On auxiliary variables and many-core architectures in computational statistics

Anthony Lee

Balliol College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2011

Acknowledgements

I have been extremely fortunate in having had a host of insightful and influential advisors in my time as a student. Nando de Freitas can be credited with first impressing upon me the notion that statistical reasoning provides a powerful and flexible framework for making sense of data in an uncertain world. Arnaud Doucet has been a constant source of wisdom, first introducing me to the field of Monte Carlo methodology in a breathtaking course and later supervising my research for a Master of Science degree at the University of British Columbia. At Oxford, Chris Holmes' exceptional understanding of both methodology and the applications of statistics has been a joy to witness and I could not have asked for a more supportive or visionary supervisor. My research has also enabled me to learn from Christophe Andrieu, whose enthusiasm, patience and sage comments make him a delight to work with. In addition, I have had the opportunity to work less closely with a number of others during my time at Oxford, including Nathaniel Korda, Mike Giles, François Caron and Christopher Yau, all of whom I have learnt much from.

I acknowledge helpful comments from examiners for my transfer, confirmation of status, and viva: Peter Clifford, George Nicholson, Geoff Nicholls and Nicolas Chopin.

I would like to thank the Clarendon Fund, whose generous scholarship enabled my studies at Oxford, as well as the Oxford-Man Institute of Quantitative Finance, which in addition to being a fantastic research environment also provided funds which were of great importance in facilitating my research.

Finally, I would like to thank my family and friends for their love and support throughout the years.

Abstract

Emerging many-core computer architectures provide an incentive for computational methods to exhibit specific types of parallelism. Our ability to perform inference in Bayesian statistics is often dependent upon our ability to approximate expectations of functions of random variables, for which Monte Carlo methodology provides a general purpose solution using a computer. This thesis is primarily concerned with exploring the gains that can be obtained by using many-core architectures to accelerate existing population-based Monte Carlo algorithms, as well as providing a novel general framework that can be used to devise new population-based methods.

Monte Carlo algorithms are often concerned with sampling random variables taking values in $\mathcal{X}$ whose density is known up to a normalizing constant. Population-based methods typically make use of collections of interacting auxiliary random variables, each of which is in $\mathcal{X}$, in specifying an algorithm. Such methods are good candidates for parallel implementation when the collection of samples can be generated in parallel and their interaction steps are either parallelizable or negligible in cost. The first contribution of this thesis is in demonstrating the potential speedups that can be obtained for two common population-based methods, population-based Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC).

The second contribution of this thesis is in the derivation of a hierarchical family of sparsity-inducing priors in regression and classification settings. Here, auxiliary variables make possible the implementation of a fast algorithm for finding local modes of the posterior density. SMC, accelerated on a many-core architecture, is then used to perform inference for a range of prior specifications to gain an understanding of sparse association signal in the context of genome-wide association studies.

The third contribution is in the use of a new perspective on reversible MCMC kernels that allows for the construction of novel population-based methods. These methods differ from most existing methods in that one can make the resulting kernels define a Markov chain on $\mathcal{X}$. A further development is that one can define kernels in which the number of auxiliary variables is given a distribution conditional on the values of the auxiliary variables obtained so far. This is perhaps the most important methodological contribution of the thesis, and the adaptation of the number of particles used within a particle MCMC algorithm provides a general purpose algorithm for sampling from a variety of complex distributions.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Statistics is the scientific study of the collection and analysis of data. This thesis is concerned with parametric statistics, in which one models data as being generated probabilistically according to a distribution with unknown parameters. Treating data as a discrete-valued random variable X, this means we use a probability mass function $f_X : \mathcal{X} \to [0, 1]$ that assigns a probability to each realization of $X \in \mathcal{X}$ that depends on the value of some finite-dimensional parameter $\theta \in \Theta$, i.e.

$$\mathbb{P}_\theta[X = x] = f_X(x; \theta)$$

If X is instead continuous, the analogous function we deal with is a probability density function $f_X : \mathcal{X} \to [0, \infty)$ and the probability that $X \in A \subseteq \mathcal{X}$ is given by

$$\mathbb{P}_\theta[X \in A] = \int_A f_X(x; \theta) dx.$$

Statistical inference in a parametric context is, loosely speaking, the process by which we infer the value of the parameter θ from data X. There are different approaches for solving this problem, and the differences are meaningful. However, this thesis is concerned primarily with Bayesian inference, a major approach, which is briefly introduced in the following section. In statistics, we are usually interested in the behaviour of f_X as θ changes and X is fixed, since X often represents data that has been realized at the time of inference. As such, f_X viewed as a function of

θ is called the likelihood function, or the likelihood.

1.1 Bayesian Statistics

In Bayesian statistics, one models the unknown parameter θ as a random variable in order to make probabilistic statements about it. In order to do this, one first defines a prior on θ. This usually takes the form $f_\theta : \Theta \to [0, 1]$ if θ is discrete-valued and $f_\theta : \Theta \to [0, \infty)$ if θ is continuous-valued. From now on, we will concentrate on the continuous-valued case, noting that the discrete-valued case is generally analogous. From a Bayesian perspective, the data X is viewed as being generated conditionally on θ so we write $f_{X|\theta}(x|\vartheta)$ to represent the density of $X = x$ given $\theta = \vartheta$. Note that a Bayesian model requires specification both of a parametrized likelihood and a prior distribution over the parameters used in the parametrization. The prior is intended to contain all prior information about θ before obtaining data x.

Equipped with a prior on θ and some data x according to $f_{X|\theta}$, the posterior density of θ at $\vartheta \in \Theta$ given x is given by Bayes' theorem:

$$f_{\theta|X}(\vartheta|x) = \frac{f_{X|\theta}(x|\vartheta) f_\theta(\vartheta)}{f_X(x)}$$

where $f_X(x) = \int_\Theta f_{X|\theta}(x|\vartheta) f_\theta(\vartheta) d\vartheta$ is often called the marginal likelihood of x.

The posterior density of θ given data x encapsulates all the information learned about θ from x and is used to compute values of interest. For example, the conditional expectation of a function $\phi : \Theta \to \mathbb{R}^k$ of the random variable θ given x can be written

$$\mathbb{E}[\phi(\theta)|X = x] = \int_\Theta \phi(\vartheta) f_{\theta|X}(\vartheta|x) d\vartheta. \tag{1.1}$$

Some examples of ϕ include $\phi(\theta) = \theta$ so that (1.1) is the mean and $\phi(\theta) = \mathbf{1}_A(\theta)$ where $A \subseteq \Theta$ so it represents $\mathbb{P}[\theta \in A|X = x]$. Here $\mathbf{1}$ denotes the indicator function. For $\phi(\theta) = f_{X'}(x'|\theta)$, it represents the predicted density of X' given x. If θ is univariate, then one can define the conditional quantiles of θ via $Q(p) = \inf\{t :$

$p \leq \int_{-\infty}^{t} f_{\theta|X}(\vartheta|x)d\vartheta\}$. In addition, the marginal likelihood $f_X(x)$ is often also a quantity of interest that cannot be computed easily. It is for this reason that many practical scenarios for inference involve the assumption that we can compute the posterior density pointwise up to a normalizing constant.

Since much of Bayesian inference involves computations of conditional expectations of functions of the random variable θ, there is a need to calculate integrals of the form of (1.1) quickly and accurately to analyze data in practical applications. However, except in restrictive cases, integrals of this form cannot be computed analytically and approximations need to be used. Even in cases where the parameters of interest take on a finite number of values and the integral is replaced by a summation, computation of expectations can be prohibitively expensive, e.g. when this finite number is too large. Advances in methodology for providing such approximations using a computer and increases in the computational power available to statisticians have nevertheless made it possible to model larger data sets and use more complex models that better describe the process by which data is generated. Much of this thesis is concerned with estimating expectations of this type using Monte Carlo methodology, which is introduced in Section 1.2.

From this point on, subscripts to denote the random variables involved in a density function are omitted unless there is ambiguity.

1.1.1 Complex Models

Imagine we have data $x_{1:n}$ and $x_i \overset{\text{iid}}{\sim} f(\cdot|\mu, \sigma^2), \forall i \in \{1, \ldots, n\}$ where f denotes the density of a normal distribution. Given a prior g on μ and assuming σ^2 is known, the posterior is given by

$$p(\mu|x_{1:n}) = \frac{\prod_{i=1}^{n} f(x_i|\mu, \sigma^2)g(\mu)}{\int_{\mathbb{R}} \prod_{i=1}^{n} f(x_i|\mu, \sigma^2)g(\mu)d\mu}. \tag{1.2}$$

If g denotes a normal distribution with a given mean and variance, then (1.2) is also a normal distribution with a mean and variance that can be calculated easily. This is an example of a conjugate prior, where the posterior distribution for μ has the same

form as the prior, but with different values for the parameters. If we have some other nontrivial distribution for g that better encapsulates our prior information about μ, however, (1.2) might not have an analytic expression. It is in this sense that advances in computational statistics have allowed statisticians to move away from the restrictions of conjugate priors when performing Bayesian inference.

For example, imagine a classical linear regression setting in which we have n observations $\mathbf{y} = \{y_i\}_{i=1}^n$, a matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{R}^{n \times p}$ of covariates and a parameter $\boldsymbol{\beta} \in \mathbb{R}^p$ such that

$$y_i = \mathbf{x}_i \boldsymbol{\beta} + \epsilon_i$$

where $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ and σ^2 is known. The posterior density for $\boldsymbol{\beta}$ is then proportional to

$$p(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y}, \sigma^2) \propto \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\beta)'(\mathbf{y} - \mathbf{X}\beta)\right\} p(\boldsymbol{\beta})$$

where $p(\boldsymbol{\beta})$ the prior on $\boldsymbol{\beta}$. When $p(\boldsymbol{\beta})$ is a multivariate normal density, the posterior can be obtained analytically. However, if we wish to model $\boldsymbol{\beta}$ *a priori* as having each component take independent values in $\{-1, 1\}$ with equal probability, i.e. $p(\boldsymbol{\beta}) = \prod_{j=1}^p \left[\frac{1}{2}\mathbf{1}_{\{-1,1\}}\beta_j\right]$, the problem becomes much more difficult. In particular, given any conditioned Ising model $(\mathbf{J}, \mathbf{h}, m, T)$ (see, e.g., Kendall 2005) with a magnetic field on a square lattice of length m, one can choose $n = p = m^2$ and $(\mathbf{X}, \mathbf{y}, \sigma^2)$ such that the posterior probability of $\boldsymbol{\beta}$ is equivalent to the probability of a configuration of spins in the given conditioned Ising model. The conditioned Ising model is defined by

$$\mathbb{P}[\boldsymbol{\beta}] \propto \exp\left\{\frac{1}{T}\left(\sum_{i=1}^{m^2}\sum_{j=1}^{i-1} J_{ij}\beta_i\beta_j + \sum_{i=1}^{m^2} h_i\beta_i\right)\right\}$$

(Kindermann et al. 1980) so letting $\mathbf{A} = \mathbf{X}'\mathbf{X}$, one can set $\mathbf{A}_{ij}$ equal to $\mathbf{J}_{ij}$ for $i \neq j$, $\mathbf{A}_{ii}$ large enough for $i \in \{1, \ldots, m^2\}$ so that $\mathbf{A}$ is symmetric positive definite, $\mathbf{y} = (\mathbf{X}')^{-1}\mathbf{h}$ and $\sigma^2 = T$ one obtains the same posterior for $\boldsymbol{\beta}$ in the linear regression problem where $\boldsymbol{\beta}$ takes values in $\{-1, 1\}^{m^2}$.

