $\alpha = 1$ while in our simulations α was also sampled and had mean 0.91 with variance 2.7. Escobar & West [30] analysed the Galaxy data using a $\mathcal{G}a(2, 0.25)$ as a prior for α and obtained a posterior for the scale parameter with $\mathbb{E}(\alpha) \approx 1$ and $\mathbb{V}(\alpha) \approx 0.2$, which is a bit larger and more peaked than what we found. Note that they also learn the prior parameters η and τ and we have used a slightly different prior for α.

We also applied our algorithm to a large simulated dataset of $10,000$ real-valued observations generated from the Dirichlet process Gaussian mixture model.

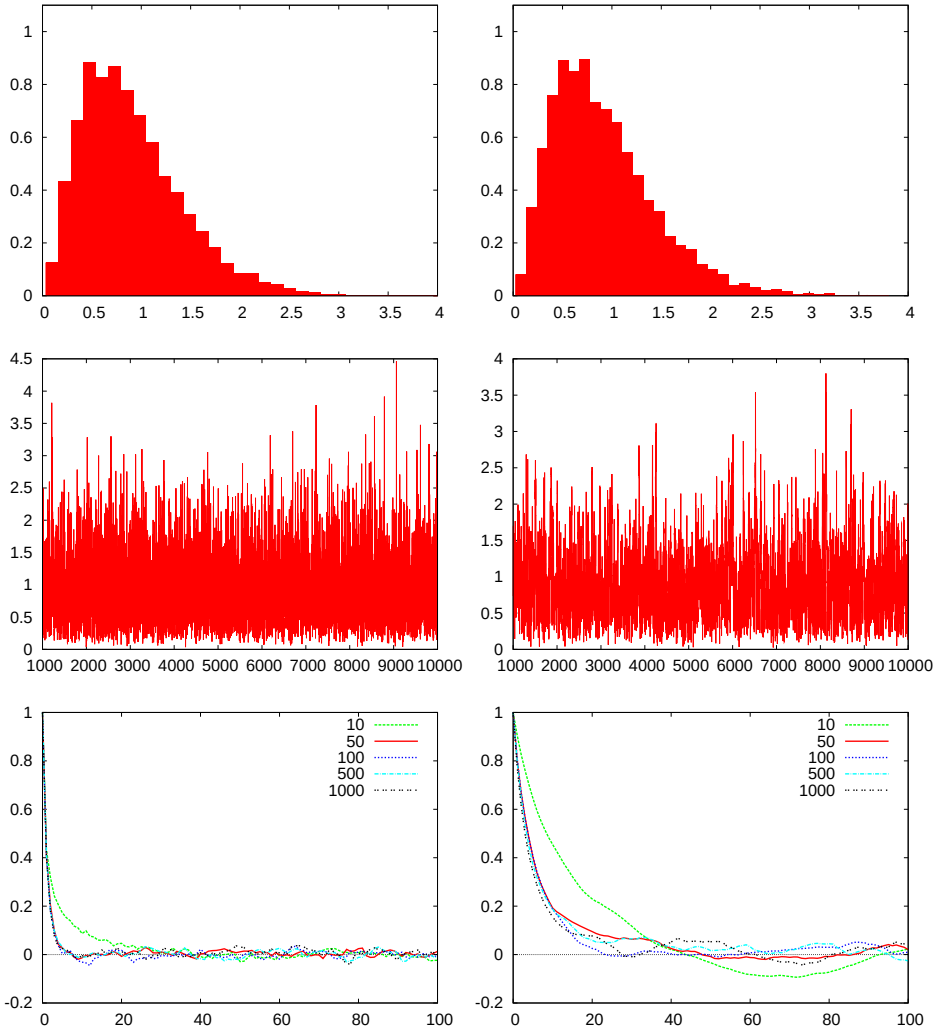


Figure 4.8: Histogram (top), trace (middle), and ACF (bottom) of parameter α for PG (left) and PMMH (right). The histogram and trace plots were obtained using 100 particles. The PG resampled at every time step, while the PMMH used dynamic resampling. Both used stratified resampling.

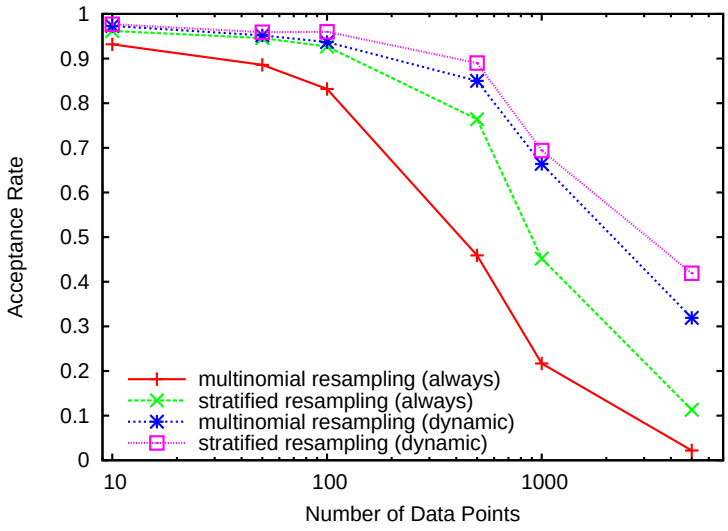


Figure 4.9: Average acceptance rate for various versions of the PMH algorithm for varying number of data points and $N = 100$

For α fixed to its true value, $\alpha = 1$, we compared the performance of the PMH sampler for varying SMC schemes and numbers of data points for $N = 100$ particles. The results are displayed in Figure 4.9. Even for only $N = 100$ particles, the acceptance rate of the PMH algorithms based on SMC proposals with dynamic resampling is equal to 0.31 for $T = 10,000$ observations.

For $T = 10,000$ we also implemented the PG sampler to sample from $\pi(\mathbf{x}_T, \alpha | y_{1:T})$ based on an SMC proposal using stratified resampling at each iteration . We ran the algorithm for $10,000$ iterations with $1,000$ iterations for the burn-in. In Figure 4.10, we display the autocorrelation function for the simulated values $\{\alpha^{(i)}\}$ and the estimate of $\pi(\alpha | y_{1:T})$. The PMMH algorithms for stratified and dynamic resampling yield similar results.

We also compared our method to Gibbs sampling. In order to keep the computational complexity the same, we allow the Gibbs sampler to run N full Gibbs updates for the state variables between each parameter update, where N is the number of particles used in the PMMH and PG samplers (see Algorithm 4.1). For this problem however, we found that *particle MCMC does not give any improvement in*

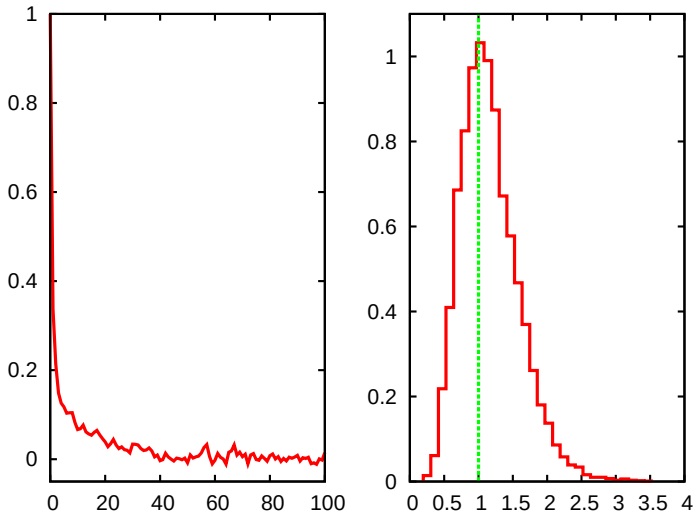


Figure 4.10: Auto-correlation function for parameter α (left) and estimate of $\pi\left(\alpha|\, y_{1:T}\right)$ (right). The vertical line corresponds to the true value.

performance over Gibbs sampling. As there is only weak dependence between the label assignments X_k and between the parameter α and X_k, the PMCMC method has only limited advantage over Gibbs. Given the greater complexity of PMCMC, the simpler Gibbs sampler would be the preferred method to use on this model.

Algorithm 4.1: Gibbs Sampler for Dirichlet Mixture Model.

```
   // see Appendix C.1 for details
1  sample α^(0) from prior
2  initialise state vector X(0)
3  For iteration i ≥ 1
4  |  sample α(i) ~ π(·|X(i − 1))
5  |  set X(i) = X(i − 1)
6  |  for k = 1, .., N do
7  |  |  for n = 1, .., T do
8  |  |  |  sample X_n(i) ~ π(X_n(i)|X_{−n}(i), y_{1:T})
```

4.4 Markov Jump Processes (Lotka-Volterra)

We consider here a discretely observed stochastic kinetic Lotka-Volterra (LV) model [9]; see [49] for a description of such models and their applications in systems biology. For this model it is not possible to compute the prior between two given times; i.e. $\mathbb{P}\left(Z_{t+T}|Z_t\right)$ (integrating out the paths between times t and $t+T$), which prevents us from designing proposals with the observations taken into account. We are thus required to propose from the prior. As we shall see, PMCMC allows us to still obtain good performance despite this limitation.

The LV model describes the evolution of two species Z_t^1 (prey) and Z_t^2 (predator) which are non-negative integer-valued processes. In a small time interval $(t, t+dt]$, there are three possible transitions for the Markov jump process (MJP) $Z_t = \left(Z_t^1, Z_t^2\right)$

$$\begin{aligned}
\mathbb{P}\left(Z_{t+dt}^1 = z_t^1 + 1, Z_{t+dt}^2 = z_t^2 \middle| z_t^1, z_t^2\right) &= \alpha\, z_t^1 dt + o\left(dt\right), \\
\mathbb{P}\left(Z_{t+dt}^1 = z_t^1 - 1, Z_{t+dt}^2 = z_t^2 + 1 \middle| z_t^1, z_t^2\right) &= \beta\, z_t^1\, z_t^2 dt + o\left(dt\right), \\
\mathbb{P}\left(Z_{t+dt}^1 = z_t^1, Z_{t+dt}^2 = z_t^2 - 1 \middle| z_t^1, z_t^2\right) &= \gamma\, z_t^2 dt + o\left(dt\right),
\end{aligned}$$

corresponding respectively to prey reproduction, predator reproduction & prey death, and predator death. We assume that we only observe the process $Y_n = \left(Y_n^1, Y_n^2\right)$ at discrete time points τ_n which is given by

$$Y_n^i = Z_{\tau_n}^i + W_n^i\,, \tag{4.6}$$

where $W_n^i \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(0, \sigma^2 I\right)$ for $i = 1, 2$. The observation times are assumed to be evenly spaced at $\tau_n = n\Delta$. We are interested here in making inference about the parameters $\theta = (\alpha, \beta, \gamma)$ which are assumed to be a priori distributed as

$$\alpha \sim \mathcal{G}a(1, 10), \quad \beta \sim \mathcal{G}a(1, 0.25), \quad \gamma \sim \mathcal{G}a(1, 7.5)$$

where $\mathcal{G}a$ is the Gamma distribution. For the initial populations we used

$$x_0 \sim \mathcal{U}(20, 80),$$

Although it is possible to analyse such models directly using reversible-jump

MCMC (RJMCMC) [9], a very popular alternative consists of using a diffusion approximation instead [49, pp. 188-189] which typically leads to 'simpler' inference algorithms [48]. For the Lotka-Volterra model, one can obtain a two-dimensional diffusion process $X_t = \left(X_t^1, X_t^2\right)$ with drift and instantaneous variance-covariance matrix given by [33]

$$\begin{pmatrix} \alpha X_t^1 - \beta X_t^1 X_t^2 \\ \beta X_t^1 X_t^2 - \gamma X_t^2 \end{pmatrix} \text{ and } \begin{pmatrix} \alpha X_t^1 + \beta X_t^1 X_t^2 & \beta X_t^1 X_t^2 \\ -\beta X_t^1 X_t^2 & \beta X_t^1 X_t^2 + \gamma X_t^2 \end{pmatrix}. \tag{4.7}$$

Most approaches rely on data augmentation (by discretising time using the Euler-Maruyama approximation) to solve the stochastic differential equations (SDEs). We applied PMCMC to this problem and have developed a Gaussian proposal for the SMC using an unscented Kalman smoother (UKS). However, we only had limited success with this approach and have found that a simpler method works much better (and faster) for this model.

Instead of transforming the problem into an SDE, we would prefer to stay in the discrete state space and use the exact model. Unfortunately we cannot evaluate the prior (transition) probability $\mathbb{P}(Z^1_{(n+1)\triangle}, Z^2_{(n+1)\triangle} | z^1_{n\triangle}, z^2_{n\triangle})$. However, we are able to sample exactly from the prior using Gillespie's algorithm [46]. And since we are able to evaluate the likelihood, we can use SMC to sample the states, and PMCMC to sample the parameters (α, β, γ). Of course for this to work efficiently, the likelihood must be sufficiently diffuse in order to get enough particles in regions of high probability.

We generated $T = 50$ observations $y_{1:50}$ by simulating the MJP $Z_t = \left(Z_t^1, Z_t^2\right)$ using Gillespie's algorithm [46, 49] and (4.6) with parameters $\alpha = 2$, $\beta = 0.05$, and $\gamma = 1.5$, $\Delta = 0.2$, and $\sigma^2 = 4$; see Figure 4.11.

Applying PMMH, the parameters were sampled using a Gaussian random walk proposal (independent for each component). In order to improve mixing, random subsets of the parameters were updated at each iteration. The number of components n to update was sampled uniformly. The variances used for the random walk proposal are $s_\alpha^2 = 0.2^2/n$, $s_\beta = 0.005^2/n$, and $s_\gamma = 0.15^2/n$.

The acceptance rate for the sampler was 36%, with an average ESS/N of 17.1% and resample rate of 95%. The simulation took 4hrs (17412.8 seconds) using a sin-

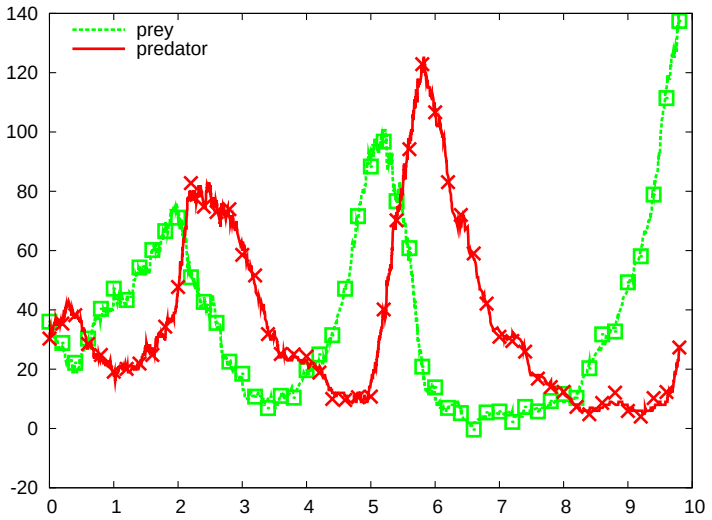


Figure 4.11: Synthetic data generated from the Lotka-Volterra model using Gillespie's algorithm. The prey (Z_t^1) and predator (Z_t^2) are shown in green and red, respectively. The squares and circles indicate the observations Y_n^1 and Y_n^2, respectively.

gle core of an Intel Core2 Duo 6300 @ 1.86GHz. Figure 4.12 shows the histogram and trace plots of the parameters. The algorithm mixes well, and the posterior is centred around the true value. The auto-correlation functions of the parameters are shown in Figure 4.13. PMMH also gives posterior estimates of the hidden states at the observation times, as shown in Figure 4.14.