When m is large, it is computationally impractical to enumerate all 2^{m^2} possible values of $\boldsymbol{\beta}$ and it is nontrivial to deal with this distribution otherwise.

1.1.2 Latent Variables

Some of the computational difficulties in Bayesian statistics involve models in which some random variables are not observed but play a crucial role in the modeling of observed data. A simple example of a latent variable model involves univariate data that comes from a Gaussian mixture model, i.e. we have

$$f(x|\mathbf{w}, \boldsymbol{\mu}, \sigma^2) = \sum_{j=1}^{d} w_j g(x|\mu_j, \sigma^2)$$

where $g(x|\mu, \sigma^2)$ is, for example, the density of a normal distribution with mean μ and variance σ^2 and $\sum_{j=1}^{d} w_j = 1$. The vector $\mathbf{w}$ can be seen as encoding the weights of each normal component in the mixture model. Given data $\mathbf{x} = \{x_i\}_{i=1}^{n}$, σ^2 and a prior distribution on $\boldsymbol{\theta} = (\mathbf{w}, \boldsymbol{\mu})$, one may wish to infer the posterior distribution of $\boldsymbol{\theta}$ given $\mathbf{x}$. While one can compute the likelihood analytically, in contrast to examples that will follow, it is occasionally useful to introduce a latent variable $\mathbf{k} \in \{1, \ldots, d\}^n$ indicating which mixture component is responsible for each observation. The joint density of observation x_i and assignment k_i is then

$$\tilde{f}(x_i, k_i|\mathbf{w}, \boldsymbol{\mu}, \sigma^2) = w_{k_i} g(x_i|\mu_{k_i}, \sigma^2)$$

and it should be clear that the marginal $\tilde{f}(x_i|\mathbf{w}, \boldsymbol{\mu}, \sigma^2) = \sum_{k_i=1}^{d} \tilde{f}(x_i, k_i|\mathbf{w}, \boldsymbol{\mu}, \sigma^2) = f(x_i|\mathbf{w}, \boldsymbol{\mu}, \sigma^2)$.

1.1.2.1 Hierarchical Models

Another example of a latent variable model involves a hierarchical specification of the prior distribution $p(\theta)$. At the stage of inference, one might have a suitable choice of parametric family for this prior but have uncertainty about what hyperparameters θ_1 the prior should take. This can be dealt with by placing a distribution on θ_1. More formally, one can define a number of levels in the hierarchy h and define the prior on θ via

$$p(\theta) = \int_{\Theta_1 \times \cdots \Theta_h} p_1(\theta|\theta_1) p_2(\theta_1|\theta_2) \cdots p_h(\theta_{h-1}|\theta_h) p_{h+1}(\theta_h) d\theta_{1:h}.$$

The variables $\theta_{1:h}$ can be thought of as latent variables, and while they allow for a rich class of prior specifications, the inclusion of extra unobserved variables can cause computational problems since the dimensionality of the integrals involved in estimating posterior expectations, for example, can increase as levels are added.

1.1.2.2 State-Space Models

A more sophisticated form of latent variable model involves observed data $\mathbf{y} = \{y_t\}_{t=1}^T$ that can be thought of as being produced sequentially in time and that is independent conditional on an underlying hidden Markov process. In the case of a discrete-time hidden process and observations, this is referred to as a state-space model and follows the general form

$$\begin{aligned} x_0 &\sim g_0(\cdot|\theta) \\ x_t &\sim g_t(\cdot|x_{t-1}, \theta), \text{ for } t = 1, \ldots, T \\ y_t &\sim f_t(\cdot|x_t, \theta), \quad \text{for } t = 1, \ldots, T \end{aligned}$$

For state-space models there are various possible parameters of interest. One may be interested in the filtering distribution at a particular time t

$$p(x_t|y_{1:t}, \theta) = \frac{\int_{\mathcal{X}^t} p(x_{0:t}, y_{1:t}|\theta) dx_{0:t-1}}{p(y_{1:t}|\theta)},$$

the distribution of $x_{0:T}$ conditional upon $y_{1:T}$ and θ

$$p(x_{0:T}|y_{1:T}, \theta) = \frac{p(x_{0:T}, y_{1:T}|\theta)}{p(y_{1:T}|\theta)},$$

or the marginal likelihood of $y_{1:T}$ given θ

$$p(y_{1:T}|\theta) = \int_{\mathcal{X}^{T+1}} p(x_{0:T}, y_{1:T}|\theta) dx_{0:T}.$$

Inference on parameters in this context can be highly challenging from a computational perspective, involving either high-dimensional densities or high-dimensional integrals that are rarely analytic.

1.1.2.3 Variable Selection

There are many contexts in which we are unsure of the model that gives rise to a given set of observations $\mathbf{y} = \{y_i\}_{i=1}^n \in \mathbb{R}^n$. A particular case of this occurs when there is a large number p of potential covariates in a regression or classification setting but there is reason to believe that not all of them even belong in the 'true' model that generated $\mathbf{y}$. Let us denote by $\gamma \in \{0,1\}^p$ a string representation of each of the 2^p possible models obtained by including a subset of the p variables, where $\gamma_j = 0$ implies that variable j is not in the model and $\gamma_j = 1$ implies that it is. Denoting the design matrix of covariates by $\mathbf{X} \in \mathbb{R}^{n \times p}$, the subset of $\mathbf{X}$ with covariates in the model γ by $\mathbf{X}_\gamma = \mathbf{X}_{1:n,\gamma}$, a vector of coefficients β_γ relating to each variable in the model $\mathcal{M}_\gamma$, a density g_γ for $\beta_\gamma|\mathcal{M}_\gamma$ and a likelihood function for each model by $f_\gamma(\mathbf{y}|\mathbf{X}_\gamma, \boldsymbol{\beta}_\gamma)$, one can define the likelihood for each model by

$$f_\gamma(\mathbf{y}|\mathbf{X}_\gamma, \mathcal{M}_\gamma) = \int_{\mathbb{R}^{||\gamma||_1}} f_\gamma(\mathbf{y}|\mathbf{X}_\gamma, \boldsymbol{\beta}_\gamma) g_\gamma(\boldsymbol{\beta}_\gamma|\mathcal{M}_\gamma) d\boldsymbol{\beta}_\gamma. \tag{1.3}$$

Given a prior probability for each model p, the posterior probability of any model is then

$$p(\mathcal{M}_\gamma|\mathbf{y}, \mathbf{X}) = \frac{f_\gamma(\mathbf{y}|\mathbf{X}_\gamma, \mathcal{M}_\gamma) p(\mathcal{M}_\gamma)}{\sum_{\gamma' \in \{0,1\}^p} f_{\gamma'}(\mathbf{y}|\mathbf{X}_{\gamma'}, \mathcal{M}_{\gamma'}) p(\mathcal{M}_{\gamma'})}. \tag{1.4}$$

The difficulty in performing inference on models in this context is twofold. First, the likelihood $f_\gamma(\mathbf{y}|\mathbf{X}_\gamma, \mathcal{M}_\gamma)$ is generally not available analytically and so the integral in (1.3) must be approximated. Second, even if the likelihood is tractable the number of possible models is 2^p so expectations involving (1.4) can become prohibitively expensive quickly as p increases, which is especially unfortunate since it is often when p is large that we wish to determine the relative probabilities of models that exclude some covariates.

1.1.2.4 Approximate Bayesian Computation

Approximate Bayesian Computation is a recent auxiliary variable approach proposed in the population genetics literature but widely applicable to situations in which one

cannot evaluate the parametrized likelihood function $f(\cdot|\theta)$ but can simulate according to it using a computer or by some other means. In the case of discrete data $\mathbf{x} \in \mathcal{X}$, the methodology is based on the principle that the probability of the data given some parameter θ is equal to $f(\mathbf{x}|\theta) = \sum_{\mathbf{z}\in\mathcal{X}} f(\mathbf{x}, \mathbf{z}|\theta) = \sum_{\mathbf{z}\in\mathcal{X}} \mathbf{1}_{\mathbf{z}}(\mathbf{x}) f(\mathbf{z}|\theta)$. In a continuous setting this identity is not generally practically useful so approximations are used instead. One such form is $\tilde{f}_1(\mathbf{x}|\theta) = \int_{\mathcal{X}} \frac{\mathbf{1}_{B_\epsilon(\mathbf{z})}(\mathbf{x})}{Z_\epsilon} f(\mathbf{z}|\theta) d\mathbf{z}$ where $B_\epsilon(\mathbf{z})$ denotes a ball of radius ϵ around $\mathbf{z}$ under some metric and $Z_\epsilon = \int_{\mathcal{X}} \mathbf{1}_{B_\epsilon(\mathbf{z})}(\mathbf{x}') d\mathbf{x}'$ is independent of the value of $\mathbf{z}$. Another is $\tilde{f}_2(\mathbf{x}|\theta) = \int_{\mathcal{X}} K_\epsilon(\mathbf{z}, \mathbf{x}) f(\mathbf{z}|\theta) d\mathbf{z}$ where K_ϵ is a kernel. In some cases, usually due to the high-dimensionality of $\mathbf{x}$, one replaces $\mathbf{x}$ and $\mathbf{z}$ by $S(\mathbf{x})$ and $S(\mathbf{z})$ respectively where $S(\cdot)$ encodes a summary statistic of the data.

Returning to the unsummarized discrete case, inference on θ is possible via the joint density $f(\mathbf{x}, \mathbf{z}|\theta)$ since we have

$$f(\theta|\mathbf{x}) = \frac{\sum_{\mathbf{z}\in\mathcal{X}} f(\mathbf{x}, \mathbf{z}|\theta) p(\theta)}{\int_\Theta \sum_{\mathbf{z}\in\mathcal{X}} f(\mathbf{x}, \mathbf{z}|\theta) p(\theta) d\theta}.$$

Although, by definition, this cannot be marginalized analytically one does have recourse to approximation methods that can work with the restriction that one can only simulate according to $f(\cdot|\theta)$.

1.2 Computational Statistics

It should be firmly established at this point that Bayesian inference for models of nontrivial complexity require computational tools that alleviate the need to deal with intractable integrals and sums. In this section a brief overview of Monte Carlo methods that are used and for which methodological improvements are suggested in this thesis is given. This is a rich field of study that is generally concerned with the approximation of expectations of functions of random variables and uses auxiliary random variables in the construction of such approximations.