Boys et al. [9] perform exact inference (i.e. without SDE approximation) on this type of model using RJMCMC and a block updating (BU) method. They use a different observation model, where the prey is observed without error, but the predator is unobserved. In their BU method, the latent process (reaction events) is updated in blocks consisting of pairs of intervals between observations, keeping the endpoints (i.e. population sizes at the beginning and end of the block) fixed and using a random walk proposal on the number of reactions. The parameters are updated conditional on sampled events. The authors note that, while RJMCMC mixes much worse than BU and requires significantly more CPU time, both "algorithms

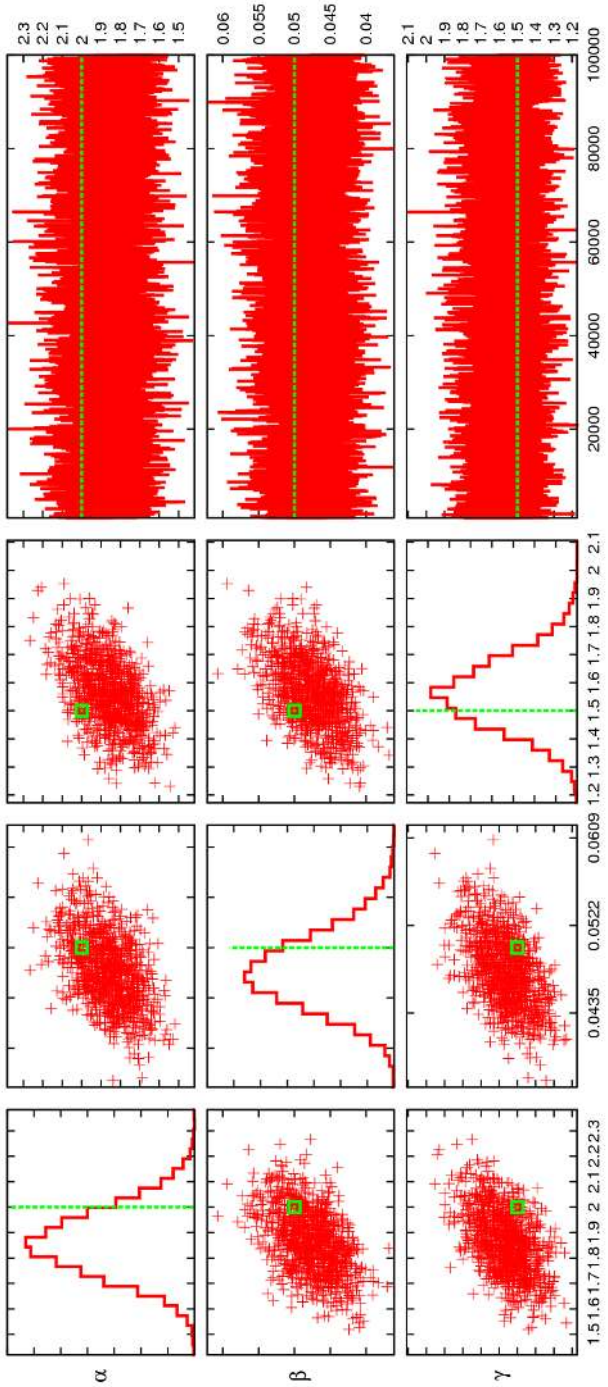


Figure 4.12: Histogram and trace plots of the sampled parameters.

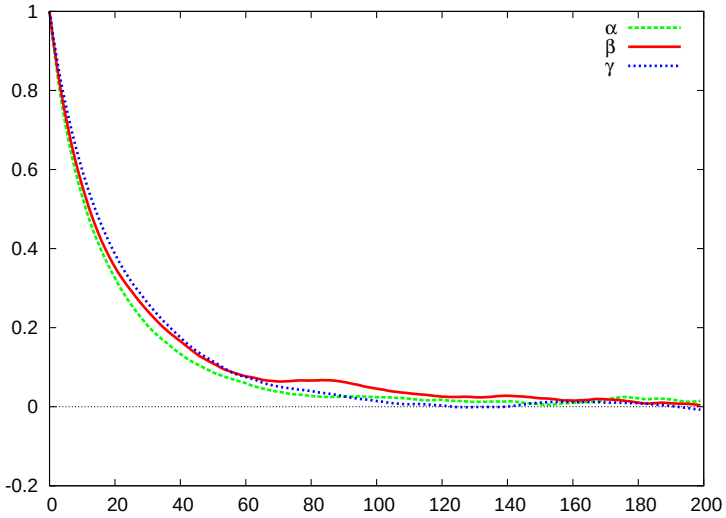


Figure 4.13: ACFs of the sampled parameters.

suffered significant mixing problems" when the predators where unobserved, and due to CPU constraints no satisfactory results were obtained using RJMCMC. As evident from the trace plots in Figure 4.12 PMCMC does not suffer from this (although note that a direct comparison should be taken with a grain of salt due to the difference in the observation model).

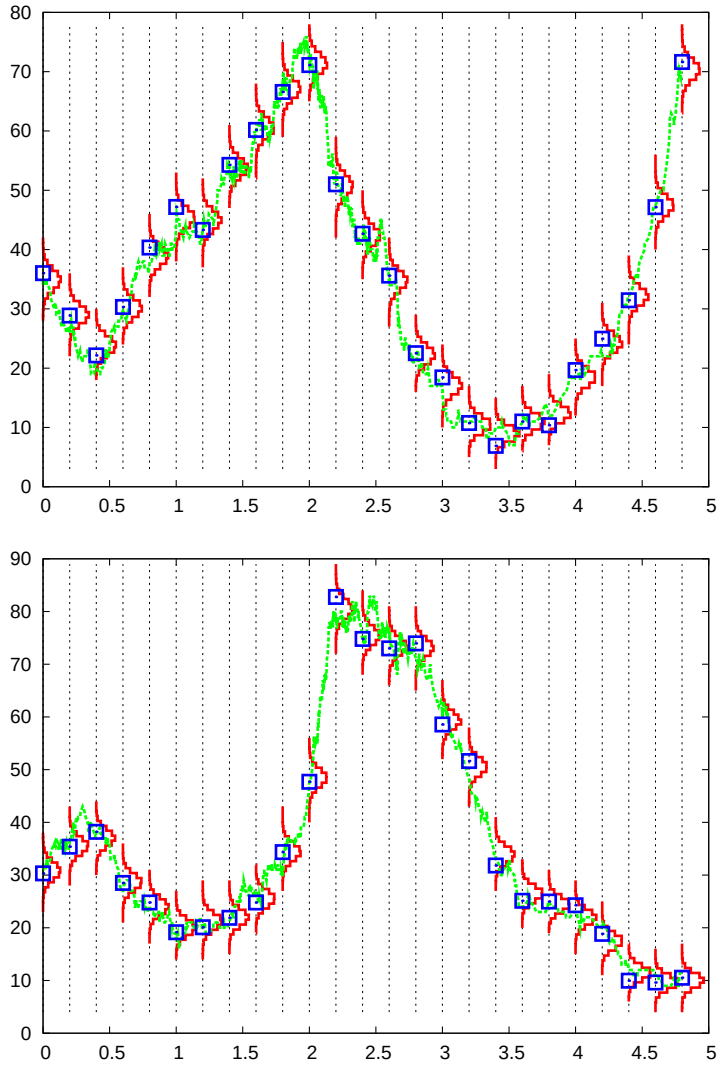


Figure 4.14: Histograms of prey (top) and predator (bottom) states. The ground truth trajectory and observations are shown in green and blue, respectively.

4.5 Lévy-Driven Stochastic Volatility Model

We now consider another model in which the prior cannot be evaluated, but we can sample from it. This model has the additional challenge of complex dependence between the parameters.

The logarithm of an asset price $y^*(t)$ is assumed to be determined by the following stochastic differential equation

$$dy^*(t) = \left(\mu + \beta\sigma^2(t)\,dt\right) + \sigma(t)\,dB(t)$$

where μ is the drift parameter, β the risk premium and $B(t)$ is a Brownian motion. The instantaneous latent variance/volatility $\sigma^2(t)$ is assumed to be stationary and independent from $B(t)$. It is modelled by the following Lévy-driven Ornstein-Uhlenbeck process proposed in [4]

$$d\sigma^2(t) = -\lambda\sigma^2(t)\,dt + dz(\lambda t) \tag{4.8}$$

where $\lambda > 0$ and $z(t)$ is a purely non-Gaussian Lévy process with positive increments with $z(0) = 0$.

We define the integrated volatility

$$\sigma^{2*}(t) = \int_0^t \sigma^2(u)\,du$$

which satisfies from (4.8)

$$\sigma^{2*}(t) = \lambda^{-1}\left\{z(\lambda t) - \sigma^2(t) + \sigma^2(0)\right\}.$$

Let Δ denote the length of time between two periods of interests, then the increments of the integrated volatility satisfy

$$\begin{aligned}\sigma_n^2 &= \sigma^{2*}(n\Delta) - \sigma^{2*}((n-1)\Delta) \\ &= \lambda^{-1}\left\{z(\lambda n\Delta) - \sigma^2(n\Delta) - z(\lambda(n-1)\Delta) + \sigma^2((n-1)\Delta)\right\}\end{aligned}$$

where

$$\begin{pmatrix} \sigma^2(n\Delta) \\ z(\lambda n\Delta) \end{pmatrix} = \begin{pmatrix} \exp(-\lambda\Delta)\,\sigma^2((n-1)\Delta) \\ z(\lambda(n-1)\Delta) \end{pmatrix} + \eta_n$$

with

$$\eta_n \overset{d}{=} \begin{pmatrix} \exp(-\lambda\Delta)\int_0^\Delta \exp(\lambda u)\,dz(\lambda u) \\ \int_0^\Delta dz(\lambda u) \end{pmatrix}. \tag{4.9}$$

By aggregating returns over a time interval of length Δ, we have

$$y_n = \int_{(n-1)\Delta}^{n\Delta} dy^*(t) = y^*(n\Delta) - y^*((n-1)\Delta)$$

thus, conditional on the volatility, we have

$$y_n \sim \mathcal{N}\left(\mu\Delta + \beta\sigma_n^2, \sigma_n^2\right).$$

Many publications have restricted themselves to the case where $\sigma^2(t)$ follows marginally a Gamma distribution in which cases the stochastic integrals appearing in 4.9 can be represented by a finite number of random variables. In this case sophisticated MCMC schemes were developed so as to perform Bayesian inference on λ and the parameters of the Gamma [42, 52, 69]. However, as outlined in [44] the use of Gamma models seems to have been mostly motivated by computational tractability. We address here the case where $\sigma^2(t)$ follows a tempered stable marginal distribution $\mathcal{TS}(\kappa, \delta, \gamma)$ which includes the class of inverse Gaussian distributions for $\kappa = \frac{1}{2}$; this class of models has been successfully used to fit the returns from exchange rate time series [5]. The tempered stable distribution does not admit a closed-form expression, but it is shown in [5] that

$$\sigma^2(0) \overset{d}{=} \sum_{i=1}^{\infty} \left\{ \left(\frac{a_i\kappa}{A_0}\right)^{-1/\kappa} \wedge e_i v_i^{1/\kappa} \right\} \tag{4.10}$$

where $\{a_i\}$, $\{e_i\}$, $\{v_i\}$ are independent of one another. The $\{e_i\}$ are i.i.d. exponential with mean $1/B$, the $\{v_i\}$ are standard uniforms whereas $a_1 < a_2 < \ldots$ are

arrival times of a Poisson process of intensity 1 and

$$A_0 = \delta 2^{\kappa} \frac{\kappa}{\Gamma(1-\kappa)}, B = \frac{1}{2}\gamma^{1/\kappa}.$$

It is also established that to ensure a $\mathcal{TS}(\kappa, \delta, \gamma)$ marginal for $\sigma^2(t)$ then $z(t)$ has to be the sum of an infinite activity Lévy process and of a compound Poisson process such that

$$\eta_n \stackrel{d}{=} \sum_{i=1}^{\infty} \begin{pmatrix} \exp(-\lambda\Delta r_i) \\ 1 \end{pmatrix} \left\{ \left(\frac{a_i \kappa}{A\lambda\Delta} \right)^{-1/\kappa} \wedge e_i v_i^{1/\kappa} \right\} + \sum_{i=1}^{N(\lambda\Delta)} \begin{pmatrix} \exp(-\lambda\Delta r_i^*) \\ 1 \end{pmatrix} c_i \tag{4.11}$$

where

$$A = \delta 2^{\kappa} \frac{\kappa^2}{\Gamma(1-\kappa)}, \; B = \frac{1}{2}\gamma^{1/\kappa}.$$

In (4.11), $\{a_i\}$, $\{e_i\}$, $\{r_i\}$, $\{r_i^*\}$, $\{v_i\}$ are independent of one another. The $\{a_i\}$, $\{e_i\}$, $\{v_i\}$ follow the same distributions as in (4.10), the $\{c_i\}$ are i.i.d. $\mathcal{G}a(1-\kappa, B)$, and $\{r_i\}$, $\{r_i^*\}$ are standard uniforms. Finally $N(\lambda\Delta)$ is a Poisson random variable of mean $\lambda\Delta\delta\gamma\kappa$. It was shown experimentally in [5] that the infinite series appearing in (4.10)-(4.11) are dominated by the first few terms, "although as κ goes to one this becomes less sharp".

Performing inference in this context is extremely challenging as, although it is possible to sample approximately from the prior, it is not possible to evaluate it pointwise. In [44] the authors have proposed an MCMC algorithm for Bayesian inference. This requires sampling the volatility process $\left\{\sigma_n^2\right\}$; this is achieved by updating $\{\eta_n\}$ one at a time with an independent MH sampler using the prior as proposal (and relying on an alternative infinite series expansion of η_n). In [21] an SMC algorithm was proposed to estimate $\left\{\sigma_n^2\right\}$ when the parameters of the models are fixed: it uses the prior as a proposal as any alternative proposal would require a pointwise evaluation of the prior.

We applied PMCMC to this model in order to perform Bayesian inference on the parameters $\{\kappa, \delta, \gamma, \lambda\}$ and we set here $\mu = \beta = 0$ as in [21]. First we ran PMMH on a synthetic data set and then used the daily Standard & Poor's 500 (S&P 500) closing prices from January 12, 2002 to December 30, 2005.

4.5.1 Synthetic Data

We generated a synthetic data set with $T = 400$ observations using parameters $\theta_{\text{true}} = (\kappa, \delta, \gamma, \lambda) = \big(0.5, \sqrt{2}, \sqrt{8}, 0.1\big)$.

The MH step in the PMMH used a Gaussian random walk proposal using the following covariance matrix Σ^{RW}, which was found by running the algorithm for some time using a diagonal covariance matrix and then computing the sample covariance:

$$\Sigma^{\text{RW}} = \begin{pmatrix} 0.00834501 & -0.0602253 & 0.0346668 & -0.00188407 \\ -0.0602253 & 0.694237 & 0.0921531 & 0.000170793 \\ 0.0346668 & 0.0921531 & 1.82894 & -0.121654 \\ -0.00188407 & 0.000170793 & -0.121654 & 0.199951 \end{pmatrix}$$

We found that the parameters κ and δ were quite correlated. To improve mixing, we alternated between the following two proposals:

$$\begin{aligned} q_1(\theta, \theta^*) &= N(\theta^*; \theta, \tfrac{1.5^2}{4}\Sigma^{\text{RW}}) \\ q_2(\theta_{1:2}, \theta^*_{1:2}) &= N(\theta^*_{1:2}; \theta_{1:2}, \tfrac{1.5^2}{2}\Sigma^{\text{RW}}_{1:2}), \quad \text{with } \theta^*_{3:4} = \theta_{3:4} \end{aligned}$$

The simulation is quite sensitive to κ, so we use a fairly informative prior. This was also observed in [44]. We used a Beta prior for κ and Gamma priors for the other three parameters:

$$\begin{aligned} \pi(\kappa) &= Be(10, 10) \\ \pi(\delta) &= Ga(1, 7) \\ \pi(\gamma) &= Ga(1, 14) \\ \pi(\lambda) &= Ga(1, 0.5) \end{aligned}$$

We ran simulations using using 50, 100, and 200 particles. The autocorrelation of the parameters is shown in Figure 4.15), the histograms and scatter plots of the parameters (using 200 particles) are given in Figure 4.16, and the trace is plots are shown in Figure 4.17. The posterior for κ is almost the same as the prior, suggesting that the prior is either too informative, or that there is very little information about κ in the data. We also tried a broader prior and found that the posterior is also similar to the prior, but the simulation does not mix as well, in particular when κ gets close to 1. The acceptance rate for the PMMH was 10%. We used dynamic

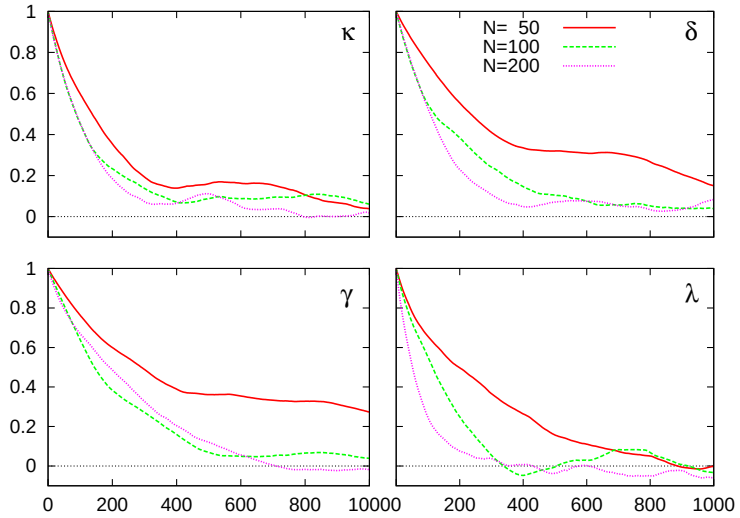


Figure 4.15: Lévy-driven SV model using synthetic data: Comparison of auto-correlation functions for different number of particles.

resampling in the SMC proposal and resampled on average at 12% of the time steps.

4.5.2 Standard & Poor's 500

The Standard & Poor's 500 (S&P 500) data set has 1000 observations, which are the daily increments in the log-prices from 12/01/2002 to 30/12/2005, and rescaled to have unit variance (the mean is 0.0079). The data is shown in Figure 4.18. Next we show simulations results using two different priors; the first one being a bit more informative than the second.

Prior 1

The MH step in the PMMH used a Gaussian random walk proposal using the following covariance matrix Σ^{RW}, which again was generated by running the algorithm for some time using a diagonal covariance matrix and then computing the

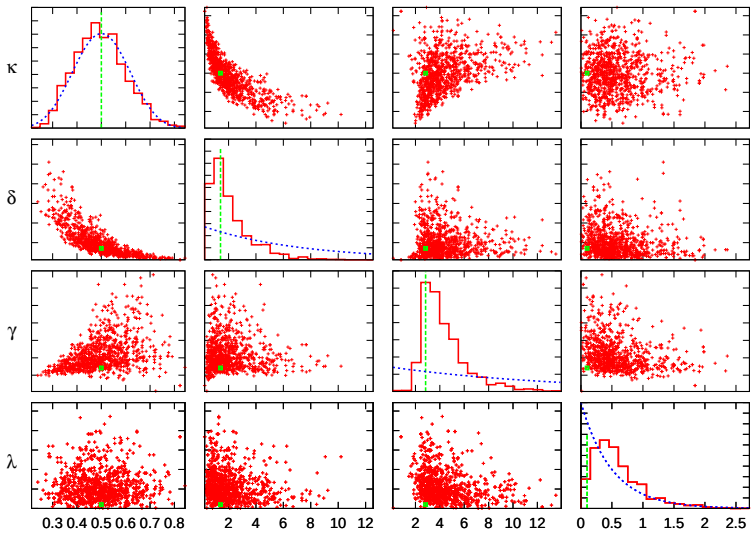


Figure 4.16: Lévy-driven SV model using synthetic data: Histogram and 2D scatter plots of sampled parameter values. The prior is shown as dashed curve and the true value is shown by the vertical dashed line: $\theta_{\text{true}} = (\kappa, \delta, \gamma, \lambda) = \left(0.5, \sqrt{2}, \sqrt{8}, 0.1\right)$. The data set has 400 observation and the simulation used 200 particles.

sample covariance:

$$\Sigma^{\text{RW}} = \begin{pmatrix} 0.00133842 & -0.104184 & 9.80768e-05 & 1.24963e-05 \\ -0.104184 & 16.3029 & 0.0881985 & -0.000947038 \\ 9.80768e-05 & 0.0881985 & 0.00670481 & 7.94049e-06 \\ 1.24963e-05 & -0.000947038 & 7.94049e-06 & 2.25697e-06 \end{pmatrix}$$

The same proposal was used as before (i.e. alternating between updating first two parameters, κ and δ, and updating all parameters jointly). The prior was chosen quite informative as follows:

$$\begin{aligned} \pi(\kappa) &= Be(4, 36) \\ \pi(\delta) &= Ga(1, 7) \\ \pi(\gamma) &= Ga(1, 14) \\ \pi(\lambda) &= Ga(1, 0.5) \end{aligned}$$

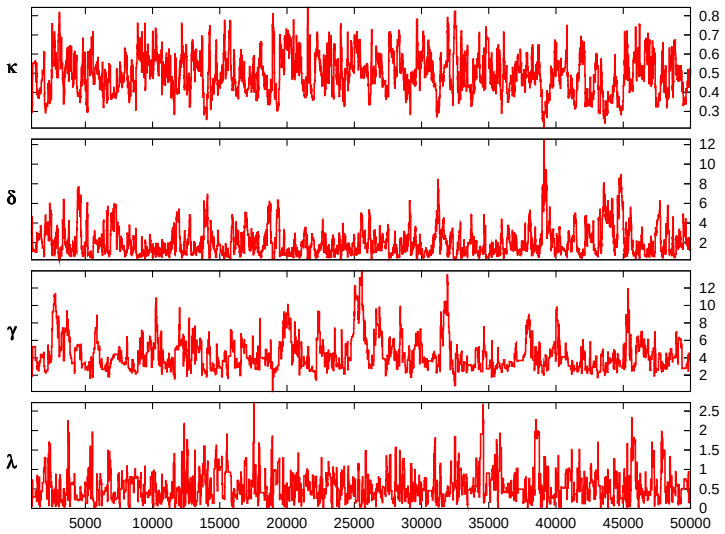


Figure 4.17: Lévy-driven SV model with synthetic data: Trace plots.

where $Be(\alpha, \beta)$ is the Beta distribution, and $Ga(k, \theta)$ is the Gamma distribution with shape k and scale θ. The histograms and scatter plots of the parameters are shown in Figure 4.19. As evident from the trace and auto-correlation plots given in Figure 4.21 and 4.20, the Markov chain mixes quite well given the complex structure of the posterior. The SMC proposal performs very well with a resample rate of only 4% and average ESS of about $0.75N$, where $N = 500$ is the number of particles used. The main difficulty lies is proposing good candidates for the parameters. The acceptance rate was 21%.

Prior 2

For this simulation we broadened the priors[1] as follows:

$$\begin{aligned} \pi(\kappa) &= Be(1, 15) \\ \pi(\delta) &= Ga(1, 20) \\ \pi(\gamma) &= Ga(1, 10) \\ \pi(\lambda) &= Ga(1, 1) \end{aligned}$$

[1]The prior for γ was tightened, but this change had negligible effect on the posterior

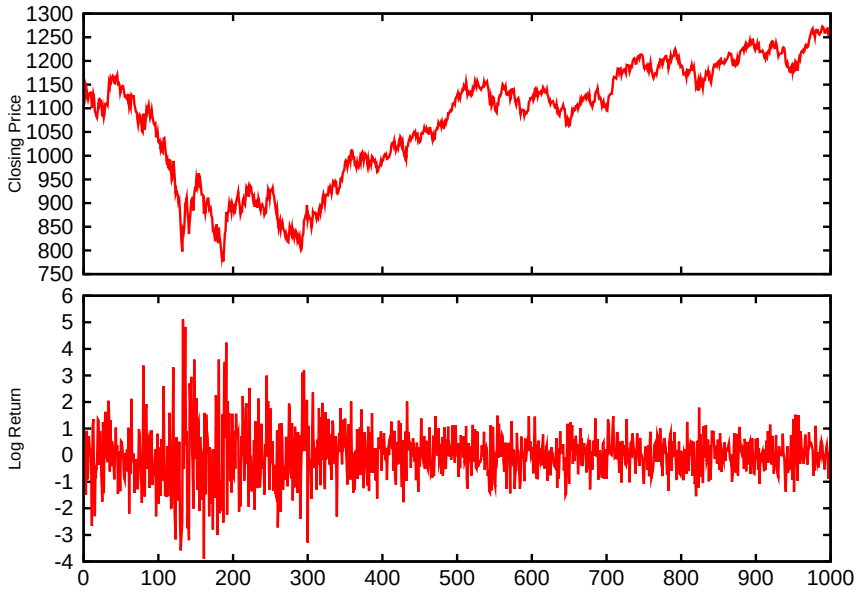


Figure 4.18: S&P 500 data set for Lévy-driven SV. Top: daily prices from from 12/01/2002 to 30/12/2005. Bottom: difference of log of prices, scaled to have unit variance.

The same covariance matrix as before was used for the random walk proposal, and we again alternated between updating the first two components and updating all parameters jointly. The acceptance rate was 23% with an average resampling rate of 4% and ESS=0.75N (again using $N = 500$ particles). The histograms and scatter plots of the samples are given in Figure 4.22. Figures 4.23 and 4.24 show the autocorrelation and trace, respectively.

Discussion

As expected from the simulations using the synthetic data, the simulation using the more informative prior 1 mixes much better than using prior 2. In both cases, we get similar results for the posterior of the parameters (see Table 4.5.2 for details).

With this application we showcased a model in which we cannot evaluate the prior, but can only sample from it. This forced us to propose from the prior, which

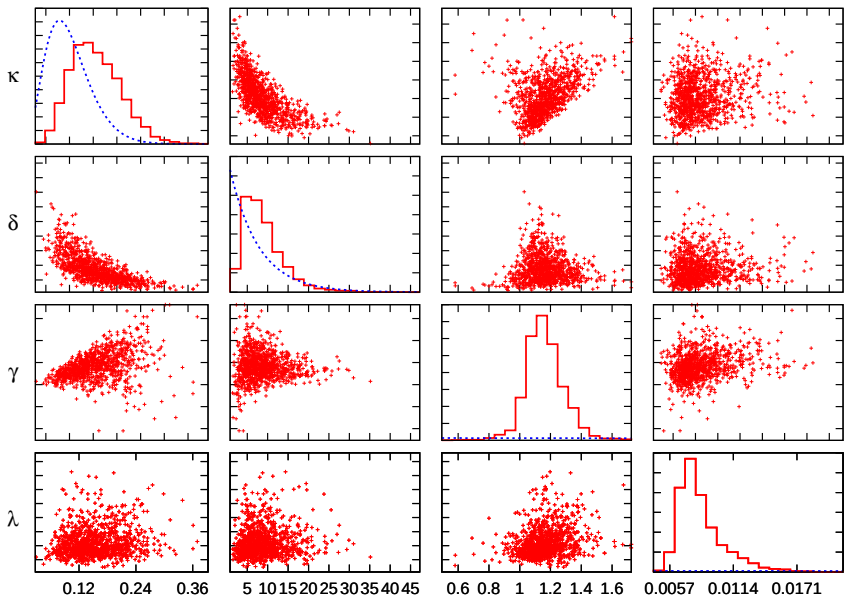


Figure 4.19: Lévy-driven SVs model using daily S&P 500 prices from 12/01/2002 to 30/12/2005: Histogram and 2D scatter plots of sampled parameter values. The prior is shown as the dashed curve. The data set has 1000 observation and the simulation used 500 particles.

can be very inefficient, especially when the observations are informative. However, using PMCMC we are still able to achieve good performance.

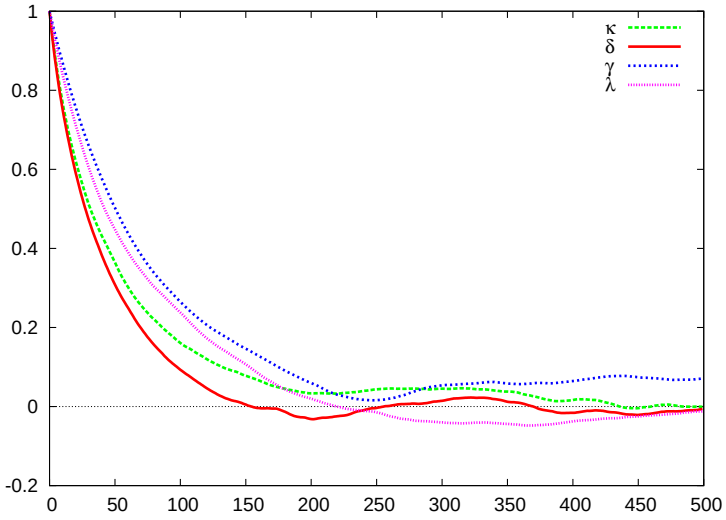


Figure 4.20: Lévy-driven SV model using daily S&P 500 prices from 12/01/2002 to 30/12/2005: autocorrelation function of sampled parameter values.

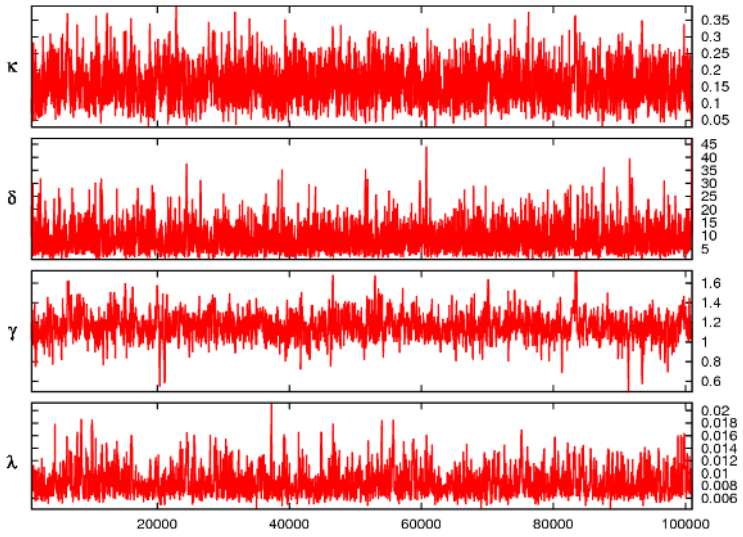


Figure 4.21: Lévy-driven SV model using daily S&P 500 prices from 12/01/2002 to 30/12/2005: Trace plots of sampled parameter values.