1.2.1 Monte Carlo Methods

This thesis is concerned primarily with a large class of computational methods for approximating integrals, and particularly those that represent expectations of functions of random variables whose density is known pointwise up to a normalizing constant. This setting is motivated by computational issues in Bayesian inference but is applicable more generally, e.g. in statistical physics where one can be interested in understanding properties of a stochastic system that does not have unknown parameters (Metropolis and Ulam 1949), or for performing a significance test when the distribution function of a test statistic is intractable (Barnard 1963). The methodology is broadly termed Monte Carlo methodology, in reference to its dependence on random samples, and in its simplest form was proposed by physicists in anticipation of access to the ENIAC machine in 1947 (Ulam et al. 1947). The introduction of the methodology to the statistical literature came in Metropolis and Ulam (1949).

1.2.2 Classical Monte Carlo

Consider the expectation of a function ϕ of random variable $X \in \mathcal{X}$, where X is distributed according to a density π:

$$\mathbb{E}_\pi[\phi] = \int_{\mathcal{X}} \phi(x)\pi(x)dx. \tag{1.5}$$

The most basic form of Monte Carlo integration involves generating N i.i.d. samples according to π, $\{X_i\}_{i=1}^N$ and computing the average

$$I_{MC}^N = \frac{1}{N}\sum_{i=1}^{N} \phi(X_i).$$

I_{MC}^N is an unbiased estimate of $\mathbb{E}_\pi[\phi]$ and has a variance given by

$$\mathbb{V}[I_{MC}^N] = \frac{\mathbb{E}_\pi[\phi^2] - \mathbb{E}_\pi[\phi]^2}{N}$$

where $\mathbb{E}_\pi[\phi^2] = \int_{\mathcal{X}} \phi(x)^2\pi(x)dx$. This is a useful method when one can sample according to π and when the numerator of the variance term is small enough that

for the number of samples N that can be generated in a reasonable amount of time, the variance is small enough for the given estimation problem. In Bayesian inference, when viewing π as the posterior distribution of the parameters of interest the first issue is usually the most problematic. Algorithm 1.1 describes a classical Monte Carlo algorithm for estimating (1.5).

Algorithm 1.1 A Classical Monte Carlo Algorithm

1. For $i = 1, \ldots, N$, sample $X_i \overset{\text{iid}}{\sim} \pi(\cdot)$.
2. Output $\frac{1}{N} \sum_{i=1}^{N} \phi(X_i)$.

Note that even if $\mathbb{E}_\pi[\phi^2]$ is not finite, I_{MC}^N can still converge almost surely to $\mathbb{E}_\pi[\phi]$ via the strong law of large numbers as $N \to \infty$ (see, e.g., Williams 1991). However, its usefulness in statistical inference is then somewhat diminished.

1.2.2.1 Rejection Sampling

In many cases, one cannot sample directly according to the density π of interest. In some circumstances, however, one can sample from a distribution with a different density q for which one can upper bound the term

$$M = \sup_{x \in \mathcal{X}} \frac{\pi(x)}{q(x)}$$

with some $C \geq M$. One can then define a joint distribution on $\mathcal{X} \times (0, 1)$ such that

$$g(x, u) = q(x)\mathcal{U}(u)$$

where $\mathcal{U}$ denotes the uniform density on $(0, 1)$. Now, the conditional density of x given $u < \frac{\pi(x)}{Cq(x)}$ is given by

$$\begin{aligned} g\left(x \middle| u < \frac{\pi(x)}{Cq(x)}\right) &= \frac{g(x, u < \frac{\pi(x)}{Cq(x)})}{g(u < \frac{\pi(x)}{Cq(x)})} \\ &= \pi(x) \end{aligned}$$

which suggests an algorithm that proposes $(x, u) \sim g$ and outputs x if $u < \frac{\pi(x)}{Cq(x)}$. This algorithm, known as rejection sampling, produces samples according to π. It is most efficient at producing samples when $C = M$ as the probability of outputting x is linear in $1/C$.

It is possible to show using essentially the same argument that one need not even be able to compute π or even q except up to a normalizing constant, i.e. we can compute only $\gamma(x) = Z\pi(x)$ where $Z = \int_{\mathcal{X}} \gamma(x)dx$ is the unknown normalizing constant for γ, and a similar unnormalized density q^* can be computed for q. This affects only the choice of C, which must be a an upper bound on the ratio of the unnormalized densities and impacts the efficiency of the algorithm for producing samples only if the bound obtained is not proportional to the bound that would have been obtained if π and q could be computed in their normalized form. Letting $C^* \geq M^* = \sup_{x \in \mathcal{X}} \frac{\gamma(x)}{q^*(x)}$, Algorithm 1.2 describes this general-purpose sampling procedure for obtaining samples according to π.

Algorithm 1.2 An Accept-Reject Algorithm

1. Sample $X \sim q(\cdot)$ and $U \sim \mathcal{U}(\cdot)$
2. If $U \leq \frac{\gamma(X)}{C^* q^*(X)}$, output X. Otherwise, return to 1.

Despite the generality and even theoretically wide applicability of this algorithm, it can be difficult to find a density q that we can sample according to and for which we can obtain a low enough upper bound C for the algorithm to be practically useful when π is high-dimensional or otherwise complex. Nevertheless, rejection sampling plays a role in other, more sophisticated Monte Carlo algorithms.

1.2.2.2 Importance Sampling

An alternative classical method for estimating expectations of the form given in (1.5) is obtained by noting that

$$\mathbb{E}_\pi[\phi] = \int_{\mathcal{X}} \phi(x)\pi(x)dx = \int_{\mathcal{X}} \phi(x)\frac{\pi(x)}{q(x)}q(x)dx \tag{1.6}$$

where $q(x) > 0$ whenever $\pi(x) > 0$. In (1.6), q can be seen as representing a proposal, or importance, density associated with a distribution from which we can sample easily. This suggests the following estimator

$$I_{IS}^N = \frac{1}{N} \sum_{i=1}^{N} \phi(X_i) \frac{\pi(X_i)}{q(X_i)}$$

where $\{X_i\}_{i=1}^N$ are now samples from the distribution associated with q. I_{IS}^N is an unbiased estimate of $\mathbb{E}_\pi[\phi]$ and has a variance given by

$$\mathbb{V}[I_{IS}^N] = \frac{\mathbb{E}_\pi[\phi^2 \frac{\pi}{q}] - \mathbb{E}_\pi[\phi]^2}{N}.$$

It is common to view I_{IS}^N as a weighted average of ϕ evaluated at the samples obtained with weights given by $w(X_i) = \frac{\pi(X_i)}{q(X_i)}$.

Algorithm 1.3 An Importance Sampling Algorithm

1. For $i = 1, \ldots, N$, sample $X_i \overset{\text{iid}}{\sim} q(\cdot)$.
2. For $i = 1, \ldots, N$, compute $w(X_i) = \frac{\pi(X_i)}{q(X_i)}$.
3. Output $\frac{1}{N} \sum_{i=1}^N w(X_i)\phi(X_i)$.

One can, in principle, obtain a lower variance for I_{IS}^N than for I_{MC}^N, depending on the choice of q. In practice, however, for a complicated target density π the issue is often trying to ensure that the variance of I_{IS}^N is not prohibitively large because the choice of q is heavily constrained to densities according to which we can sample easily. Furthermore, in many situations one attempts to choose a q that approximates π irrespective of ϕ but where the quantity $\sup_{x \in \mathcal{X}} w(x)$ is finite since this is a sufficient condition for $\mathbb{V}[I_{IS}^N]$ to be finite if $\mathbb{V}[I_{MC}^N]$ is finite. Furthermore, in many cases one is interested in computing the expectation of more than one function of X under π and so tuning the proposal for a particular function can be counterproductive.

In the event that one can only compute π up to a normalizing constant, i.e. we can compute $\gamma(x) = Z\pi(x)$ where $Z = \int_{\mathcal{X}} \gamma(x)dx$, one can perform a normalized version of importance sampling that is no longer unbiased but retains consistency.

The estimator is given by

$$I_{NIS}^N = \sum_{i=1}^N W_i \phi(X_i)$$

where $\{X_i\}_{i=1}^N$ are again samples from the distribution associated with q,

$$W_i = \frac{w(X_i)}{\sum_{j=1}^N w(X_j)}$$

and $w(X_i) = \frac{\gamma(X_i)}{q(X_i)}$.

Algorithm 1.4 A Normalized Importance Sampling Algorithm

1. For $i = 1, \ldots, N$, sample $X_i \overset{\text{iid}}{\sim} q(\cdot)$.
2. For $i = 1, \ldots, N$, compute $w(X_i) = \frac{\gamma(X_i)}{q(X_i)}$.
3. For $i = 1, \ldots, N$, compute $W_i = \frac{w(X_i)}{\sum_{j=1} w(X_j)}$.
4. Output $\sum_{i=1}^N W_i \phi(X_i)$.

Unlike rejection sampling, importance sampling does not obtain samples according to π but instead weights samples from another distribution for the purpose of estimating (1.5). It is similarly general and theoretically even more widely applicable since it does not require the knowledge, or even existence, of an upper bound for $\gamma(x)/q(x)$. Nevertheless, this simple estimator when π is high-dimensional or otherwise complex has unacceptably high variance in many cases. However, it is used as a methodological component in a variety of more sophisticated algorithms.

1.2.3 Markov Chain Monte Carlo

The introduction of Markov chain Monte Carlo (MCMC) methodology in Metropolis et al. (1953) has revolutionized our ability to estimate expectations of the form given in (1.5), although it took about 35 years for it to be utilized prominently in solving computational issues in Bayesian inference (see, e.g., Robert and Casella 2011).

The general methodology involves the definition of a Markov transition kernel K that leaves the density π invariant, and is additionally aperiodic and ρ-irreducible

for some non-zero measure ρ. More formally, if we let $\{X_n : n \in \mathbb{Z}_+\}$ be a sequence of random variables generated by sampling X_0 from some initial distribution and then for each $n \in \mathbb{N}$ sampling $X_n \sim K(X_{n-1}, \cdot)$, K is π-invariant if

$$\int_{\mathcal{X}} \pi(dx)K(x, dy) = \pi(dy).$$

Irreducibility and aperiodicity then assure us that π is the unique invariant density of K, which satisfies

$$\|K^n(x, \cdot) - \pi(\cdot)\| \to 0 \tag{1.7}$$

as $n \to \infty$ for π-almost all x, and for all $x \in \mathcal{X}$ if K is Harris recurrent (Athreya et al. 1996; Tierney 1994). Here $\|\cdot\|$ denotes total variation distance and (1.7) holding for π-almost all x is what is meant when one says that K has an equilibrium distribution with density π.