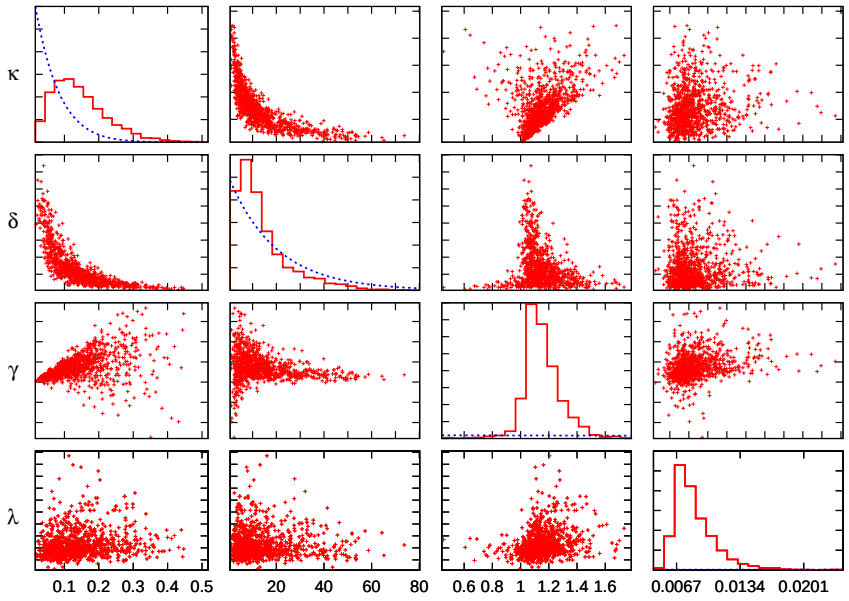


Figure 4.22: Lévy-driven SV model using daily S&P 500 prices from 12/01/2002 to 30/12/2005: Histogram and 2D scatter plots of sampled parameter values. The priors are shown as dashed curves and are $\pi(\kappa) = Be(1, 15)$, $\pi(\delta) = Ga(1, 20)$, $\pi(\gamma) = Ga(1, 10)$, $\pi(\lambda) = Ga(1, 1)$

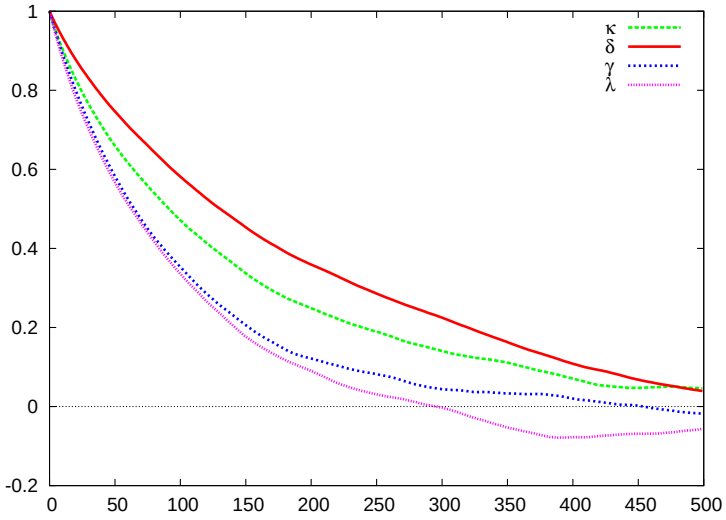


Figure 4.23: Lévy-driven SV model using daily S&P 500 prices from 12/01/2002 to 30/12/2005: autocorrelation function of sampled parameter values.

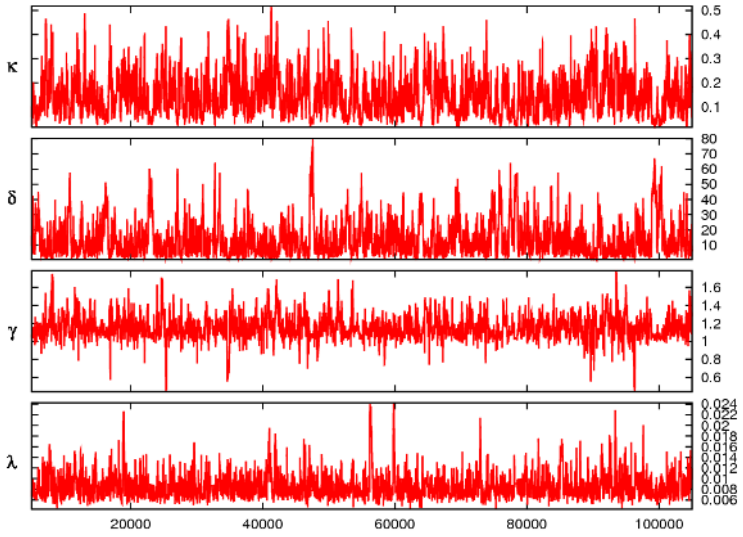


Figure 4.24: Lévy-driven SV model using daily S&P 500 prices from 12/01/2002 to 30/12/2005: Trace plots of sampled parameter values.

	Prior 1			
parameter	κ	δ	γ	λ
prior	$\mathcal{B}e(4, 36)$	$\mathcal{G}a(1, 7)$	$\mathcal{G}a(1, 14)$	$\mathcal{G}a(1, 0.5)$
mean	0.1555	8.939	1.163	0.008472
std	0.0524	5.034	0.126	0.002055
10th percentile	0.0923	3.877	1.027	0.006461
median	0.1502	7.883	1.151	0.007924
90th percentile	0.2260	15.254	1.322	0.011302
	Prior 2			
parameter	κ	δ	γ	λ
prior	$\mathcal{B}e(1, 15)$	$\mathcal{G}a(1, 20)$	$\mathcal{G}a(1, 10)$	$\mathcal{G}a(1, 1)$
mean	0.1452	14.004	1.155	0.008661
std	0.0794	11.523	0.134	0.002337
10th percentile	0.0555	3.630	1.030	0.006501
median	0.1301	10.443	1.136	0.008071
90th percentile	0.2555	30.212	1.326	0.011558

Table 4.7: Posterior statistics of parameters for S&P 500 data set.

4.6 Tempering

Following the idea outlined in [65, Sec. 4.2.2] we use particle Metropolis Hastings to sample from a multimodal target distribution. By using a sequence of tempered distributions we sample a chain of variables such that the marginal of the last variable in the chain is our target distribution. The sequence of distributions is built such that each two consecutive distributions are similar. An alternative method is *Tempered Transitions* by Neal [66], where the first distribution in the chain is the target distribution, and the following distributions are progressively easier to sample from (i.e. tempered) until the centre of the chain, at which point the distributions start approaching the target distribution, with the last distribution being the target. The method is shown in Algorithm 15.

Note that for this case it is actually preferable to use SMC, as in [65], instead of PMH. In fact the only potential benefit of using PMH is its iterative nature, which allows us run it for an unspecified amount of time, e.g. until some time limit or a sufficient number of samples have been generated. However, this can also be accomplished using the sequentially interacting Markov chain Monte Carlo (SIMCMC) sampler [10], though we have not yet made any comparison with this method.

Algorithm 4.2: Tempered Transitions

Initialise $X(s=0)$

At iteration $s \geq 1$

Set $X_0^* = X(s-1)$

For $n = 1, \ldots, 2p$: Sample $X_n^* \sim T_n(\cdot|X_{n-1}^*)$

With probability:

$$1 \wedge \prod_{j=0}^{2p} \frac{\pi_{n+1}(X_n)}{\pi_n(X_n)} \tag{4.12}$$

set $X(s) = X_{2p}^*$, otherwise set $X(s) = X(s-1)$

The target density is $\pi(\cdot) = \pi_0(\cdot)$ and the sequence of distributions $\pi_0, \ldots, \pi_{2p}$ with $\pi_{p-n} = \pi_{p+n}$, such that for $n \leq p$, $\pi_n(\cdot)$ is easier to sample from than π_{n-1}. We also define a sequence of transition kernels $T_1, \ldots, T_{2p}$, where $T_n(X, \cdot)$ has $\pi_m(\cdot)$ as an invariant distribution, with $m = n$ for $n \leq p$ and $m = 2p - n + 1$ for

$n > p$. The following reversibility condition must hold:

$$\pi_n(x)\, T_n(x, x') \;=\; \pi_n(x')\, T_{2p-n+1}(x', x), \quad \text{for } \; n < p \tag{4.13}$$

Note that we may choose T_n to be the same as T_{2p-n+1}.

Tempered Sampling with PMH and CBMC

For the PMH and CBMC samplers we only need the second half of the sequence defined above. In this context we label them $\bar{\pi}_1, \ldots, \bar{\pi}_p$, with $\bar{\pi}_p = \pi_0$. We also choose different transition kernels more suitable for the particle method. This gives rise to joint distributions as given in [65] using the artificial backwards transitions $L_n(x_{n+1}, x_n)$:

$$\tilde{\pi}_n(\mathbf{x}_{1:n}) = \tilde{\gamma}_n(\mathbf{x}_{1:n})/Z_n \tag{4.14}$$

where

$$\tilde{\gamma}_n(\mathbf{x}_{1:n}) = \gamma_n(x_n) \prod_{j=1}^{n-1} L_n(x_{n+1}, x_n) \tag{4.15}$$

and $\gamma_j(x_j) \propto \bar{\pi}_j(x_j)$. The transition kernels $T_n(x_n|\mathbf{x}_{n-1})$ have $\bar{\pi}_n$ as invariant distribution.

Using the second half this sequence of target distributions we can sample "chains" of variables from the joint target distribution and obtain the marginal by simply discarding all but the last variable in the chain.

Mixture of Gaussians

We used the above algorithm to sample from a highly multimodal distribution. The example was taken from [66]. The probability distribution function (PDF) was a mixture of over 4000 Gaussians in 2-D with means separated by many times their widths. Despite being low dimensional, this is a difficult distribution to sample from, using MCMC and pretending we only have access to $\pi_n(x, y)$ (Eqn. 4.16) pointwise and up to an unknown normalising constant, and serves well to demon-

strate tempering using PMH.

$$\begin{aligned}\pi_0(x,y) &= \sum_{i=-5}^{5}\sum_{j=-5}^{5}\exp\left(-\tfrac{|(x,y)-\mu_{1,i,j}|^2}{2\sigma^2})\right) \\ &+ \sum_{i=-5}^{5}\sum_{j=-5}^{5}\exp\left(-\tfrac{|(x,y)-\mu_{2,i,j}|^2}{2\sigma^2})\right) \\ &+ \sum_{i=-22}^{22}\sum_{j=-22}^{22}\exp\left(-\tfrac{|(x,y)-\mu_{3,i,j}|^2}{2\sigma^2})\right) \\ &+ \sum_{i=-22}^{22}\sum_{j=-22}^{22}\exp\left(-\tfrac{|(x,y)-\mu_{4,i,j}|^2}{2\sigma^2})\right)\end{aligned} \tag{4.16}$$

with $\sigma = 0.001$ and

$$\begin{aligned}\mu_{1,i,j} &= (0.0025i + 15, 0.0025j + 15) \\ \mu_{2,i,j} &= (0.1500i - 15, 0.1500j + 15) \\ \mu_{3,i,j} &= (0.0025i - 15, 0.0025j - 15) \\ \mu_{4,i,j} &= (0.1500i + 15, 0.1500j - 15)\end{aligned} \tag{4.17}$$

We then define a tempering schedule by choosing a sequence $\beta_1, \ldots, \beta_p$ with $\beta_p = 1$ and $0 < \beta_n < \beta_{n+1} \leq 1$. The distributions are then $\bar{\pi}_n = \pi_0^{\beta_n}$.

The performance of the PMH sampler for this example was compared with Neal's tempered transitions (NTT) and CBMC, the results are given in Table 4.8. Each algorithm was run 3 times for 8 hours (the results shown are averages). For each algorithm we chose 5 settings (number of particles and temperatures) that gave the best performance in terms of mean and standard deviation of the target distribution. PMH gives comparable performance as both CBMC and tempered transitions. Figure 4.25 shows the trace of the last 3000 samples for a PMH run using 5 particles and 20 temperatures.

As we can see, there is a tradeoff between the number of particles N, the number of distributions p, and the number of samples to generate, given a fixed computation budget. In this model the best setting, among the ones tried, was to use only a small number of particles ($N = 5$), a medium number of tempered distributions ($p = 20$), and opt for a larger number of samples. Unfortunately we have no theoretical result to tell us this "sweet" spot in advance and one should perform a few short simulations at various settings to determine a "good" setting using standard MCMC and SMC diagnostics to measure performance.

Alg.	*samples*	N	p	μ_x	var(x)	μ_y	var(y)	err(μ)	err(var)	*Acc*	*R*
PMH	78400	5	20	-0.0874	226.6	-13.31	49.16	0.0874	0.579	12	23
PMH	14200	5	100	-0.0918	226.8	-13.21	51.75	0.134	2.070	33	1
PMH	87600	10	10	0.310	226.8	-13.21	52.00	0.326	2.319	14	75
PMH	42600	20	10	0.407	226.7	-13.31	49.68	0.407	0.102	24	76
PMH	38000	10	20	-0.160	225.7	-13.11	54.93	0.257	5.362	22	36
CBMC	2932	20	50	0.0948	227.4	-13.33	48.41	0.0975	1.393	44	-
CBMC	6177	5	100	0.119	226.4	-13.33	47.83	0.121	1.884	37	-
CBMC	16021	10	20	0.0203	226.1	-13.43	47.72	0.119	2.093	31	-
CBMC	7836	20	20	0.0158	226.6	-13.37	47.08	0.0649	2.603	37	-
CBMC	33241	5	20	-0.0943	225.5	-13.14	54.34	0.192	4.823	20	-
Neal	27689	1	100	-0.121	227.0	-13.27	51.22	0.126	1.559	16	-
Neal	14206	1	200	0.227	228.7	-13.29	50.56	0.228	2.075	27	-
Neal	53000	1	50	-0.606	225.4	-13.45	43.33	0.621	6.509	64	-
Neal	145500	1	20	1.650	221.4	-13.58	43.66	1.672	8.091	0.72	-
Neal	300000	1	10	0.337	216.5	-12.27	72.30	1.089	24.86	0.07	-

Table 4.8: Simulation results for a mixture of Gaussians with 4292 modes. Each algorithm was run for the same amount of time (3 times 8 hrs) and used 10 MH steps at each temperature. N is the number of particles and p is the number of distributions (temperatures) used. *Acc* is the average acceptance rate and *R* is the resampling rate for PMH.

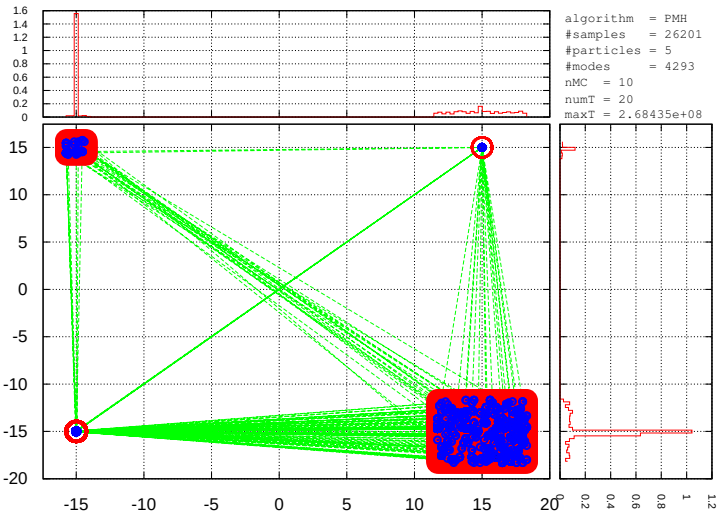


Figure 4.25: Trace and marginal posterior distributions for mixture of Gaussians distribution using PMH with 5 particles and numT=20 temperatures (length of sequence of distributions), a maximum temperature of 2^{28}=2.7$\times 10^8$, and 10 Metropolis-Hastings steps within each temperature. The large (bottom-left) frame shows the trace of the last 1000 iterations (lines and small blue circles) and the locations of the 4292 modes of the target distribution (larger red circles). The top and the right frames show the histograms of the marginals using 26201 samples.

Density Estimation using Finite Mixture of Gaussians

Given data $y_1, \ldots, y_c$, which are independent and identically distributed, we want to infer the distribution from which they arise. We model the distribution as a finite mixture of Normal distributions as in [65]:

$$y_i|\theta_r \;\sim\; \sum_{j=1}^{r} \omega_j \mathcal{N}(\mu_j, \lambda_j^{-1}) \tag{4.18}$$

where $\theta_r = (\mu_{1:r}, \lambda_{1:r}, \omega_{1:r})$, $2 \leq r < \infty$ and the number of Gaussians, r, is known. The weights ω_j must sum to one. The priors are exchangeable for each mixture

component:

$$\begin{aligned} \mu_j &\sim \mathcal{N}(\xi, \kappa^{-1}) && (4.19) \\ \lambda_j &\sim \mathcal{G}a(\nu, \chi), \quad \nu = \text{shape}, \ \chi = \text{scale} && (4.20) \\ \omega_{1:r-1} &\sim \mathcal{D}(\rho) && (4.21) \end{aligned}$$

The parameters ξ and κ are set to the midpoint of the data range R and a small multiple ι of $1/R^2$:

$$\xi = \frac{\min(y_i) + \max(y_i)}{2} \qquad \kappa = \frac{\iota}{[\max(y_i) - \min(y_i)]^2} \tag{4.22}$$

The parameters for the Gamma distribution are set as follows:

$$\begin{aligned} (2 =)\nu &> 1 > g = 0.2 \\ h &= 10\,[\max(y_i) - \min(y_i)]^{-2} \\ \chi &= h/g \end{aligned}$$

The Dirichlet prior has parameter $\rho = (1, \ldots, 1)$. The log prior is given by:

$$\log f(\theta_r) = \sum_{j=1}^{r} \log \left\{ \mathcal{N}(\mu_j | \xi, \kappa^{-1}) \ \mathcal{G}a(\lambda_j | \nu, \chi) \ \right\} + \log \mathcal{D}(\omega_{1:r} | \rho)$$

We define a sequence of distributions which starts with the prior and ends with the full posterior:

$$\pi_p(\theta_r) \ \propto \ l(y_{1:c} | \theta_r)^{\phi_p} \ f(\theta_r), \quad p = 1, \ldots, n \tag{4.23}$$

with $0 \leq \phi_1 < \ldots < \phi_n = 1$

We have compared the performance of PMH with NTT and with CBMC. The data set was taken from [65] and is shown in Figure 4.26. The acceptance rates for the three different algorithms are presented in Figure 4.27. The algorithm settings are chosen such that all have comparable computational complexity. PMH achieves at least double the acceptance rate over the other methods, using 20 particles and a sequence of 50 distributions (temperatures). The tempered transitions

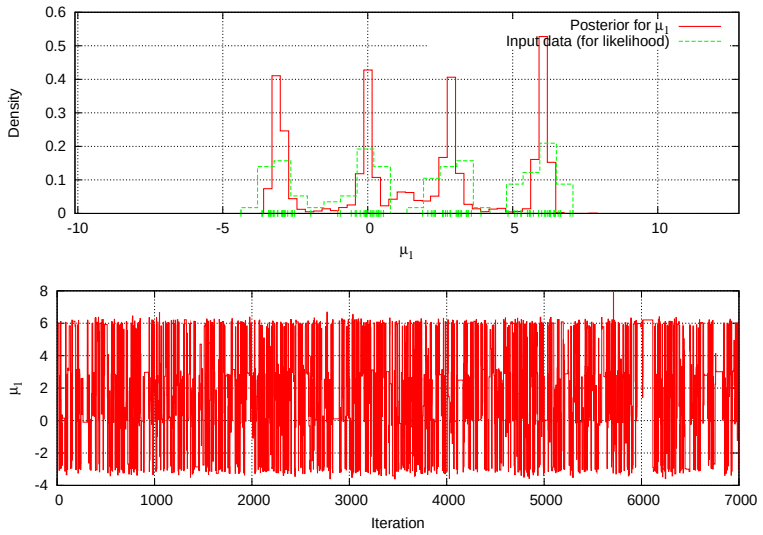


Figure 4.26: Data set for 4-component mixture of Gaussians (green) and posterior probability for parameter μ_1 (red) using PMH with $p = 50$ and 20 particles. The temperatures $(1/\beta_n)$ schedule was linear with a maximum temperature of 50. Bottom: trace of μ_1.

algorithm requires a much finer temperature scale with a sequence of 1000 distributions. CBMC fails to achieve acceptance rates of more than a few percent.

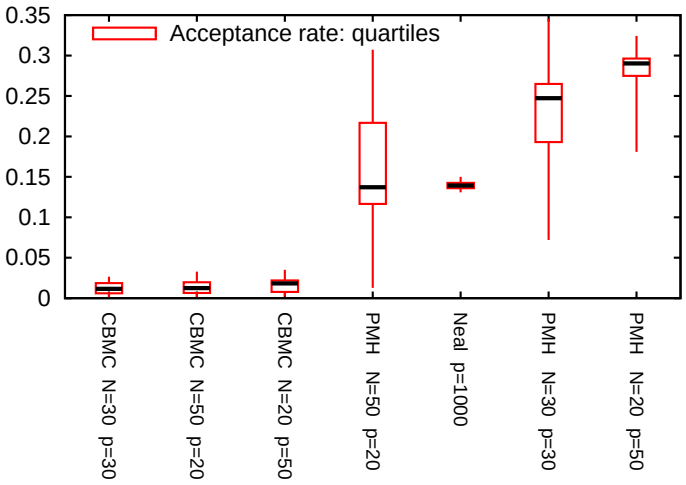


Figure 4.27: Acceptance rates for finite mixture model using PMH, CBMC, or NTT

Chapter 5

Conclusion

We have presented a new class of MCMC algorithms which rely on proposal distributions built using SMC methods. One of the major advantages of this approach is that it "automatically" builds efficient high-dimensional proposal distributions whilst requiring the practitioner to design only low-dimensional proposal distributions. It offers the possibility to simultaneously update large vectors of dependent random variables. This strategy is computationally expensive but to some extent unavoidable and useful in complex scenarios for which standard proposals are likely to fail. Indeed, as demonstrated in simulation, they can allow algorithms to better explore the posterior distribution in multimodal cases for example. In addition, in practice these algorithms can be combined with "standard" local moves which are computationally cheaper.

It is difficult to specify in advance when PMCMC will yield better performance than some alternative, possibly simpler, methods. However, we can take some lessons from the applications presented in this thesis and provide a few (rough) guidelines, as shown in Table 5.1. We only list the most commonly used algorithms, namely Gibbs, MH, and SMC, as well as the the methods introduced in this thesis. As a rule of thumb, the simpler methods should be tried first (unless it is clear that they will be inefficient). If they do not perform well, then implementing a PMCMC algorithm requires only little extra work, as it is based on MCMC and SMC, so part of the work would have already been done.

For models in which the prior cannot be evaluated, but only sampled from (e.g.

	$\mathbf{x}$ low dimensional		$\mathbf{x}$ high dimensional	
dependence	weak	strong	weak	strong
in $\mathbf{x}$	G or MH	MH	G or SMC	SMC or PMH
between θ and $\mathbf{x}$	G or MH	MH	PG	PMMH

Table 5.1: These are rough guidelines for algorithm choice. θ represents the parameters and $\mathbf{x}$ the states. G stands for Gibbs. PMH would be used for subblock updating.

MJP or Lévy-driven SV models with certain marginals), PMCMC may be the only feasible method.

We believe that many problems where SMC methods have already been successfully used could benefit from the particle MCMC methodology. These include contingency tables [15], graphical models [53], changepoint models [36], population dynamic models in ecology [12], volatility models in financial econometrics [16, 25], partially observed diffusions [37], population genetics [24],[61, Section 4.1.2], systems biology [48, 49], and experimental design problems [59]. The CBMC method, to which our approach is related, is a very popular method in computational chemistry and physics which has been widely used for molecular and polymer simulation [40], and particle MCMC algorithms might also prove a useful alternative in these areas. In this thesis we have addressed some of these topics to demonstrate the use of PMCMC. We are already aware of recent successful applications of PMCMC methods in econometrics [38] and statistics [7].

There are also numerous possible extensions to the PMCMC framework. Some have already been added, such as stratified sampling. Other resampling techniques could potentially also be added, for example antithetic sampling. In practice, the performance of the particle MCMC are closely related to the variance of the SMC estimates of the normalising constants. The SMC literature has many more techniques that could be used within PMCMC. For example we might investigate whether the SMC smoothing techniques in [47, 58] could be used to design better proposal distributions. We can also add adaptive strategies to set the algorithm parameters in PMCMC. It would nice to automatically adjust the number of particles used in the proposal to ensure a reasonable acceptance rate. In order to speed up the computation, a parallel implementation of the SMC component would also

be very useful. For example [63] and [54] have already demonstrated a parallel particle filter running on the GPU of a graphics card.

From a theoretical point of view, it is possible to study how "close" the particle MCMC algorithms are from the ideal MCMC algorithms they are approximating (corresponding to $N \to \infty$) using the techniques developed in [1].

Bibliography

[1] C. Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient computation. *Annals Statist.*, 2007. URL http://www.maths.bris.ac.uk/~maxca/preprints/andrieu_roberts_2006.pdf. in press. → pages 27, 44, 96, 107

[2] C. Andrieu, J. De Freitas, and A. Doucet. Sequential MCMC for bayesian model selection. In *Proceedings IEEE Workshop Higher Order Statistics*, pages 130–134, 1999. → pages 20

[3] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo. *Submitted to Journal of the Royal Statistical Society*, November 2007. → pages 1

[4] O. Barndorff-Nielsen and N. Shephard. Non-Gaussian Orstein-Uhlenbeck-based models and some of their uses in financial economics. *J. Royal Statist. Soc.*, 63:167–241, 2001. → pages 73

[5] O. Barndorff-Nielsen and N. Shephard. Normal modified stable processes. *Theory Proba. Math. Statist.*, 65:1–19, 2001. → pages 74, 75

[6] M. A. Beaumont. Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164:1139–1160, 2003. → pages 44

[7] M. Belmonte, O. Papaspiliopoulos, and M. Pitt. Particle filter estimation of duration-type models. Technical report, Department of Statistics, Warwick University, 2008. → pages 95

[8] P. Bourne and H. Weissig. *Structural Bioinformatics*. John Wiley & Sons Publishing, New York, 2003. → pages 53

[9] R. J. Boys, D. Wilkinson, and T. Kirkwood. Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and Computing*, 18

(2):125–135, 2008. ISSN 0960-3174. doi:10.1007/s11222-007-9043-x. → pages 67, 68, 69

[10] A. Brockwell, P. D. Moral, and A. Doucet. Sequentially interacting Markov chain Monte Carlo. *unpublished*, 2008. → pages 86

[11] T. Brunette and O. Brock. Improving protein structure prediction with model-based search. *Bioinformatics*, 21:i66–i74, 2005. doi:10.1093/bioinformatics/bti1029. → pages 53

[12] S. Buckland, K. Newman, C. Fernández, L. Thomas, and J. Harwood. Embedding population dynamic models in inference. *Statist. Sci.*, 22:44–5, 2007. → pages 2, 15, 95

[13] O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models*. Springer-Verlag, New York, 2005. → pages 15

[14] C. Carter and R. Kohn. On Gibbs sampling for state space models. *Biometrika*, 81:541–553, 1994. → pages 45

[15] Y. Chen, P. Diaconis, S. Holmes, and J. Liu. Sequential Monte Carlo methods for statistical analysis of tables. *J. Amer. Statist. Assoc.*, 100: 109–120, 2004. → pages 2, 3, 95

[16] S. Chib, M. K. Pitt, and N. Shephard. Likelihood based inference for diffusion driven state space models. *unpublished*, pages 1–33, 2006. URL http://www.olin.wustl.edu/faculty/chib/absdenew.htm. → pages 2, 95

[17] N. Chopin. A sequential particle filter method for static models. *Biometrika*, 89:539–552, 2002. → pages 3, 42

[18] N. Chopin. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *Annals Statist.*, 32:2385–2411, 2004. → pages 12

[19] N. Combe, T. J. H. Vlugt, P. R. T. Wolde, and D. Frenkel. Dynamic pruned-enriched Rosenbluth method. *Molecular Physics*, 101:1675–1682, 2003. → pages 15, 44

[20] S. Consta, N. B. Wilding, D. Frenkel, and Z. Alexandrowicz. Recoil growth: An efficient simulation method for multi-polymer systems. *Journal of Chemical Physics*, 110:3220–3228, 1999. → pages 15

[21] D. Creal. Analysis of filtering and smoothing algorithms for Lévy-driven stochastic volatility models. *Comp. Statist. Data Analy.*, 52:2863–2876, 2008. → pages 75

[22] T. Creighton. *Protein Folding*, pages 1–55. W.H. Freeman and Company, New York, USA, 1992. → pages 53

[23] D. Crisan and T. Lyons. Minimal entropy approximations and optimal algorithms for the filtering problem. *Monte Carlo Meth. Appli.*, 8:343–356, 2002. → pages 11

[24] M. De Iorio and R. Griffiths. Importance sampling on coalescent histories. *Adv. Appl. Proba.*, 36:417–433, 2004. → pages 95

[25] P. Dellaportas, D. Denison, and C. Holmes. Flexible threshold models for modelling interest rate volatiliy. *Econometrics Review*, 26:419–437, 2007. → pages 95

[26] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10:197–208, 2000. → pages 46

[27] A. Doucet, J. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001. → pages 2, 8, 46

[28] A. Doucet, S. J. Godsill, and C. P. Robert. Marginal maxiumum a posteriori estimation using Markov chain Monte Carlo. *Statistics and Computing*, 12: 77–84, 2002. → pages 36

[29] J. Durbin and S. Koopman. *Time Series Analysis by State Space Methods*. Oxford University Press, 2001. → pages 15

[30] M. Escobar and M. West. Bayesian density estimation and inference using mixtures. *J. Amer. Statist. Assoc.*, 90:577–588, 1995. → pages 62, 63, 114, 115

[31] P. Fearnhead. MCMC, sufficient statistics and particle filter. *J. Comp. Graph. Stat.*, 11:848–862, 2002. → pages 20

[32] P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Journal of Statistics and Computing*, 14:11–21, 2004. doi:10.1023/B:STCO.0000009418.04621.cd. → pages 2, 3, 42, 61, 62, 63

[33] P. Fearnhead. Computational methods for complex stochastic systems: Alternatives to MCMC. Technical report, Department of Mathematics and Statistics, Lancaster University, 2007. → pages 68