The set of possible kernels satisfying these constraints for a given target density π is very large and difficult to characterize. Of particular interest to statisticians is the estimate of (1.5) when using the samples $\{X_i\}_{i=1}^N$

$$I_{MCMC}^N = \frac{1}{N}\sum_{i=1}^{N} \phi(X_i) \tag{1.8}$$

and how it converges to (1.5). Conditions for convergence and for the satisfaction of a central limit theorem are technical in nature and cannot even be briefly overviewed here. The interested reader is referred to Roberts and Rosenthal (2004) to obtain a review of some conditions that have been obtained.

In practice, the condition that K be π-invariant is often ensured via π-reversibility, where this means for all $x, y \in \mathcal{X}$

$$\pi(dx)K(x, dy) = \pi(dy)K(y, dx).$$

That this is a sufficient condition for π-invariance follows immediately since

$$\int_{\mathcal{X}} \pi(dx)K(x, dy) = \int_{\mathcal{X}} \pi(dy)K(y, dx) = \pi(dy).$$

K can be a mixture or cycle of π-invariant kernels $K_1, \ldots, K_m$ where a mixture

takes the form

$$K(x,\cdot) = \sum_{j=1}^{m} w_j K_j(x,\cdot)$$

with $\sum_{j=1}^{m} w_j = 1$ and a cycle takes the form

$$K(x,\cdot) = (K_1 K_2 \cdots K_m)(x,\cdot)$$

Note that a cycle of π-reversible kernels is not necessarily reversible but that mixtures and cycles of π-invariant kernels are π-invariant. The conditions, however, for ensuring irreducibility and aperiodicity for combinations of kernels are not straightforward and may need to be checked in practice, especially when the combined kernels are not irreducible on their own.

1.2.3.1 Metropolis-Hastings

The Metropolis-Hastings algorithm (Hastings 1970), a generalization of Metropolis et al. (1953), provides a flexible framework within which one can produce a π-invariant kernel K. Many MCMC kernels used in practice are combinations of Metropolis-Hastings kernels. The framework involves only the specification of a proposal Markov transition kernel $Q(x,dy) = q(x,y)dy$ that one can sample from. Sampling $z \sim K(x,\cdot)$ is defined as follows: at point x, sample $y \sim Q(x,\cdot)$ and set $z = y$ with probability $\alpha(x,y)$ and set $z = x$ otherwise, where

$$\alpha(x,y) = \min\left\{1, \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\right\}$$

is called the Metropolis-Hastings acceptance probability and we have assumed that $X_0 = x_0$ with $\pi(x_0) > 0$. It is straightforward to show that K is reversible with respect to π. Importantly, it is only necessary to be able to compute π up to a normalizing constant since the ratio of $\frac{\pi(y)}{\pi(x)}$ does not depend on this typically unknown quantity.

Popular default choices of q are a symmetric density, in which case K is called a random walk Metropolis kernel, and a density that does not depend on x, in which case K is called an independent Metropolis-Hastings kernel. In some cases

Algorithm 1.5 A Metropolis-Hastings Algorithm

Given x_0, at times $t = 0, 1, \ldots$

1. Sample $y \sim q(x_t, \cdot)$ and $u \sim \mathcal{U}$.
2. If $u \leq \min\left\{1, \frac{\pi(y)q(y,x_t)}{\pi(x_t)q(x_t,y)}\right\}$, set $x_{t+1} = y$.
   Otherwise, set $x_{t+1} = x_t$.

one can split x into components, e.g. $\mathbf{x} = (x_1, x_2)$, and for component 1 define $q_1((x_1, x_2), (y_1, y_2)) = \pi(y_1|x_2)\delta_{x_2}(y_2)$ with an analogous density q_2 for component 2. The Metropolis-Hastings acceptance ratio for a move using either q_1 or q_2 is always 1 and a mixture or cycle of kernels K_1 and K_2 associated with these proposals is irreducible, aperiodic and Harris recurrent under some conditions (Tierney 1994; Robert and Casella 2004; Roberts and Rosenthal 2006). Combinations of kernels of this form are known as Gibbs samplers, and have been used extensively in Bayesian inference since Geman and Geman (1984) and Gelfand (1990) although very closely related approaches had been circulating in statistical physics and statistics before this time (see, e.g., Robert and Casella 2011).

1.2.3.2 Practical MCMC in Statistics

It is worth mentioning that kernels for which (1.7) holds are not necessarily useful from an estimation perspective, since the equilibrium distribution of K having density π is not sufficient for the variance of (1.8) to be finite and low enough for practical use. Therefore, there is a strong interest in constructing kernels that have a fast rate of convergence, defined appropriately, for a given target density π, and for which (1.8) has a low asymptotic variance for ϕ's of interest.

For most statistical applications, one would like the an MCMC kernel to be geometrically ergodic, where this is usually defined as there existing $M : \mathcal{X} \to \mathbb{R}^+$, and $\rho < 1$ such that

$$\|K^n(x, \cdot) - \pi(\cdot)\| \leq M(x)\rho^n \tag{1.9}$$

for π-almost all x, where $\|\cdot\|$ denotes total variation distance (see, e.g., Nummelin and Tweedie 1978; Tierney 1994; Roberts and Rosenthal 1997). In this case we can

loosely refer to ρ as the inverse geometric rate of convergence. Geometric ergodicity can in some situations imply that a central limit theorem holds for (1.8) (see, e.g., Roberts and Rosenthal 1997, 2004), which is typically what is required for practical use of MCMC in statistics.

1.2.3.3 Auxiliary Variables and Population-Based MCMC

The introduction of auxiliary variables in MCMC methodology dates back at least to Swendsen and Wang (1987) and Edwards and Sokal (1988) in the physics literature and was further generalized in Besag and Green (1993) in statistics. The principle is simple: given a target density π on $\mathcal{X}$ one can construct an extended target density $\bar{\pi}$ on an extended space $\mathcal{X} \times U$ such that

$$\int_U \bar{\pi}(x, u)du = \pi(x)$$

so that the x-component of samples generated according to $\bar{\pi}$ are distributed according to π. This can be useful when, for example, we can construct an MCMC kernel that is $\bar{\pi}$-invariant and has a faster rate of convergence than an alternative MCMC kernel that is π-invariant that we might run instead.

On some occasions, the auxiliary variables u represent interpretable quantities in the model. For example, one can imagine extending the mixture model posterior density described in Section 1.1.2 with the latent variable $\mathbf{k}$ that assigns each observation to the mixture component that generated it. However, interpretability of auxiliary variables is not necessary.

A particularly fruitful and flexible framework for sampling from π in some contexts is known as population-based MCMC, in which U is $\mathcal{X}^{M-1}$ and $u = x^{(1:M-1)}$ represents $M - 1$ auxiliary variables, each of which are in $\mathcal{X}$. For convenience, we write $x^{(M)} = x$, the variable that we would like to sample according to the density π. In general, each variable $x^{(i)}$ in the extended target density is independently distributed

according to a distribution with density π_i, for $i \in \{1, \ldots, M\}$, where $\pi_M = \pi$, i.e.

$$\bar{\pi}(x^{(1:M)}) = \prod_{i=1}^{M} \pi_i(x^{(i)})$$

One fairly general view of this methodology, which has origins in Geyer (1991) and Hukushima and Nemoto (1996), is the following. One first defines a set of component-wise kernels $K_1^{(1:M)}$ such that $K_1^{(i)}$ operates only on the value of $x^{(i)}$ and is π_i-invariant, irreducible and aperiodic for $x^{(i)}$. Such kernels are clearly also $\bar{\pi}$-invariant on the joint space. Additionally, at least one other kernel operates on more than one component to induce interaction between the M components in an attempt to increase the rate of convergence of the combination of kernels that are used to define the actual Markov transition kernel that produces samples $\{X_n^{(1:M)} : n \in \mathbb{Z}_+\}$. Let the number of interaction kernels be R, and denote the set of these kernels by $K_2^{(1:R)}$. Each kernel $K_2^{(j)}$ will act on a subset of components, whose indices we can write as $\mathcal{I}_j \subset \{1, \ldots, M\}$. We denote the actual kernel that combines the component-wise kernels $K_1^{(1:M)}$ and interaction kernels $K_2^{(1:R)}$ using cycles and mixtures by K. Note that in some cases some of the component-wise kernels can be reducible, but irreducibility of K on $\bar{\pi}$ is nevertheless ensured by their combination with the interaction kernels.

Given π, there are a wide variety of possible choices for M, $\pi_{1:M-1}$, $K_1^{(1:M)}$, R, $\mathcal{I}_{1:R}$ and $K_2^{(1:R)}$ which will affect the rate of convergence of K. Jasra et al. (2007) gives a review of some of these. Note that this methodology is termed population-based due to the prominent characteristic that a population of samples in the same space as $x = x^{(M)}$ is used to facilitate sampling from π. Population-based methods in general will play a large part in the rest of this thesis.

1.2.4 Sequential Monte Carlo

Another advanced Monte Carlo methodology that has made a considerable impact on our ability to estimate conditional expectations is called sequential Monte Carlo (SMC) and has its origins in the bootstrap filter of Gordon et al. (1993). SMC is a powerful extension of importance sampling involving sequential importance sam-

pling and regular interaction between samples through resampling. They are most commonly associated with state-space models, and the algorithms derived from the methodology in this context are often referred to as particle filters. A necessarily brief description of algorithms in this setting will be presented here with interested readers referred to Doucet and Johansen (2008) and Liu (2001) for recent, more comprehensive surveys of the field. SMC methods are not restricted to this type of model, however, and Del Moral et al. (2006) makes explicit the generality with which one can devise an SMC algorithm that targets an arbitrary density π.

1.2.4.1 SMC for State-Space Models

Following Gordon et al. (1993), the main application area of SMC methodology was in state-space models, which are defined in Section 1.1.2.2. A standard introduction to SMC in state-space models usually first defines a standard importance sampling algorithm that will fail to produce estimates of conditional expectations with acceptable variance in most situations. Assume for the moment that we are interested in computing conditional expectations, for a given value of θ, of a function ϕ of $x_{0:T}$ given data $y_{1:T}$ and some parameter θ. Note that we have data $y_{1:T}$ from a model where we can express the conditional density of the latent variables $x_{0:T}$ given the data $y_{1:T}$ and θ in closed form up to a normalizing constant:

$$p(x_{0:T}|y_{1:T},\theta) = \frac{p(x_{0:T},y_{1:T}|\theta)}{p(y_{1:T}|\theta)} = \frac{g_0(x_0|\theta)\prod_{t=1}^{T} g_t(x_t|x_{t-1},\theta)f_t(y_t|x_t,\theta)}{p(y_{1:T}|\theta)}$$

where $p(y_{1:T}|\theta)$ cannot be computed analytically.

It is nontrivial to design and sample from a proposal distribution $q(x_{0:T}|y_{1:T},\theta)$ that will give reasonable variance in reasonable time. Furthermore, this problem has been motivated in some cases by situations in which one wants to perform inference on $x_{0:t}$, or more realistically x_t, "on-line", i.e. one wishes to compute conditional expectations with respect to the conditional density of x_t given $y_{1:t}$ and θ as t increases in order to make decisions in a timely fashion, precluding the conditioning of the proposal for x_j on y_{j+k} for large k. As such, it seems we should consider

constrained proposals of the form

$$q(x_{0:T}|y_{1:T},\theta) = q_0(x_0|\theta)\prod_{t=1}^{T} q_t(x_t|y_{1:t},x_{0:t-1},\theta)$$

An importance sampling algorithm in this setting, with N samples $x_{0:T}^{(1:N)}$ has unnormalized importance weights given by

$$w(x_{0:T}^{(i)}) = \frac{p(x_{0:T}^{(i)},y_{1:T}|\theta)}{q(x_{0:T}^{(i)}|y_{1:T},\theta)}$$

for $i \in \{1,\ldots,N\}$ where, due to the special nature of the proposal target densities we can rewrite this as

$$\frac{g_0(x_0^{(i)})}{q_0(x_0^{(i)})}\prod_{t=1}^{T}\frac{g_t(x_t^{(i)}|x_{t-1}^{(i)},\theta)f_t(y_t|x_t,\theta)}{q_t(x_t^{(i)}|y_{1:t},x_{0:t-1}^{(i)},\theta)}.$$

If one defines weights $w_0^{(i)} = \frac{g_0(x_0^{(i)})}{q_0(x_0^{(i)})}$ and

$$w_t^{(i)} = w_{t-1}^{(i)}\frac{g_t(x_t^{(i)}|x_{t-1}^{(i)},\theta)f_t(y_t|x_t^{(i)},\theta)}{q_t(x_t^{(i)}|y_{1:t},x_{0:t-1}^{(i)},\theta)}$$

for $t \in \{1,\ldots,T\}$ then this defines an incremental weighting scheme through time t for each sample. Of course, the particles and weights produced still come from an importance sampling algorithm and in this case for most models the variance of estimates of conditional expectations using these weights is prohibitively high. Indeed, treating data as randomly generated from the model, it is the case that the variance of the importance weights increases stochastically over time (Doucet et al. 2000).

The major innovation in SMC methodology is the introduction of a resampling step that introduces interactions between the samples or particles. A generic SMC algorithm is given in Algorithm 1.6. The resampling step at time t can be performed by sampling N indices $\mathcal{I}$ from a categorical distribution with parameter $W_t^{(1:N)}$ and then setting $x_{0:t}^{(1:N)} = \tilde{x}_{0:t}^{\mathcal{I}}$, although other schemes are possible. A common criterion for deciding whether or not to resample is to resample if the effective sample size (ESS) is less than some threshold value, e.g. $N/2$. This criterion was proposed in

Liu and Chen (1995) with the ESS at time t defined as

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

Algorithm 1.6 A Generic Sequential Monte Carlo Algorithm

1. At time $t = 0$.
   - For $i = 1, \ldots, N$, sample $\tilde{x}_0^{(i)} \sim q_0(\cdot)$
   - For $i = 1, \ldots, N$, evaluate the importance weights:
   $$w_0(\tilde{x}_0^{(i)}) = \frac{1}{N} \frac{g_0(\tilde{x}_0^{(i)}|\theta)}{q_0(\tilde{x}_0^{(i)}|\theta)}$$
   - For $i = 1, \ldots, N$, normalize the importance weights:
   $$W_0^{(i)} = \frac{w_0(\tilde{x}_0^{(i)})}{\sum_{j=1}^{N} w_0(\tilde{x}_0^{(j)})}$$
   - If some criterion is satisfied, resample N particles $x_0^{(1:N)}$ according to $(\tilde{x}_0^{(1:N)}, W_0^{(1:N)})$ and set $w_0^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.
   If not, set $(x_0^{(1:N)}, w_0^{(1:N)}) = (\tilde{x}_0^{(1:N)}, W_0^{(1:N)})$
2. For times $t > 0$.
   - For $i = 1, \ldots, N$, sample $\tilde{x}_t^{(i)} \sim q_t(\cdot|y_t, x_{t-1}^{(i)})$ and set $\tilde{x}_{0:t}^{(i)} \stackrel{\text{def}}{=} (x_{0:t-1}^{(i)}, \tilde{x}_t^{(i)})$
   - For $i = 1, \ldots, N$, evaluate the importance weights:
   $$w_t(\tilde{x}_{0:t}^{(i)}) = w_{t-1}^{(i)} \frac{f_t(y_t|\tilde{x}_t^{(i)}, \theta) g_t(\tilde{x}_t^{(i)}|\tilde{x}_{t-1}^{(i)}, \theta)}{q_t(\tilde{x}_t^{(i)}|y_t, \tilde{x}_{t-1}^{(i)}, \theta)}$$
   - For $i = 1, \ldots, N$, normalize the importance weights:
   $$W_t^{(i)} = \frac{w_t(\tilde{x}_{0:t}^{(i)})}{\sum_{j=1}^{N} w_t(\tilde{x}_{0:t}^{(j)})}$$
   - If some criterion is satisfied, resample N particles $x_{0:t}^{(1:N)}$ according to $(\tilde{x}_{0:t}^{(1:N)}, W_t^{(1:N)})$ and set $w_t^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.
   If not, set $(x_{0:t}^{(1:N)}, w_t^{(1:N)}) = (\tilde{x}_{0:t}^{(1:N)}, W_t^{(1:N)})$.

The intuition behind resampling is that particles with low weights are replaced by particles with large weights, in general, and hence computational effort is concen-

trated on those particles that have been good approximations of the target density so far. Furthermore, the replacement is done in a principled fashion, such that the expected number of replicates of a particle is proportional to its weight. Indeed, in Chapter 5 we will see that SMC methods are an elegant and principled form of importance sampling on an extended space. A central limit theorem that further justifies the methodology in some settings is given in Del Moral (2004) and Chopin (2004).

Important products of Algorithm 1.6 are empirical approximations of the distribution with density $p(x_{0:t}|y_{1:t}, \theta)$ for any $t \in \{0, \ldots, T\}$ and an approximation of $p(y_{1:t}|\theta)$ for $t \in \{1, \ldots, T\}$. These are given by

$$\hat{P}^N(dx_{0:t}|y_{1:t}) = \sum_{i=1}^{N} W_t^{(i)} \delta_{\tilde{x}_{0:t}^{(i)}}(dx_{0:t})$$

at time t and

$$\hat{p}^N(y_{1:t}|\theta) = \prod_{j=1}^{t} \sum_{i=1}^{N} w_j(\tilde{x}_{0:j}^{(i)})$$

where the former is immediate and the latter can be derived as an approximation to $p(y_{1:t}|\theta)$ using the decomposition $p(y_{1:t}|\theta) = p(y_1|\theta) \prod_{j=2}^{t} p(y_j|y_{1:j-1}, \theta)$. Note that the empirical approximation of the distribution immediately after resampling contributes higher variance to estimates of conditional expectations if used, and should hence be avoided.

1.2.4.2 SMC Samplers

Following the success of SMC methods for dynamic problems, Del Moral et al. (2006) laid the foundations for use of SMC methodology for arbitrary target densities. This foundation also exemplifies how SMC methods are an alternative type of population-based Monte Carlo method to population-based MCMC. Given a density π that we wish to sample from, one can define a sequence of auxiliary distributions via $\pi_{0:T-1}$ and set $\pi_T = \pi$. Using SMC, one can approximate each of these distributions in turn. Assume we can compute each density π_i up to a normalizing constant, i.e. we

can compute

$$\gamma_i(x) = \pi(x) \int_{\mathcal{X}} \gamma_i(z) dz$$

Now define a set of Markov transition kernels $\{K_i\}_{i=1}^T$ which will be used to 'move' particles at times $t = 0, \ldots, T-1$. The proposal density of a particle $x_{0:t}$ at time t is given by

$$\eta_t(x_{0:t}) = \eta(x_0) \prod_{j=1}^{t} K_j(x_{j-1}, x_j)$$

A necessary ingredient for an SMC approach is to be able to weight each particle at each time, and to be able to compute these weights incrementally. The approach in Del Moral et al. (2006) identifies a suitable set of extended target densities $\{\tilde{\pi}_t\}_{t=0}^T$ that accomplish this. In particular, let

$$\tilde{\pi}_t(x_{0:t}) = \pi_t(x_t) \prod_{j=0}^{t-1} L_j(x_{j+1}, x_j)$$

where $\{L_j\}_{j=0}^{T-1}$ can be chosen arbitrarily. Then it follows that

$$w_t(x_{0:t}) = \frac{\tilde{\pi}_t(x_{0:t})}{\eta_t(x_{0:t})}$$

satisfies

$$w_t(x_{0:t}) = w_{t-1}(x_{0:t-1}) \frac{\pi_t(x_t) L_{t-1}(x_t, x_{t-1})}{\pi_{t-1}(x_{t-1}) K_t(x_{t-1}, x_t)}$$

which immediately suggests Algorithm 1.7.

In the algorithm, it suffices to compute γ_t in lieu of π_t as the unknown normalizing constant affects each particle's weights proportionally. In addition to a particle approximation of each density π_t via, at time t,

$$\hat{P}_t^N(dx_t) = \sum_{i=1}^{N} W_t^{(i)} \delta_{\tilde{x}_t^{(i)}}(dx_t)$$

the quantity

$$\frac{\widehat{Z_t}}{Z_0}^N = \prod_{j=1}^{t} \sum_{i=1}^{N} w_j(\tilde{x}_{0:j}^{(i)})$$

is an estimate of the ratio of the normalizing constants for γ_t and γ_0.