[34] P. Fearnhead. Online inference for multiple changepoint problems. *J. Royal Statist. Soc. B*, 69:589–605, 2007. → pages 42

[35] P. Fearnhead and J. R. Clifford. On-line inference for hidden Markov models via particle filters. *Statist. Soc. B*, 65:887–899, 2003. → pages 55

[36] P. Fearnhead and L. Meligkotsidou. Filtering methods for mixture models. *Journal of Computational & Graphical Statistics*, 16:586–607, September 2007. → pages 15, 95

[37] P. Fearnhead, O. Papaspiliopoulos, and G. Roberts. Particle filters for partially-observed diffusion. *J. Royal Statist. Soc. B*, 69:589–605, 2008. doi:10.1111/j.1467-9868.2008.00661.x. → pages 15, 95

[38] T. Flury and N. Shephard. Bayesian inference based only on simulated likelihood: particle filter analysis of dynamic economic models. Technical report, Nuffield College, Oxford University, 2008. → pages 95

[39] D. Frenkel. *Waste-recycling Monte Carlo*, pages 127–138. Springer-Verlag, New York, December 2006. ISBN 354035283X. URL http://www.springer.com/materials/book/978-3-540-35283-9. → pages 28

[40] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press (Elsevier), 2nd edition edition, 2002. → pages 15, 43, 46, 53, 95

[41] S. Frühwirth-Schnatter. *Finite Mixture and Markov Switching Models*. Springer Verlag, New York, 2006. → pages 15, 45

[42] S. Frühwirth-Schnatter and L. Sögner. Bayesian estimation of stochastic volatility models based on OU processes with marginal gamma law. *Ann. Instit. Statist. Math.* to appear. → pages 74

[43] S. G. Particle filters in state space models with the presence of unknown static parameters. *IEEE Trans. Signal Processing*, 50:281–289, 2002. → pages 20

[44] M. Gander and D. Stephens. Stochastic volatility modelling in continuous time with general marginal distributions: Inference, prediction and model selection. *J. Statist. Plann. Inf.*, 137:3068–3081, 2007. → pages 74, 75, 76

[45] W. Gilks and C. Berzuini. Following a moving target - Monte Carlo inference for dynamic Bayesian models. *J. Royal Statist. Soc. B*, 63: 127–146, 2001. → pages 3, 42

[46] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81:2340–2361, 1977. → pages 68

[47] S. Godsill, A. Doucet, and M. West. Monte Carlo smoothing for nonlinear time series. *J. Amer. Statist. Assoc.*, 99:156–168, 2004. → pages 46, 95

[48] A. Golightly and D. J. Wilkinson. Bayesian sequential inference for stochastic kinetic biochemical network models. *J. Comp. Biology*, 13: 838–851, 2006. → pages 2, 68, 95

[49] A. Golightly and D. J. Wilkinson. Bayesian sequential inference for nonlinear multivariate diffusions. *Statistics and Computing*, 16:323–338, 2006. → pages 67, 68, 95

[50] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE-Proceedings-F 140*, pages 107–113, 1993. → pages 15

[51] P. Grassberger. Pruned-enriched Rosenbluth method: Simulations of θ polymers of chain length up to 1 000 000. *Phys. Rev. E*, 56:3682 – 3693, 1997. → pages 15

[52] J. Griffin and M. Steel. Inference with non-Gaussian ornstein-unlhenbeck processes for stochastic volatility. *J. Econometrics*, 134:605–644, 2006. → pages 74

[53] F. Hamze and N. de Freitas. Hot coupling: a particle approach to inference and normalization on pairwise undirected graphs. In *Adv. Neural Info. Proc. Sys.*, volume 18. MIT Press, 2005. → pages 2, 3, 95

[54] G. Hendeby, J. D. Hol, R. Karlsson, and F. Gustafsson. A graphics processing unit implementation of the particle filter. In *European Signal Processing Conference (EUSIPCO)*, 2007. → pages 96

[55] H. Hsu, V. Mehra, W. Nadler, and P. Grassberger. Growth-based optimization algorithm for lattice heteropolymers. *Physical Review E*, 68(2): 21113, 2003. → pages 55

[56] C. Jarzynski. Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.*, 78:2690–2693, 1997. → pages 42

[57] G. Kitagawa. Monte Carlo filter and smoother for non-gaussian nonlinear state space models. *J. Comp. Graph. Statist.*, 5:1–25, 1996. → pages 11, 33, 46

[58] G. Kitagawa and S. Sato. *Monte Carlo smoothing and self-organising state-space models*, chapter 9. Springer-Verlag, New York, 2001. → pages 95

[59] H. Kueck, N. de Freitas, and A. Doucet. SMC samplers for Bayesian optimal nonlinear design. In *IEEE Nonlinear Statistical Signal Processing Workshop*, pages 99–102, Cambridge, UK, 13-15 Sept. 2006. ISBN 978-1-4244-0581-7. doi:10.1109/NSSPW.2006.4378829. → pages 95

[60] J. Liu, F. Liang, and W. Wong. The use of multiple-try method and local optimization in metropolis sampling. *J. Amer. Statist. Assoc.*, 95:121–134, 2000. → pages 44

[61] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer Verlag, New York, 1st edition edition, 2001. → pages 1, 2, 3, 8, 11, 41, 53, 54, 61, 95, 106

[62] K. Mengersen and R. Tweedie. Rates of convergence of the Hastings and Metropolis algorithms. *Annals Statist.*, 24:101–121, 1996. → pages 104

[63] A. S. Montemayor, J. J. Pantrigo, Ángel Sánchez, and F. Fernández. Particle filter on GPUs for real-time tracking. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Posters*, page 94, New York, NY, USA, 2004. ACM. ISBN 1-58113-896-2. doi:10.1145/1186415.1186526. → pages 96

[64] P. D. Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer-Verlag, New York, 2004. → pages 12, 13

[65] P. D. Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society B*, 68:411–436, 2006. → pages 2, 3, 42, 43, 86, 87, 90, 91

[66] R. M. Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6:353–366, 1996. → pages xi, 86, 87

[67] M. K. Pitt and N. Shephard. Filtering via simulation: auxiliary particle filter. *J. Am. Statist. Assoc.*, 94:590–599, 1999. → pages 2, 51

[68] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2nd edition edition, 2004. → pages 1, 5, 6, 54, 62

[69] G. Roberts, O. Papaspiliopoulos, and P. Dellaportas. Bayesian inference for non-Gaussian Orstein-Uhlenbeck stochastic volatility processes. *J. Royal Statist. Soc. B*, 66:369–393, 2004. → pages 74

[70] M. Rosenbluth and A. Rosenbluth. Monte Carlo simulations of the average extension of molecular chains. *J. Chem. Phys.*, 23:356–359, 1955. → pages 14

[71] N. Shephard and M. K. Pitt. Likelihood analysis of non-Gaussian measurement time series. *Biometrika*, 84:653–667, 1997. → pages 18, 46, 51

[72] A. Shmygelska and H. H. Hoos. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *Bioinformatics*, 6:30, 2005. → pages 53

[73] J. Siepmann. A method for the direct calculation of chemical potentials for dense chain systems. *Mol. Phys.*, 70:1145–1158, 1990. → pages 14

[74] J. Siepmann and D. Frenkel. Configurational-bias Monte Carlo: a new sampling scheme for flexible chains. *Mol. Phys.*, 75:59, 1992. → pages 43, 53

[75] C. Thachuk, A. Shmygelska, and H. H. Hoos. A replica exchange Monte Carlo algorithm for protein folding in the HP model. *Bioinformatics*, 8:342, 2007. → pages 53, 55, 59

[76] J. Zhang, S. Kou, and J. S. Liu. Biopolymer structure simulation and optimization via fragment regrowth Monte Carlo. *J. Chem. Phys.*, 126(22): 225101, 2007. → pages xi, 55, 60

Appendix A

Proofs

Proof of Theorem 2. Assumption (A2) ensures that q^N covers the support of $\tilde{\pi}^N$ and hence that the PMH defines an irreducible and aperiodic Markov chain with invariant distribution $\tilde{\pi}^N$ from Theorem 1. It follows that the marginal distribution of the sequence of random variables generated by the algorithm converges to $\tilde{\pi}^N$. Since $\mathbf{x}(i) = \mathbf{x}_p^{K(i)}(i)$ we have established the first result. Now under (A3) we have for all $\overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}$

$$\frac{\tilde{\pi}^N\left(k, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}\right)}{q^N(k, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1})} = \frac{\hat{Z}^N}{Z} < Z^{-1} \prod_{n=1}^{p} B_n < +\infty.$$

For an independent MH algorithm this implies uniform geometric ergodicity towards $\tilde{\pi}^N$, with a rate at least $1 - Z/\left(\prod_{n=1}^{p} B_n\right)$; see for example [62, Theorem 2.1]. This, together with a reasoning similar to above concerning $\mathbf{x}(i)$, we deduce the second statement of the theorem. ■

Proof of Theorem 3. To simplify notation, we note $\mathbf{z} = (\overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1})$. The transition probability of the PMH is of the form

$$P\left((k, \mathbf{z}), \left(\dot{k}, \dot{\mathbf{z}}\right)\right) = \dot{w}_p^{\dot{k}} \times \psi\left(\dot{\mathbf{z}}\right) \times \alpha\left(\mathbf{z}, \dot{\mathbf{z}}\right) + \left(\sum_{m=1}^{N} \int \check{w}_p^m \times \psi\left(\check{\mathbf{z}}\right) \times \left(1 - \alpha\left(\mathbf{z}, \check{\mathbf{z}}\right)\right) d\check{\mathbf{z}}\right) . \delta_{(k,\mathbf{z})}\left(\dot{k}, \dot{\mathbf{z}}\right) \tag{A.1}$$

where

$$\alpha\left(\mathbf{z},\dot{\mathbf{z}}\right)=1\wedge\frac{Z^{N}\left(\dot{\mathbf{z}}\right)}{Z^{N}\left(\mathbf{z}\right)}\,.$$

Note the important fact used below that the acceptance probability is independent of k and $\dot{k}$ (which, as pointed out earlier, means that $\dot{k}$ needs to be sampled only when a new set of particles has been accepted). Now consider the function

$$F\left(k,\mathbf{z}\right):=\sum_{l=1}^{N}f\left(\mathbf{x}_{p}^{l}\right)\mathbb{I}\left\{l=k\right\}\ ,$$

and note the fact that since by construction $\tilde{\pi}^{N}\left(k,\mathbf{z}\right)$ is invariant with respect to P,

$$\begin{aligned}\sum_{k,\dot{k}=1}^{N}\int\tilde{\pi}^{N}\left(k,\mathbf{z}\right)P\left(\left(k,\mathbf{z}\right),\left(\dot{k},\dot{\mathbf{z}}\right)\right)F\left(\dot{k},\dot{\mathbf{z}}\right)d\mathbf{z}d\dot{\mathbf{z}} \qquad (A.2)\\ =\quad\sum_{\dot{k}=1}^{N}\int\tilde{\pi}^{N}\left(\dot{k},\dot{\mathbf{z}}\right)F\left(\dot{k},\dot{\mathbf{z}}\right)d\dot{\mathbf{z}}\quad=\quad\mathbb{E}_{\pi}(f)\end{aligned}$$

Let $\mathbf{Z}\sim\tilde{\pi}^{N}$, $\dot{\mathbf{Z}}\sim\psi$ (defined in (3.2)) and U be a random variable uniformly distributed on $[0,1]$. Then, using (A.1), the following quantity is an estimate of the left hand side of (A.2)

$$\left(\sum_{\dot{k}=1}^{N}\dot{W}_{p}^{\dot{k}}f(\dot{\mathbf{Z}}_{p}^{\dot{k}})\right)\mathbb{I}\left\{U<\alpha\left(\mathbf{Z},\dot{\mathbf{Z}}\right)\right\}+\mathbb{I}\left\{U>\alpha\left(\mathbf{Z},\dot{\mathbf{Z}}\right)\right\}\sum_{k=1}^{N}F\left(k,\mathbf{Z}\right)\tilde{\pi}^{N}\left(k|\mathbf{Z}\right),\qquad(A.3)$$

and it can be checked using (3.4) that

$$\tilde{\pi}^{N}\left(\,k|\,\mathbf{z}\right)=\frac{\tilde{\pi}^{N}\left(k,\mathbf{z}\right)}{\tilde{\pi}^{N}\left(\mathbf{z}\right)}=w_{p}^{k}\,.$$

The result of the theorem follows straightforwardly from this and the fact that under (A1)-(A2), the simulated Markov chain is ergodic *i.e.* as $i\rightarrow\infty$, $\left(K(i),\mathbf{Z}(i)\right)\sim\tilde{\pi}^{N}\left(k,\mathbf{z}\right)$. ■

Proof of Theorem 4. To establish the result, we first rewrite (3.15) as

$$\tilde{\pi}^N\left(\theta, k, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}\right) = \frac{1}{N^p}\pi\left(\theta, \mathbf{x}_p^k\right) \prod_{i=1, i\neq i_1^k}^{N} M_1^\theta\left(x_1^i\right) \prod_{n=2}^{p} r\left(\mathbf{a}_{n-1}^{-i_n^k} \middle| \mathbf{w}_{n-1}, a_{n-1}^{i_n^k}\right) \prod_{i=1, i\neq i_n^k}^{N} M_n^\theta\left(\mathbf{x}_{n-1}^{a_{n-1}^i}, x_n^i\right) \tag{A.4}$$

Notice that with $i^k = (i_1^k, i_2^k, \ldots, i_p^k)$ the ancestral lineage of $\mathbf{x}_p^k$, we have

$$\tilde{\pi}^N\left(\theta, \mathbf{x}_p^k, i^k\right) = \frac{1}{N^p}\pi\left(\theta, \mathbf{x}_p^k\right) .$$

Now the PG sampler can be interpreted as a standard Gibbs sampler of invariant distribution (3.15) which iterates the following steps

(a) $\theta|(\mathbf{x}_p^k, i^k) \sim \tilde{\pi}^N\left(\cdot|k, \mathbf{x}_p^k, i^k\right) = \pi\left(\theta|\mathbf{x}_p^k\right)$

(b) $(\overline{\mathbf{x}}_1^{-i_1^k}, ..., \overline{\mathbf{x}}_p^{-i_p^k}, \mathbf{a}_1^{-i_2^k}, \ldots, \mathbf{a}_{p-1}^{-i_p^k}) \sim \tilde{\pi}^N\left(\cdot|\theta, k, \mathbf{x}_p^k, i^k\right)$

(c) $k|(\theta, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}) \sim \tilde{\pi}^N(\cdot|\theta, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1})$

with $\tilde{\pi}^N(k|\theta, \overline{\mathbf{x}}_1, ..., \overline{\mathbf{x}}_p, \mathbf{a}_1, \ldots, \mathbf{a}_{p-1}) = w_p^k$. Note that (**a**) might appear unusual but leaves (3.15) invariant and is known in the literature under the name "collapsed Gibbs sampling" [61, Section 6.7].

Let $A \in \mathcal{B}(\Theta), B \in \mathcal{B}(\mathcal{X}^p)$, $C \in \mathcal{B}(\mathcal{X}^{(N-1)p} \times \{1, \ldots, N\}^{(N-1)p})$, $k \in \{1, \ldots, N\}$ and $i \in \{1, \ldots, N\}^p$ be such that $\tilde{\pi}^N(\{k\} \times A \times B \times \{i\} \times C) > 0$.