Algorithm 1.7 A Generic SMC Sampler

1. At time $t = 0$.

   - For $i = 1, \ldots, N$, sample $\tilde{x}_0^{(i)} \sim \eta(\cdot)$
   - For $i = 1, \ldots, N$, evaluate the importance weights:

$$w_0(\tilde{x}_0^{(i)}) = \frac{1}{N} \frac{\gamma_0(\tilde{x}_0^{(i)})}{\eta(\tilde{x}_0^{(i)})}$$

   - For $i = 1, \ldots, N$, normalize the importance weights:

$$W_0^{(i)} = \frac{w_0(\tilde{x}_0^{(i)})}{\sum_{j=1}^N w_0(\tilde{x}_0^{(j)})}$$

   - If some criterion is satisfied, resample N particles $x_0^{(1:N)}$ according to $(\tilde{x}_0^{(1:N)}, W_0^{(1:N)})$ and set $w_0^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.
     If not, set $(x_0^{(1:N)}, w_0^{(1:N)}) = (\tilde{x}_0^{(1:N)}, W_0^{(1:N)})$

2. For times $t > 0$.

   - For $i = 1, \ldots, N$, sample $\tilde{x}_t^{(i)} \sim K_t(x_{t-1}^{(i)}, \cdot)$ and set $\tilde{x}_{0:t}^{(i)} \stackrel{\text{def}}{=} (x_{0:t-1}^{(i)}, \tilde{x}_t^{(i)})$
   - For $i = 1, \ldots, N$, evaluate the importance weights:

$$w_t(\tilde{x}_{0:t}^{(i)}) = w_{t-1}^{(i)} \frac{\gamma_t(\tilde{x}_t^{(i)}) L_{t-1}(\tilde{x}_t^{(i)}, \tilde{x}_{t-1}^{(i)})}{\gamma_{t-1}(\tilde{x}_{t-1}^{(i)}) K_t(\tilde{x}_{t-1}^{(i)}, \tilde{x}_t^{(i)})}$$

   - For $i = 1, \ldots, N$, normalize the importance weights:

$$W_t^{(i)} = \frac{w_t(\tilde{x}_{0:t}^{(i)})}{\sum_{j=1}^N w_t(\tilde{x}_{0:t}^{(j)})}$$

   - If some criterion is satisfied, resample N particles $x_{0:t}^{(1:N)}$ according to $(\tilde{x}_{0:t}^{(1:N)}, W_t^{(1:N)})$ and set $w_t^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.
     If not, set $(x_{0:t}^{(1:N)}, w_t^{(1:N)}) = (\tilde{x}_{0:t}^{(1:N)}, W_t^{(1:N)})$.

A popular, easy to implement and versatile choice for each K_t is to use a π_t-invariant MCMC kernel. In this case, one possible and computationally available choice for L_{t-1} is

$$L_{t-1}(x_t, x_{t-1}) = \frac{\pi_t(x_{t-1})K_t(x_{t-1}, x_t)}{\pi_t(x_t)}$$

since for this choice the weight $w_t(x_{0:t})$ has the form

$$w_t(x_{0:t}) = w_{t-1}(x_{0:t-1})\frac{\pi_t(x_{t-1})}{\pi_{t-1}(x_{t-1})}.$$

Although this choice of backwards kernel L_{t-1} is somewhat unappealing due to the lack of dependence on the rate of convergence of K_t, it provides a useful default and allows for the use of complex proposal kernels that take advantage of the rich methodology that has already been developed for MCMC. It is also well documented that in this case, since the weights do not depend on the proposed points, it is advantageous to perform the resampling step before moving the particles when a resampling step is performed. Because this particular implementation of an SMC sampler is used often in this thesis, it is presented in Algorithm 1.8.

For this algorithm, the particle approximation of each density π_t is given by

$$\hat{P}_t^N(dx_t) = \sum_{i=1}^{N} W_t^{(i)} \delta_{x_t^{(i)}}(dx_t)$$

and the quantity

$$\frac{\widehat{Z_t}}{Z_0}^N = \prod_{j=1}^{t}\sum_{i=1}^{N} \tilde{w}_j^{(i)}$$

is an estimate of the ratio of the normalizing constants for γ_t and γ_0.

1.3 Many-Core Computation

A key theme in this thesis is the recent trend in computer architecture from single-core to multi-core and further to many-core or massively multi-core processors (Sutter 2005; Asanovic et al. 2006). From the last half of the 20th Century until today, the number of transistors that can be placed on an integrated circuit has increased

Algorithm 1.8 An SMC Sampler with MCMC Kernels

1. At time $t = 0$.
   - For $i = 1, \ldots, N$, sample $\tilde{x}_0^{(i)} \sim \eta(\cdot)$
   - For $i = 1, \ldots, N$, evaluate the importance weights:
   $$\tilde{w}_0^{(i)} = w_0(\tilde{x}_0^{(i)}) = \frac{1}{N} \frac{\gamma_0(\tilde{x}_0^{(i)})}{\eta(\tilde{x}_0^{(i)})}$$
   - For $i = 1, \ldots, N$, normalize the importance weights:
   $$W_0^{(i)} = \frac{w_0(\tilde{x}_0^{(i)})}{\sum_{j=1}^N w_0(\tilde{x}_0^{(j)})}$$
   - If some criterion is satisfied, resample N particles $x_0^{(1:N)}$ according to $(\tilde{x}_0^{(1:N)}, W_0^{(1:N)})$ and set $w_0^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.
   If not, set $(x_0^{(1:N)}, w_0^{(1:N)}) = (\tilde{x}_0^{(1:N)}, W_0^{(1:N)})$
2. For times $t > 0$.
   - For $i = 1, \ldots, N$, evaluate the importance weights:
   $$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \frac{\gamma_t(x_{t-1}^{(i)})}{\gamma_{t-1}(x_{t-1}^{(i)})}$$
   - For $i = 1, \ldots, N$, normalize the importance weights:
   $$W_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^N \tilde{w}_t^{(j)}}$$
   - If some criterion is satisfied, resample N particles $\tilde{x}_{t-1}^{(1:N)}$ according to $(x_{t-1}^{(1:N)}, W_t^{(1:N)})$ and set $w_t^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.
   If not, set $(\tilde{x}_{t-1}^{(1:N)}, w_t^{(1:N)}) = (x_{t-1}^{(1:N)}, W_t^{(1:N)})$.
   - For $i = 1, \ldots, N$, sample $x_t^{(i)} \sim K_t(\tilde{x}_{t-1}^{(i)}, \cdot)$.

exponentially. Through a variety of mechanisms, this had allowed central processing unit (CPU) manufacturers to make CPU execution times for serial code decrease exponentially until only a few years ago. Unfortunately, issues with heat dissipation, power consumption and current leakage in single-core processors have compromised this trend, forcing manufacturers to use the still increasing number of transistors on a chip by introducing separate cores onto the same die. Such multi-core processors are capable of executing different threads, or streams of instructions, simultaneously. As a result, only programs that have multiple threads that can be executed simultaneously can benefit from the increasing number of cores available on processors today, though improvements in the performance of each core continue to be made.

This emerging trend has important implications for the practice of Bayesian statistics, which relies heavily on algorithms from computational statistics such as those discussed in Section 1.2. Foremost is the idea that next-generation computers require next-generation methodology that is ideally suited to their capabilities, at least for problems that have yet to be solved for ostensibly computational reasons. The exact nature of the current architectural trend suggests that this means we are obliged to devise algorithms that are parallelizable, in some sense, so as to benefit from increasing numbers of processor cores. A distinction, however, should be made between the algorithmic implications of a classical distributed computing or idealized parallel architecture and the type of architecture that we can expect to have at our disposal in the coming years.

1.3.1 Parallel & Distributed Computing

From a very general perspective, a framework in which arbitrary calculations can be performed in parallel and have their results instantaneously available to any other processor should be able to provide significant speedups for large classes of algorithms. However, modeling any real system of multiple processors in such a way belies physical constraints that can severely affect the reliability of the runtime analysis of algorithms. Two particular types of systems that have been used extensively in the past are broadly termed parallel and distributed systems, where the former refers to a system in which all processors have access to shared memory

and the latter refers to a system in which memory is not shared and information must be passed between processors by some alternative, usually slower, mechanism.

A major issue in distributed computing systems is the frequency and amount of information-passing between computing nodes. It is quite possible to split work between nodes in a cluster and have the total run-time exceed the amount that would be needed for the work to run on a single node alone. This occurs, generally, if the splitting of the work does not ensure that the amount of information that needs to be accessed by the various nodes is small. Scalability is often not a problem for distributed systems but they can be expensive and are often a shared resource amongst researchers. On the other hand, an advantage of distributed systems is often that different nodes can be highly specialized for different types of tasks.

In parallel computing, the first concern is alleviated somewhat by access to shared memory, although any memory access often remains a considerable bottleneck that cannot be ignored. However, in many such systems expense and scalability have been an issue. In addition, often the processors are homogeneous, precluding task specialization on a hardware level. Multi-core and many-core systems fall under this umbrella term, with the possibility of heterogeneous systems via a distributed computing framework. A particular type of parallel computing architecture that holds special promise at this time for scientific computing is classified as a single instruction, multiple data (SIMD) system.

1.3.2 Single Instruction, Multiple Data

SIMD systems are a classification of computer architectures from Flynn's taxonomy (Flynn 1966). Such systems are characterized by being able to execute the same instruction on multiple inputs, or data, simultaneously and can be compared to single instruction, single data (SISD) and multiple instruction, multiple data (MIMD) systems. An SISD system executes a single instruction on a single piece of data at a time while an MIMD system can execute different instructions on different pieces of data in parallel. A SIMD architecture is clearly constrained compared to an MIMD system as distinct instructions cannot be executed in parallel but less restricted than

an SISD system, which has no parallel capabilities whatsoever. Note that these restrictions do not indicate that a particular type of architecture will be better for statistical computing in general, since the implementation of the architecture, type of algorithm one performs and cost are all important factors.

It is instructive to view different architectures as differing solutions to a resource allocation problem. Each implementation of an architecture requires an allocation of certain numbers of transistors to various tasks such as flow control, data processing units and memory components. For a SISD architecture, one typically devotes many transistors to flow control in order to optimize the amount of computation that can be done whereas on an SIMD architecture one relies on the parallel execution of instructions for computational efficiency and, importantly, shares flow control between all processor cores. This can leave many more transistors free to be used as data processing units and memory components. This should highlight the fact that different architectures give advantages only for specific types of algorithms. A data-parallel algorithm that is often executing identical instructions on different pieces of data is essentially making better use of a fixed number of transistors in an SIMD system compared to an SISD system. In contrast, an algorithm that is not data parallel will generally make better use of a fixed number of transistors in the more general-purpose SISD system. MIMD architectures are often quite complicated and considerably more expensive, with a cluster of workstations being the 'cheap' implementation of this type of architecture.

In Chapter 2 we will see that a powerful but inexpensive SIMD architecture is now available for general-purpose computing that can offer considerable improvements in computational power for certain types of data-parallel algorithms.

1.4 Thesis Outline

Chapter 2 introduces graphics cards as an accessible architectural prototype for many-core computational statistics algorithms, with a particular emphasis on the potential importance of population-based Monte Carlo methods as a result of emerg-

ing hardware trends. This chapter additionally provides motivation for the following chapters.

Chapter 3 introduces a hierarchical framework for constructing sparsity-inducing priors, which provides a Bayesian interpretation and generalization of many popular penalized optimization approaches to variable selection. With one particular prior from this framework, a full Bayesian analysis is performed with a sequence of priors using SMC methodology on a graphics card. Readers interested in the methodological contributions contained within this thesis might read this chapter more casually or skip it entirely. In the context of the rest of the thesis, it constitutes an example of both the computational benefits of isolating suitable auxiliary variables within a statistical model and the ability to accelerate a computationally demanding SMC algorithm on a many-core device.

Chapter 4 provides a fertile perspective on Metropolis-Hastings, auxiliary variables and pseudo-marginal approaches that provides insight into a host of existing reversible MCMC kernels as well as potential improvements in a variety of applications. The MCMC kernels in this chapter are often motivated by their ability to attain an improved rate of convergence when certain auxiliary variables can be generated in parallel. In addition, locally adaptive MCMC kernels are introduced which allow for the adaptation of the number of random variables sampled while ensuring that the resulting transition kernels' equilibrium distributions remain unchanged. This type of adaptation is perhaps the most important methodological contribution of the thesis. Applications to approximate Bayesian computation and Bayesian variable selection are presented.

Chapter 5 extends the perspective developed in Chapter 4 to particle methods, specifically SMC methodology, used as proposals within an MCMC kernel. Due to the dependency of the proposed samples, such methods can have improved computational performance even if the auxiliary variables are not generated in parallel, which could motivate their use more widely. The adaptation of the number of particles used in an SMC sampler is perhaps the most important general purpose algorithm introduced by the thesis, and the presentation of the material is in many ways geared toward its eventual appearance. The perspective is also applied to two other recent

MCMC schemes involving auxiliary variables to demonstrate the benefits of viewing MCMC kernels within this framework. Applications to parameter estimation in state-space models, high-dimensional Bayesian variable selection and MCMC with doubly intractable distributions are discussed.

Chapter 2

Many-Core Population-Based Monte Carlo Methods

This chapter introduces graphics cards as an accessible architectural prototype for many-core computational statistics algorithms and demonstrates their utility in a number of population-based Monte Carlo methods. A discussion of the relative amenability of the methods proposed for the applications chosen, and the opportunities presented by many-core computation for computational statistics is followed by a more general discussion on population-based methods and their future.

2.1 Introduction

Graphics processing units (GPUs) are integrated circuits originally developed and used as dedicated aids for calculating quantities related to real-time graphics rendering. Recently, however, there has been an emerging literature on their use for scientific computing. Examples include Stone et al. (2007) and Friedrichs et al. (2009), which discuss their use in molecular modelling and dynamics.

The potential of parallel processing to aid in statistical computing is well documented (see, e.g., Kontoghiorghes 2006). However, previous studies have dealt

mainly with distributed multi-core clusters of CPUs for implementation. In contrast, graphics cards for certain generic types of computation offer parallel processing speedups with advantages on a number of fronts, including:

- Cost: graphics cards are relatively cheap. The cards described in this chapter retail at under £150.
- Accessibility: graphics cards are readily obtainable from high street computer stores or over the internet.
- Maintenance: the devices are self-contained and can be hosted on conventional desktop and laptop computers.
- Speed: in line with multi-core CPU clusters, graphics cards offer significant speedup, albeit for a restricted class of scientific computing algorithms.
- Power: GPUs are low energy consumption devices compared to clusters of traditional computers, with a graphics card requiring around 200 Watts. While improvements in energy efficiency are application-specific, it is reasonable in many situations to expect a GPU to use around 10 per cent of the energy of an equivalent CPU cluster.
- Dedicated and local: the graphics cards slot into conventional computers offering the user ownership without the need to transport data externally.

The idea of splitting the computational effort of parallelizable algorithms amongst processors is certainly not new to statisticians. In fact, distributed systems and clusters of computers have been around for decades. Previous work on parallelization of MCMC methods on a group of networked computers include, among others, Sohn (1995), Rosenthal (2000) and Brockwell (2006). Rosenthal (2000) discusses how to deal with computers running at different speeds and potential computer failure while Sohn (1995) and Brockwell (2006) discuss the parallel implementation of a sequence of MCMC kernels by pre-computing multiple execution paths. The latency and bandwidth of communication in these systems make them suitable only in cases where communication between streams of computation, or threads, is infrequent and low in volume. In other words, while many algorithms involve computation

that could theoretically be distributed amongst processors, the overhead associated with distributing the work erases any speedup. In contrast, many-core processor communication has very low latency and very high bandwidth due to high-speed memory that is shared amongst the cores. Low latency here means the time for a unit of data to be accessed or written to memory by a processor is low whilst high bandwidth means that the amount of data that can be sent in a unit of time is high. For many algorithms, this makes parallelization viable where it previously was not. In addition, the energy efficiency of a many-core computation compared to a single-core or distributed computation can be improved. This is because the computation can both take less time and require less overhead.

We choose to investigate the speed up for the simulation of random variates from complex distributions, a common computational task when performing inference using Monte Carlo (see, e.g., Robert and Casella 2004). In particular, we focus on population-based MCMC methods and SMC methods for producing random variates as these are not algorithms that typically see significant speedup on clusters due to the need for frequent, high-volume communication between computing nodes. For the examples we consider, we find that computation time can be significantly lowered for all applications, and drastically lowered in some cases. This means that we can obtain the samples we want in seconds instead of hours and minutes instead of days. The types of speedup observed are dependent on the ability of the methods to be parallelized. In particular, speedup increases with the number of auxiliary distributions in population-based MCMC and the number of particles in SMC until some device-specific capacity is reached.

The algorithms are implemented using the Compute Unified Device Architecture (CUDA) and make use of GPUs which support this architecture. CUDA offers a fairly mature development environment via an extension to the C programming language. We estimate that a programmer proficient in C should be able to code effectively in CUDA within a few weeks of dedicated study. For our applications we use CUDA version 2.1 with an NVIDIA GTX 280 as well as an NVIDIA 8800 GT. The GTX 280 has 30 multiprocessors while the 8800 GT has 14 multiprocessors. For all current NVIDIA cards, a multiprocessor comprises 8 arithmetic logic units (ALUs), 2 special units for transcendental functions, a multithreaded instruction

unit and on-chip shared memory. In addition to having more multiprocessors than the 8800 GT, each GTX 280 multiprocessor itself has more registers, can support more active threads and includes one double-precision ALU. For single-precision floating point computation, one can think of the GTX 280 as having 240 (30×8) and the 8800 GT as having 112 (14×8) single processors respectively. At present, the retail price of the GTX 280 is just over double that of the 8800GT and it requires just over twice the power.

2.2 Graphics Cards

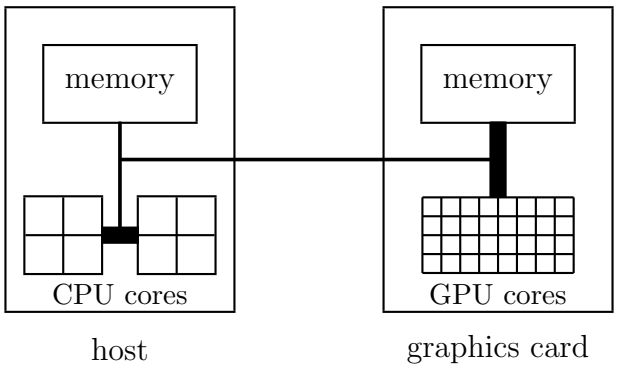


Figure 2.1: Link between host and graphics card. The thicker lines represent higher data bandwidth while the squares represent processor cores.

GPUs have evolved into many-core processing units, currently with up to 30 multiprocessors per card, in response to commercial demand for real-time graphics rendering, and independently of demand for many-core processors in the scientific computing community. As such, the architecture of GPUs is very different to that of conventional central processing units (CPUs). An important difference is that GPUs devote proportionally more transistors to ALUs and less to caches and flow control in comparison to CPUs. This makes them less general purpose but highly effective for data-parallel computation with high arithmetic intensity, i.e. computations where the same instructions are executed on different data elements and where the ratio of arithmetic operations to memory operations is high. This single instruction, multiple data (SIMD) architecture puts a heavy restriction on the types of computation that optimally utilize the GPU but in cases where the architecture is suitable it reduces overhead.

```
__global__ void importance_sample(int N, float* d_array,
        float* d_array_out) {
    // thread id = threads per block * block id + thread id in block
    const int tid = blockDim.x * blockIdx.x + threadIdx.x;
    // total number of threads = threads per block * number of blocks
    const int tt = blockDim.x * gridDim.x;
    int i;
    float w, x;
    for (i = tid; i < N; i += tt) {
        x = d_array[i];
        w = target_pdf(x) / proposal_pdf(x);
        d_array_out[i] = phi(x) * w;
    }
}
```

Figure 2.2: Kernel that evaluates an importance weight and test function

Figure 2.1 gives a visualization of the link between a host machine and the graphics card, emphasizing the data bandwidth characteristics of the links and the number of processing cores. A program utilizing a GPU is hosted on a CPU with both the CPU and the GPU having their own memory. Data is passed between the host and the device via a standard memory bus, similar to how data is passed between main memory and the CPU. The memory bus between GPU memory and the GPU cores is both wider and has a higher clock rate than a standard bus, enabling much more data to be sent to the cores than the equivalent link the host. This type of architecture is ideally suited to data-parallel computation since large quantities of data can be loaded into registers for the cores to process in parallel. In contrast, typical computer architectures use a cache to speed up memory accesses using locality principles that are generally good but do not fully apply to data-parallel computations, with the absence of temporal locality most notable.

2.2.1 Programming with Graphics Cards

CUDA provides the interface to compliant GPUs by extending the C programming language. Programs compiled with CUDA allow computation to be split between the CPU and the GPU. In this sense, the GPU can be treated as an additional, specialized processor for data-parallel computation. In the following text, host code refers to code that is executed on the CPU whilst device code is code that is executed on the GPU. We present a simple example in Figures 2.2-2.4, explained below, that

```
__device__ float target_pdf(float x) {
    return 1.0f / sqrtf(2 * PI) * exp(-(x - 1.5) * (x - 1.5) / 0.5f)
         + 1.0f / sqrtf(2 * PI) * exp(-(x + 1)   * (x + 1)   / 0.5f);
}

__device__ float proposal_pdf(float x) {
    return 1.0f / sqrtf(2 * PI) * exp(-x * x / 2.0f);
}

__device__ float phi(float x) {
    return x * x;
}
```

Figure 2.3: Device functions for evaluating the target density, the proposal density and the test function.

computes a classical importance sampling estimate (see Section 1.2.2.2). In the code snippets, keywords in the C language are in bold face whilst CUDA keywords are both bold and italicized. A line beginning with a "//" is a comment and is ignored by the compiler.

CUDA allows users to define special functions, called kernels, that are called by the host code to be executed in parallel on the GPU by a collection of threads. Figure 2.2 shows an example of a kernel function, which can be invoked in host code using the syntax

```
importance_sample<<<nb,nt>>>(N, d_array, d_array_out);
```

where `nb` is the number of blocks of threads and `nt` is the number of threads per block. The total number of threads created by this call is the product of `nb` and `nt` and one can think of each thread as being a single stream of computation. For most kernels, the numbers of threads and blocks can be changed to tune performance on different cards or with different data. A more detailed description of blocks and threads and their relation to the hardware is given in Section 2.2.2.