From (A5) it is possible to show that accessible sets for the G sampler are also marginally accessible by the PG sampler *i.e.* more precisely if $A \times B \in \mathcal{B}(\Theta) \times \mathcal{X}^p$ is such that $\mathcal{L}_G\left((\theta(j), \mathbf{x}(j)) \in A \times B\right) > 0$ for some finite $j > 0$ then also $\mathcal{L}_{PG}\left((K(j), \theta(j), \mathbf{x}(j), I(j)) \in \{k\} \times A \times B \times \{i\}\right) > 0$ for all $k \in \{1, \ldots, N\}$ and $i \in \{1, \ldots, N\}^p$. From this and the assumed irreducibility of the G sampler in (A6), we deduce that if $\pi((\theta, \mathbf{x}) \in A \times B) > 0$ then there exists a finite j such that $\mathcal{L}_{PG}\left((K(j), \theta(j), \mathbf{x}(j), I(j)) \in \{k\} \times A \times B \times \{i\}\right) > 0$ for all $k \in \{1, \ldots, N\}$ and $i \in \{1, \ldots, N\}^p$. Now because $\pi((\theta, \mathbf{x}) \in A \times B) > 0$ and step (**b**) corresponds to sampling from the conditional distribution of $\tilde{\pi}^N$, we

deduce that

$$\mathcal{L}_{PG}\left((K(j+1),\theta(j+1),\mathbf{x}\left(j+1\right),I(j+1),\ \overline{\mathbf{x}}_1^{-I_1^{K(j+1)}}(j+1),...,\overline{\mathbf{x}}_p^{-I_p^{K(j+1)}}(j+1),\right.$$
$$\left.\mathbf{A}_1^{-I_2^{K(j+1)}}(j+1),\ldots,\mathbf{A}_{p-1}^{-I_p^{K(j+1)}}(j+1)\right)\in\{k\}\times A\times B\times\{i\}\times C\Bigg)>0$$

and the irreducibility of the PG sampler follows. Aperiodicity can be proved by contradiction. Indeed from (A5) we deduce that if the PG sampler is periodic, then so is the G sampler, which contradicts (A6). ■

Proof of Theorem 5. The proof of the first part of the theorem is similar to the proof of Theorem 1 and is shown below. The second part of the proof is a direct consequence of [1, Theorem 1] and (A5)-(A7).

$$\begin{aligned}\frac{\tilde{\pi}^N(\cdot)}{q^N(\cdot)}&=\frac{1}{N^p}\frac{\pi\left(\theta,\mathbf{x}_p^k\right)}{q\left(\theta^*,\theta\right)\times w_p^k\times M_1^\theta(x_1^{i_1^k})\prod_{n=2}^p r(i_{n-1}^k|\mathbf{w}_{n-1})M_n^\theta(\mathbf{x}_{n-1}^{i_{n-1}^k},x_n^{i_n^k})}\\&=\frac{\pi\left(\theta,\mathbf{x}_p^k\right)}{q\left(\theta^*,\theta\right)M_1^\theta(x_1^{i_1^k})\prod_{n=2}^p M_n^\theta(\mathbf{x}_{n-1}^{i_{n-1}^k},x_n^{i_n^k})}\frac{\frac{1}{N^p}}{\prod_{n=1}^p w_n^{i_n^k}}\\&=\frac{\pi\left(\theta,\mathbf{x}_p^k\right)}{q\left(\theta^*,\theta\right)M_1^\theta(x_1^{i_1^k})\prod_{n=2}^p M_n^\theta(\mathbf{x}_{n-1,n}^{i_{n-1}^k},x_n^{i_n^k})}\frac{\frac{1}{N^p}\prod_{n=1}^p\sum_{i=1}^N w_n\left(\mathbf{x}_n^i\right)}{\prod_{n=1}^p w_n(\mathbf{x}_n^{i_n^k})}\\&=\frac{\cancel{\gamma\left(\theta,\mathbf{x}_p^k\right)}/Z}{q\left(\theta^*,\theta\right)}\frac{\widehat{\gamma}^N\left(\theta\right)}{\cancel{\gamma\left(\theta,\mathbf{x}_p^k\right)}}\quad=\quad\frac{1}{Z}\frac{\widehat{\gamma}^N(\theta)}{q\left(\theta^*,\theta\right)}\end{aligned}$$

where $\widehat{\gamma}^N\left(\theta\right)=\widehat{Z}^N$ is given in (2.6). In the manipulations above we have used (A1) on the second line, whereas the final result is obtained thanks to the definitions of the incremental weights (2.2)

$$w_1\left(\mathbf{x}_1^i\right):=\frac{\gamma_1(\theta,\mathbf{x}_1^i)}{M_1^\theta(\mathbf{x}_1^i)},\quad w_n\left(\mathbf{x}_n^i\right):=\frac{\gamma_n\left(\theta,\mathbf{x}_n^i\right)}{\gamma_{n-1}\left(\theta,\mathbf{x}_{n-1}^{A_{n-1}^i}\right)M_n^\theta\left(\mathbf{x}_{n-1}^{A_{n-1}^i},X_n^i\right)},$$

and of the normalising constant estimate (2.6). ■

Appendix B

Nonlinear State-Space Model

B.1 Gibbs Sampling for State-Space Model

We want to sample the time series using Gibbs sampling. Thus we need the conditional probabilities $\pi(\theta|\mathbf{x}_{1:T})$ and $\pi(x_j|\mathbf{x}_{-j}, \mathbf{y}_{1:T})$.

B.1.1 Sampling Parameters

For now, assume that we are interested in the variance of the noise in the state transition, i.e. σ_V^2.

$$\begin{aligned}
\pi(\sigma_V^2|\mathbf{x}_{1:T}) &\propto \pi(\mathbf{x}_{1:T}|\sigma_V^2)\,\pi(\sigma_V^2) \\
&= \pi(\sigma_V^2)\,\pi(x_1)\prod_{j=2}^{T}\frac{1}{\sqrt{2\pi\sigma_V^2}}\exp\left\{-\frac{[x_j-f(x_{j-1})]^2}{2\sigma_V^2}\right\} \\
&\propto \pi(\sigma_V^2)\,\sigma_V^{-(T-1)}\exp\left\{-\left(\tfrac{1}{2}\textstyle\sum_{j=2}^{T}[x_j-f(x_{j-1})]^2\right)\sigma_V^{-2}\right\} \\
&= \pi(\sigma_V^2)\,\sigma_V^{-(T-1)}\,e^{-\beta_V\sigma_V^{-2}}
\end{aligned}$$

where we set $\beta_V = \frac{1}{2}\sum_{j=2}^{T}[x_j - f(x_{j-1})]^2$. Setting $\kappa_V = \sigma_V^{-2}$ we get:

$$\pi(\kappa_V|\mathbf{x}_{1:T}) \propto \pi(\kappa_V)\,\kappa_V^{(T-1)/2}\,e^{-\beta_V\kappa_V} = \pi(\kappa_V)\,\kappa_V^{(T-1)/2}\,e^{-\beta_V\kappa_V}$$

Now we need to choose a prior for κ_V. A convenient one is the Gamma distribution

with shape ξ_v and scale γ_v:

$$\pi(\kappa_V|\mathbf{x}_{1:T}) \propto \kappa_V^{(T-1)/2+\xi_V-1} e^{-\left(\beta_V+\frac{1}{\gamma_v}\right)\kappa_V} \tag{B.1}$$

$$\propto \mathcal{G}a\left(\kappa_V\,;\,\frac{T-1}{2}+\xi_V,\,\left(\beta_V+\frac{1}{\gamma_V}\right)^{-1}\right) \tag{B.2}$$

Note that this implies

$$\pi(\sigma_W^2) = \mathcal{IG}a(\xi_v, 1/\gamma_v)$$

Similarly, for the variance of the observation error we get:

$$\pi(\sigma_W^2|\mathbf{x}_{1:T}) \propto \pi(\mathbf{x}_{1:T}|\sigma_V^2, \mathbf{y}_{1:T})\,\pi(\sigma_W^2) \tag{B.3}$$

$$= \pi(\sigma_W^2)\,\pi(x_0)\prod_{j=1}^{T}\frac{1}{\sqrt{2\pi\sigma_W^2}}\exp\left\{-\frac{[y_j-g(x_j)]^2}{2\sigma_W^2}\right\} \tag{B.4}$$

$$\propto \pi(\sigma_W^2)\,\sigma_W^{-T}\,e^{-\beta_W\sigma_W^{-2}} \tag{B.5}$$

where $\beta_W = \frac{1}{2}\sum_{j=1}^{T}[y_j - g(x_j)]^2$. Using the same approach as above, we sample $\kappa_W = \sigma_W^{-2}$ using a Gamma prior with shape ξ_W and scale γ_W:

$$\pi(\kappa_W|\mathbf{x}_{1:T}) \propto \kappa_W^{T/2+\xi_W-1} e^{-\left(\beta_W+\frac{1}{\gamma_W}\right)\kappa_W}$$

$$\propto \mathcal{G}a\left(\kappa_W\,;\,\frac{T}{2}+\xi_W,\,\left(\beta_W+\frac{1}{\gamma_W}\right)^{-1}\right) \tag{B.6}$$

B.1.2 Sampling State Variables

The conditional distribution for the latent variables x_i is as follows:

$$\pi(x_j|x_{-j}, \mathbf{y}_{1:T}) = \begin{cases} \pi(x_{j+1}|x_j)\,\pi(x_j|y_j), & \text{for } j=1 \\ \pi(x_j|x_{j-1})\,\pi(x_{j+1}|x_j)\pi(x_j|y_j), & \text{for } 1<j<T \\ \pi(x_j|x_{j-1})\,\pi(x_j|y_j), & \text{for } j=T \end{cases}$$

with

$$\pi(x_j|x_{j-1}) = \mathcal{N}\big(x_j;\, f(x_{j-1}), \sigma_V^2\big)$$
$$\pi(x_j|y_j) = \mathcal{N}\big(y_j;\, g(x_j), \sigma_W^2\big)$$

In general we cannot sample from this directly (except for some simple functions f and g). We therefore use a few MH steps within the Gibbs sampler. The proposal for this is the same as for the PMCMC (i.e. proposing from prior or using the extended Kalman filter (EKF) proposal).

Appendix C

Dirichlet Mixture Model: Derivation

C.1 Gaussian Mixture

The posterior is proportional to the prior times the likelihood:

$$\mathbb{P}(x_{1:n}|y_{1:n},\alpha) \;\propto\; \mathbb{P}(x_{1:n}|\alpha)\,\pi(y_{1:n}|x_{1:n},\alpha) \;=\; \mathbb{P}(x_{1:n}|\alpha)\,\pi(y_{1:n}|x_{1:n}) \quad \text{(C.1)}$$

The prior $\mathbb{P}(x_{1:n}|\alpha)$ is given by equation 4.4

$$\mathbb{P}(x_{1:n}|\alpha) = \prod_{i=1}^{n} \mathbb{P}(x_i|x_{1:i-1},\alpha) \quad = \quad \frac{\alpha^{k_n}\prod_{j=1}^{k_n}(m_n^j-1)!}{\alpha\,(\alpha+1)...(\alpha+n-1)} \quad \text{(C.2)}$$

The likelihood factorises since the observations are independent given the cluster:

$$\begin{aligned}\pi(y_{1:n}|x_{1:n},\alpha) = \prod_{j=1}^{k_n} \pi(y_{\{s|x_s=j\}}|x_{1:n},\alpha) \;&=\; \prod_{j=1}^{k_n}\int \pi(y_{\{s|x_s=j\}},\theta_j|x_{1:n})\,d\theta_j \\ = \prod_{j=1}^{k_n}\int \left(\prod_{\{i|x_i=j\}} \pi(y_i|\theta_j)\right)\pi(\theta_j)\,d\theta_j \;&=\; \prod_{j=1}^{k_n}\int \left(\prod_{\{i|x_i=j\}} g_{\theta_j}(y_i)\right) d\mathbb{G}_0(\theta_j)\end{aligned} \quad \text{(C.3)}$$

We can now substitute for g_{θ_j} and $\mathbb{G}_0$:

$$
\begin{aligned}
&\pi(y_{1:n}|x_{1:n},\alpha) \\
&\quad= \prod_{j=1}^{k_n} \int \left(\prod_{\{i|x_i=j\}} \mathcal{N}(y_i;\, \mu_j, \sigma_j^2) \right) \mathcal{IG}a(\sigma_j^2;\, a, b)\, \mathcal{N}(\mu_j;\, \eta, \tau\sigma_j^2)\, d\mu_j\, d(\sigma_j^2) \\
&\quad= \prod_{j=1}^{k_n} \int \left(\prod_{\{i|x_i=j\}} \frac{\exp\left\{-\frac{(y_i-\mu_j)^2}{2\sigma_j^2}\right\}}{\sqrt{2\pi\sigma_j^2}} \right) \\
&\qquad\qquad \times \left(\frac{b^a \exp\left\{\frac{-b}{\sigma_j^2}\right\}}{\Gamma(a)\left(\sigma_j^2\right)^{a+1}} \right) \left(\frac{\exp\left\{-\frac{(\mu_j-\eta)^2}{2\tau\sigma_j^2}\right\}}{\sqrt{2\pi\tau\sigma_j^2}} \right) d\mu_j\, d(\sigma_j^2) \\
&\quad= \prod_{j=1}^{k_n} \left(\frac{b^a}{\Gamma(a)} \right) \int \frac{\exp\left\{-\frac{\sum_{\{i|x_i=j\}}(y_i-\mu_j)^2}{2\sigma_j^2} - \frac{b}{\sigma_j^2} - \frac{(\mu_j-\eta)^2}{2\tau\sigma_j^2}\right\}}{\left(2\pi\sigma_j^2\right)^{m_n^j/2}\left(\sigma_j^2\right)^{a+1}\left(2\pi\tau\sigma_j^2\right)^{1/2}}\, d\mu_j\, d(\sigma_j^2) \\
&\quad\propto \prod_{j=1}^{k_n} \frac{b^a}{\Gamma(a)} \int \frac{\left(\sigma_j^2\right)^{-(1+a+\frac{m_n^j+1}{2})}}{\sqrt{\tau}} \\
&\qquad\qquad \times \exp\left\{ -\frac{1}{\sigma_j^2}\left(b + \frac{1}{2}\sum_{\{i|x_i=j\}} (y_i-\mu_j)^2 + \frac{1}{2\tau}(\mu_j-\eta)^2 \right) \right\} d\mu_j\, d(\sigma_j^2)
\end{aligned}
$$

We now have to complete the square in order to integrate out μ_j:

$$
\begin{aligned}
\Phi_n^j &= \sum_{\{i|x_i=j\}} (y_i-\mu_j)^2 + \frac{1}{\tau}(\mu_j-\eta)^2 \\
&= \mu_j^2\left(m_n^j + \frac{1}{\tau}\right) - 2\mu_j\left[\left(\sum_{\{i|x_i=j\}} y_i\right) - \frac{\eta}{\tau}\right] + \left(\sum_{\{i|x_i=j\}} y_i^2\right) + \frac{\eta^2}{\tau}
\end{aligned}
$$

Now define the first and second moment:

$$\tilde{y}^j_{n,1} = \frac{1}{m^j_n} \sum_{\{i|x_i=j\}} y_i \equiv \bar{y}^j_n \tag{C.4}$$

$$\tilde{y}^j_{n,2} = \frac{1}{m^j_n} \sum_{\{i|x_i=j\}} y_i^2 \equiv (\hat{\sigma}^j_n)^2 + (\bar{y}^j_n)^2 \tag{C.5}$$

We continue to complete the square and further simplify the expression Φ^j_n:

$$\begin{aligned}
\Phi^j_n &= \mu_j^2 \left(m^j_n + \frac{1}{\tau} \right) - 2\mu_j \left(m^j_n \tilde{y}^j_{n,1} - \frac{\eta}{\tau} \right) + m^j_n \tilde{y}^j_{n,2} + \frac{\eta^2}{\tau} \\
&= \left(m^j_n + \frac{1}{\tau} \right) \left(\mu_j - \frac{m^j_n \tilde{y}^j_{n,1} - \eta/\tau}{m^j_n + \frac{1}{\tau}} \right)^2 \\
&\quad + m^j_n \left[\left(\tilde{y}^j_{n,2} - (\tilde{y}^j_{n,1})^2 \right) + \frac{(\tilde{y}^j_{n,1} - \eta)^2}{1 + m^j_n \tau} \right]
\end{aligned}$$

Now we end up with an expression that has the form of a Gaussian in μ_j: If we now reinsert this into the above exponential, we can easily marginalise out μ_j.

$$\begin{aligned}
\pi(y_{1:n}|x_{1:n}, \alpha) \propto \prod_{j=1}^{k_n} \frac{b^a}{\Gamma(a)\sqrt{\tau}} \int \left(\sigma_j^2\right)^{-(1+a+\frac{m^j_n+1}{2})} \\
\times \exp\left\{ -\frac{1}{\sigma_j^2} \left(b + \frac{m^j_n}{2} \left[\left(\tilde{y}^j_{n,2} - (\tilde{y}^j_{n,1})^2 \right) + \frac{(\tilde{y}^j_{n,1} - \eta)^2}{1 + m^j_n \tau} \right] \right) \right\} \\
\times \exp\left\{ -\frac{1}{2\sigma_j^2} \left(m^j_n + \frac{1}{\tau} \right) \left(\mu_j - \frac{m^j_n \tilde{y}^j_{n,1} - \eta/\tau}{m^j_n + \frac{1}{\tau}} \right)^2 \right\} d\mu_j \, d(\sigma_j^2)
\end{aligned}$$

For ease of notation, we define

$$\Psi^j_n = \left(b + \frac{m^j_n}{2} \left[\left(\tilde{y}^j_{n,2} - (\tilde{y}^j_{n,1})^2 \right) + \frac{(\tilde{y}^j_{n,1} - \eta)^2}{1 + m^j_n \tau} \right] \right) \tag{C.6}$$

The integral over μ_j simply evaluates to the normalising constant of a Gaussian:

$$\begin{aligned}\pi(y_{1:n}|x_{1:n},\alpha) &\propto \prod_{j=1}^{k_n} \frac{b^a}{\Gamma(a)\sqrt{\tau}} \int \left(\sigma_j^2\right)^{-(1+a+\frac{m_n^j+1}{2})} \sqrt{\frac{2\pi\sigma_j^2}{m_n^j+\frac{1}{\tau}}} e^{-\Psi_n^j/\sigma_j^2}\, d(\sigma_j^2) \\ &= \prod_{j=1}^{k_n} \frac{b^a}{\Gamma(a)\sqrt{\tau}} \sqrt{\frac{2\pi}{m_n^j+\frac{1}{\tau}}} \int \left(\sigma_j^2\right)^{-(1+a+\frac{m_n^j}{2})} e^{-\Psi_n^j/\sigma_j^2}\, d(\sigma_j^2)\end{aligned}$$

This integral has the form of the Inverse-Gamma distribution, thus we can solve it analytically and the result is the normalising constant:

$$\int x^{-(\alpha+1)}\, \exp\left(\frac{-\beta}{x}\right)\, dx = \Gamma(\alpha)\, \beta^{-\alpha} \tag{C.7}$$

Thus we get

$$\begin{aligned}\pi(y_{1:n}|x_{1:n},\alpha) &\propto \prod_{j=1}^{k_n} \left(\frac{b^a}{\Gamma(a)\sqrt{\tau}} \sqrt{\frac{2\pi}{m_n^j+\frac{1}{\tau}}}\right) \Gamma\left(a+\frac{m_n^j}{2}\right)\, (\Psi_n^j)^{-(a+\frac{m_n^j}{2})} \\ &\propto \prod_{j=1}^{k_n} \left\{ \frac{b^a\, \Gamma\left(a+m_n^j/2\right)}{\Gamma(a)\sqrt{m_n^j\tau+1}} \left(b+\frac{m_n^j}{2}\left[\left(\hat{\sigma}_n^j\right)^2+\frac{(\bar{y}_n^j-\eta)^2}{1+m_n^j\tau}\right]\right)^{-(a+\frac{m_n^j}{2})} \right\}\end{aligned} \tag{C.8}$$

The derivation for sampling α conditional on the current state (number of clusters) is outlined in [30]. We repeat it here for convenience:

We first determine the probability distribution of the number of clusters given α and the number of data points.

$$\mathbb{P}(k_n|\alpha,n) = c_n(k_n)\, n!\, \alpha^{k_n} \frac{\Gamma(\alpha)}{\Gamma(\alpha+n)} \tag{C.9}$$

where $c_n(k_n) = \mathbb{P}(k_n|\alpha=1,n)$. It is possible to write the above fraction of Gamma functions in terms of a Beta function, which has the following definition

and property:

$$\beta(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}dt = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$$

Thus rewriting this and inserting it into the equation of $\mathbb{P}(k_n|\alpha,n)$ gives (where we let $x = \alpha+1$, $y = n$):

$$\begin{aligned}\mathbb{P}(k_n|\alpha,n) &= c_n(k_n)\, n!\, \alpha^{k_n} \frac{\Gamma(\alpha+1)/\alpha}{\Gamma(\alpha+1+n)/(\alpha+n)} \\ &= c_n(k_n)\, n!\, \alpha^{k_n} \frac{(\alpha+n)\,\beta(\alpha+1,n)}{\alpha\Gamma(n)}\end{aligned}$$

We now replace the Beta function with its definition as an integral:

$$\mathbb{P}(k_n|\alpha,n) = \frac{c_n(k_n)\, n!}{\Gamma(n)}\, \alpha^{k_n-1}(\alpha+n) \int_0^1 t^{\alpha}(1-t)^{n-1}dt$$

By Bayes' formula the posterior for α conditional on k_n is thus:

$$\pi(\alpha|k_n,n) \;\propto\; \pi(\alpha)\pi(k_n|\alpha,n) \;\propto\; \pi(\alpha)\,\alpha^{k_n-1}(\alpha+n) \int_0^1 t^{\alpha}(1-t)^{n-1}dt$$

This suggests that we can interpret $\pi(\alpha|k_n,n)$ as the marginal of a joint distribution for α and another variable v defined on the continuous space $v \in (0,1)$:

$$\pi(\alpha,v|k_n,n) \propto \pi(\alpha)\,\alpha^{k_n-1}(\alpha+n)\,v^{\alpha}(1-v)^{n-1}$$

Thus we can now establish conditional posteriors $\pi(\alpha|v,k_n,n)$ and $\pi(v|\alpha,k_n,n)$. Under a Gamma prior for α (Eqn. 4.5) we have: [1]

$$\begin{aligned}\pi(\alpha|v,k_n,n) &\propto \mathcal{G}a(\alpha;c,d)\,\alpha^{k_n-1}\,(\alpha+n)\,v^{\alpha} \;\propto\; \alpha^{c-1}\,e^{-\alpha/d}\,\alpha^{k_n-1}(\alpha+n)\,v^{\alpha} \\ &= (\alpha+n)\,\alpha^{c+k_n-2}\,e^{-\alpha/d+\alpha\log(v)} \\ &= \alpha^{c+k_n-1}\,e^{-\alpha(1/d-\log(v))} + n\,\alpha^{c+k_n-2}\,e^{-\alpha(1/d-\log(v))}\end{aligned}$$

[1] Note that Escobar and West [30] define $\mathcal{G}a(a,b)$ with b being the scale parameter on pg. 584, which is a typo and b is actually the rate parameter (inverse of scale).

This can be written as a mixture of two Gamma distributions:

$$(\alpha|v, k_n, n) \sim \pi_v \, \mathcal{G}a\,(c + k_n,\, \theta_v) + (1 - \pi_v)\, \mathcal{G}a\,(c + k_n - 1,\, \theta_v) \qquad \text{(C.10)}$$

where $\theta_v = \left(\frac{1}{d} - \log(v)\right)^{-1}$ is the scale parameter, and the weights π_v are given by the ratio of the normalising constants of the two Gamma distributions. Recall the density of the Gamma distribution is defined as:

$$\mathcal{G}a(x; k, \theta) = x^{k-1} \frac{e^{-x/\theta}}{\theta^k \, \Gamma(k)} \quad \text{for } x > 0 \text{ and } k, \theta > 0 \qquad \text{(C.11)}$$

where k and θ denote the shape and scale, respectively.

$$\frac{\pi_v \, n}{1 - \pi_v} \quad = \quad \frac{\theta_v^{c+k_n} \, \Gamma(c + k_n)}{\theta_v^{c+k_n-1} \, \Gamma(c + k_n - 1)} \quad = \quad \frac{c + k_n - 1}{\frac{1}{d} - \log(v)} \qquad \text{(C.12)}$$

Thus we get:

$$\pi_v = \left\{ 1 + \frac{n \left(\frac{1}{d} - \log(v)\right)}{c + k_n - 1} \right\}^{-1} \qquad \text{(C.13)}$$

Now consider the conditional distribution for $v|\alpha, k_n, n$:

$$\pi(v|\alpha, k_n, n) \propto v^{\alpha}(1 - v)^{n-1} \qquad (0 < v < 1)$$

This is simply a Beta distribution with mean $(\alpha + 1)/(\alpha + n + 1)$:

$$\pi(v|\alpha, k_n, n) = \mathcal{B}(v;\ \alpha + 1,\ n) \qquad \text{(C.14)}$$

Thus we can sample α by first sampling $v \sim \mathcal{B}(\alpha + 1,\ n)$ and then sample α from the mixture of Gamma distributions defined above.

C.2 Implementation

The optimal proposal for the particle filter is $q(x_n|x_{1:n-1}, y_{1:n}, \alpha) = \pi_n(x_n|x_{n-1})$:

$$\begin{aligned} q(x_n|x_{1:n-1}, y_{1:n}, \alpha) = \mathbb{P}(x_n|x_{1:n-1}, y_{1:n}, \alpha) &= \frac{\pi(y_{1:n}|x_{1:n}, \alpha)\,\mathbb{P}(x_n|x_{1:n-1}, \alpha)}{\pi(y_{1:n}|x_{1:n-1}, \alpha)} \\ &= \frac{\pi(y_{1:n}|x_{1:n}, \alpha)\,\mathbb{P}(x_n|x_{1:n-1}, \alpha)}{\sum_{x'_n} \pi(y_{1:n}|x'_n, x_{1:n-1}, \alpha)\,\mathbb{P}(x'_n|x_{1:n-1}, \alpha)} \end{aligned} \tag{C.15}$$

Since we have a finite (and generally low) number of clusters, we can compute $q(x_n|x_{1:n-1}, y_{1:n}, \alpha)$ for all possible values of x_n. Thus we can compute the normalising constant and sample from $\mathbb{P}(x_n|x_{1:n-1}, y_{1:n}, \alpha)$ exactly.

$$x_n \sim \frac{\sum_{j=1}^{k_{n-1}+1} \pi(y_{1:n}|x_n, x_{1:n-1}, \alpha)\,\mathbb{P}(x_n|x_{1:n-1}, \alpha)\,\delta_j(x_n)}{\sum_{x'_n=1}^{k_{n-1}+1} \pi(y_{1:n}|x'_n, x_{1:n-1}, \alpha)\,\mathbb{P}(x'_n|x_{1:n-1}, \alpha)}$$

As noted above, in SMC the optimal proposal for x_n is $\pi_n(x_n|x_{1:n-1})$, which gives the following (incremental) weight (the dependence on α is suppressed for ease of notation):

$$\begin{aligned} w_n(x_n) = \frac{\pi_n(x_{1:n})}{\pi_{n-1}(x_{1:n-1})\,\pi_n(x_n|x_{1:n-1})} &= \frac{\pi_n(x_{1:n-1})\,\cancel{\pi_n(x_n|x_{1:n-1})}}{\pi_{n-1}(x_{1:n-1})\,\cancel{\pi_n(x_n|x_{1:n-1})}} \\ = \frac{\mathbb{P}(x_{1:n-1}|y_{1:n})}{\mathbb{P}(x_{1:n-1}|y_{1:n-1})} &= \frac{\sum_{x'_n} \mathbb{P}(x'_n, x_{1:n-1}|y_{1:n})}{\mathbb{P}(x_{1:n-1}|y_{1:n-1})} \\ = \frac{\sum_{x'_n} \pi(y_{1:n}|x'_n, x_{1:n-1})\,\mathbb{P}(x'_n, x_{1:n-1}) \,/\, \pi(y_{1:n})}{\pi(y_{1:n-1}|x_{1:n-1})\,\mathbb{P}(x_{1:n-1}) \,/\, \pi(y_{1:n-1})} & \\ \propto \frac{\sum_{x'_n} \pi(y_{1:n}|x'_n, x_{1:n-1})\;\mathbb{P}(x'_n|x_{1:n-1})}{\pi(y_{1:n-1}|x_{1:n-1})} & \end{aligned} \tag{C.16}$$

$$\begin{aligned} &= \sum_{x'_n} \frac{\prod_{j=1}^{k_n} f(m_n^j, \bar{y}_n^j, (\hat{\sigma}_n^j)^2|x_{1:n-1}, x'_n, y_{1:n})}{\prod_{j=1}^{k_{n-1}} f(m_{n-1}^j, \bar{y}_{n-1}^j, (\hat{\sigma}_{n-1}^j)^2|x_{1:n-1}, y_{1:n-1})}\;\mathbb{P}(x'_n|x_{1:n-1}) \\ &= \frac{\alpha f(1, y_n, 0)}{n-1+\alpha} + \sum_{j=1}^{k_{n-1}} \frac{m_{n-1}^j}{n-1+\alpha}\;\frac{f(m_n^j,\, \bar{y}_n^j,\, (\hat{\sigma}_n^j)^2|x_{1:n-1}, x_n{=}j, y_{1:n})}{f(m_{n-1}^j,\, \bar{y}_{n-1}^j,\, (\hat{\sigma}_{n-1}^j)^2|x_{1:n-1}, y_{1:n-1})} \end{aligned} \tag{C.17}$$

where f is the likelihood contribution from a cluster at time n (the sufficient statistics obviously depend on x_n):

$$f(m_n^j, \bar{y}_n^j, (\hat{\sigma}_n^j)^2 | x_{1:n}, y_{1:n}) = \\ \frac{b^a \, \Gamma\left(a + m_n^j/2\right)}{\Gamma(a)\sqrt{m_n^j \tau + 1}} \left(b + \frac{m_n^j}{2} \left[\left(\hat{\sigma}_n^j\right)^2 + \frac{(\bar{y}_n^j - \eta)^2}{1 + m_n^j \tau} \right] \right)^{-(a + \frac{m_n^j}{2})}$$

Note that the weight for the particle is independent of the choice of x_n. We also see that the numerator in equation (C.16) is simply the normalisation factor used in equation (C.15). This allows for an efficient computation of the particle weight since both the normalisation constant and the likelihood of the previous time steps have already been computed.