A kernel is defined with the `__global__` qualifier. Kernels are special in that they are always invoked in parallel with the numbers of blocks and threads specified and have a void return type. As such, program correctness depends only on how the threads invoked in the kernel call modify memory on the graphics card. In particular,

code must be written that guarantees the correct modifications to memory once all threads have completed, especially when threads interact during execution by reading and writing from shared memory locations. In Figure 2.2, a kernel is defined that takes as input an array of random values sampled from a proposal distribution and places, for each value, the product of the test function and the importance weight at that value in a separate array. One can see that each thread is responsible for `N / tt` values, assuming `N` is a multiple of `tt`. Within a kernel, special functions can be called that have been defined with the `__device__` qualifier. These functions can only be called by `__global__` functions or `__device__` functions themselves. In Figure 2.2, `target_pdf`, `proposal_pdf` and `phi` are examples of this, and their definitions are provided in Figure 2.3. The target is an equally weighted, two-component mixture of normals with equal variances of 0.25 and means at -1 and 1.5 while the proposal is a standard normal distribution. The test function squares its input so that the integral that is estimated is the expectation of the second moment of a random variable distributed according to the target density. In this particular kernel we see that each thread first computes its absolute thread identifier `tid` and the total number of threads `tt`. It then computes an importance weight and evaluates the test function for each value in `d_array` it is responsible for and stores the result in `d_array_out`. Since there is no thread interaction in this example kernel, it is reasonably straightforward to verify its correctness.

Figure 2.4 gives a snippet of code that is run on the host and completes our example. First, memory is allocated on both the host and the graphics card using the `malloc` and `cudaMalloc` functions respectively. The host function `populate_randn` then puts `N` standard normal random variates in `array`. These values are copied into the GPU array, `d_array`, via the `cudaMemcpy` function. In Figure 2.1, this is a transfer along the memory bus that connects host and graphics card memory. At this point, the kernel is called with 64 blocks of 128 threads per block. The `reduce` function is a CPU function that returns the sum of the elements in a GPU array. Of course, this function can itself invoke a GPU kernel. Finally, the importance sampling estimate is obtained by dividing this sum by `N` and memory is freed. Note that this code has been written so as to expose the most common functions that are used in GPU programming using CUDA. For example, it would be faster to create the random variates on the GPU itself but this would not have allowed any memory

```
int N = 16777216;

float h_sum, result;
float* d_array;
float* d_array_out;

float* array = (float*) malloc(N * sizeof(float));
cudaMalloc((void **) &d_array, N * sizeof(float));
cudaMalloc((void **) &d_array_out, N * sizeof(float));

populate_randn(array, N);

cudaMemcpy(d_array, array, N * sizeof(float), cudaMemcpyHostToDevice);

is<<<64,128>>>(N, d_array, d_array_out);
h_sum = reduce(N, d_array_out);
result = h_sum / N;

free(array);
cudaFree(d_array);
cudaFree(d_array_out);
```

Figure 2.4: Host code

transfer operations to be shown here.

This basic example highlights the most important characteristics of CUDA programs: memory management, kernel specification and kernel invocation. Memory management is a key component in algorithm design using graphics cards since there is often need for transfers between CPU and GPU memory as standard host functions can only access CPU memory and kernels can only access GPU memory. The fundamental memory operations are `cudaMalloc`, `cudaMemcpy` and `cudaFree`, which are GPU analogues to the standard C functions `malloc`, `memcpy` and `free`. Kernel specification requires ensuring that correct output will be given once all threads have returned. In the above example, it is clear that all concurrent threads will be executing the same instructions in parallel because there is no conditional branching, which occurs when different instructions are executed in concurrent threads based on the result of a data-dependent runtime comparison. Indeed, while it is possible to specify arbitrary conditional branches within a kernel, this can lead to slow performance since threads in a SIMD architecture execute sequentially when they are not executing the same instructions, which can be devastating to performance. An important constraint on kernel code that is not illustrated explicitly in the above code is that neither recursive functions nor function pointers can be

defined in device code. This is due to the fact that kernel functions are completely determined at compile time with `__device__` functions simply inlined, or inserted into the kernel, during compilation. With respect to kernel invocation, the number of threads and blocks assigned to each kernel can be decided at runtime in host code. This is useful since computation time can depend strongly on these numbers and optimal configurations will vary across graphics cards and features of the data. A final remark is that the level of abstraction provided by CUDA is close to the hardware operations on the device. While this often increases programming effort, it does remind programmers of the benefits of writing, for example, kernels with minimal interaction between threads and avoiding branching.

2.2.2 Blocks and Threads

CUDA abstracts the hardware of the GPU into blocks and threads to simultaneously provide a relatively simple view of the architecture to developers while still allowing a low-level abstraction of the hardware for performance reasons. One can generally think of each thread as being computed on a virtual processor. The block abstraction is necessary to provide the concept of a virtual microprocessor. Threads within a block are capable of more interaction than threads in separate blocks, mainly due to the fact that all threads in a block will be executed on the same microprocessor. As such, they have access to very fast, dynamically allocated, on-chip memory and can perform simple barrier synchronization. In Section 2.2.1, this advanced functionality is not required by the example kernel.

It is important to note that blocks and threads are still very much virtual constructs. At runtime, multiple blocks may be executed concurrently on the same multiprocessor. With respect to ALU execution, operations are performed on groups of 32 threads at a time, which allows each of the 8 scalar processors to perform 4 identical instructions in quick succession in an ideal setting. The group of 32 threads that executes simultaneously is called a warp.

2.2.3 Single Precision Issues

The current generation of GPUs is 4-8 times faster at single precision arithmetic than double precision. Although this ratio will decrease in the future, there will probably remain a factor 2 difference in speed, the same as for Intel CPUs when using SSE instructions. This raises the question of whether single precision arithmetic is adequate for statistical applications.

There are two particular areas in which care must be taken. The first concerns the much more limited range of single precision floating point numbers. Because of their 8-bit exponent, their magnitude must lie in the approximate range $[10^{-38}, 10^{+38}]$, whereas the magnitude of double precision variables is in the approximate range $[10^{-308}, 10^{+308}]$. Consequently, when working in single precision it is often necessary to work with log-likelihoods, rather than the likelihoods themselves though this is rarely a restriction for statisticians.

The second area of potential issues concerns the averaging of N floating point values, for $N \gg 1$. The simplest implementation uses an accumulator, to which the values are added one at a time. However, this may lead to a large increase in the error due to finite machine precision. When all of the values are of the same sign, the relative error is amplified by a factor of $O(N)$ in the worst case, and $O(\sqrt{N})$ in the more typical case where the rounding error at each step can be modelled as a random variable with zero mean. This behaviour is well understood Higham (1993, 2002) and the growth can be reduced to $O(\log N)$ by using a binary tree summation algorithm in which the values are summed in pairs, and then those new values are summed in pairs, and the process is repeated until a single value is obtained. This is the natural approach for the parallel implementation of a reduction operation. Example code is provided by NVIDIA on their CUDA website, and the implementation in our code is based on this.

Despite these concerns, single precision seems perfectly sufficient for the applications in this paper. The statistical variability due to the use of random numbers within the algorithms exceeds the perturbations due to finite machine precision.

2.2.4 GPU Parallelizable Algorithms

In general, if a computing task is well-suited to SIMD parallelization then it will be well-suited to computation on a GPU. In particular, data-parallel computations with high arithmetic intensity are able to attain maximum performance from a GPU, where arithmetic intensity refers to the ratio of arithmetic to memory operations. This is because the volume of very fast arithmetic instructions can hide the relatively slow memory accesses. It is crucial to determine whether a particular computation is data-parallel on the instruction level when determining suitability. From a statistical simulation perspective, integration via classical Monte Carlo or importance sampling are ideal computational tasks in a SIMD framework. This is because each computing node can produce and weight a sample in parallel, assuming that the sampling procedure and the weighting procedure have no conditional branches. If these methods do branch, speedup can be compromised by many computing nodes running idle while others finish their tasks. This can occur, for example, if the sampling procedure uses rejection sampling.

In contrast, if a computing task is not well-suited to SIMD parallelization then it will not be well-suited to computation on a GPU. In particular, task-parallel computations in which one executes different instructions on the same or different data cannot utilize the shared flow control hardware on a GPU and often end up running sequentially. Even when a computation is data-parallel, it might not give large performance improvements on a GPU due to memory constraints. This can be due to the number of registers required by each thread (see Sections 2.4.2 and 2.5) or due to the size and structure of the data necessary for the computation requiring large amounts of memory to be transferred between the host and the graphics card. The latter issue is analogous to the issue of thrashing in virtual memory systems and can occur, for example, when an algorithm iterates over a block of data that will not fit in memory.

There are also many computational tasks in statistical computing that are just difficult to parallelize. For example, standard Metropolis-Hastings MCMC with a single chain is difficult to parallelize in the general case because it is a naturally sequential algorithm. Parallelization of this type of method usually involves parallelization of

the target density evaluation as in Suchard and Rambaut (2009), the sampling from or evaluation of the proposal density or computation of multiple possible execution paths as in Sohn (1995); Brockwell (2006); Cui et al. (2011) as opposed to the definition of an MCMC algorithm that explicitly takes account of additional auxiliary variables produced in parallel.

The availability of new hardware suited to parallel computation motivates use of a new model of computation for developing and analyzing the efficacy of statistical algorithms. In some cases, existing algorithms will require little modification to take advantage of this technology whilst in others major changes may have to be made. There is also the potential for both previously impractical and novel algorithms to become important tools for statisticians.

2.2.5 Random Number Generation

One important aspect of any Monte Carlo simulation is the generation of pseudo-random numbers. Fortunately, many uniform pseudorandom number generators can be implemented efficiently in parallel. The key idea is that each thread generates a contiguous block of numbers within a single overall stream. The thread can jump to the start of its block of numbers using a "skip-ahead" algorithm which enables it to skip n places in $O(\log n)$ operations (see, e.g., L'Ecuyer et al. 2002). The uniform pseudorandom numbers can then be transformed to match various different output distributions as needed. In our applications we use a parallelized version of the multiple recursive generator MRG32k3a presented in L'Ecuyer (1999) as well as a parallelized version of a xorshift random number generator Marsaglia (2003). In the case of the xorshift random number generator, more time must be spent to compute the seeds for each thread before any computation is done but the random number generation itself is faster and the initialization can be done offline.

2.3 Parallelizing Population-Based Monte Carlo Methods

In this section we consider a number of sampling methods for which parallel implementations can be produced without significant modification. There is an abundance of statistical problems that are essentially computational in nature, especially in Bayesian inference, as noted in Chapter 1. We concentrate on the problem of sampling from a probability distribution whose density π we can compute pointwise and up to a normalizing constant, i.e. we can compute $\gamma(\cdot)$ where $\pi(\mathbf{x}) = \gamma(\mathbf{x})/Z$. A common motivation for wanting samples from π is so we can compute expectations of functions of random variables that are distributed according to π. If we denote such a function by ϕ, the expectation of interest is

$$\mathbb{E}_\pi[\phi] \stackrel{\text{def}}{=} \int_{\mathbf{x}\in\mathcal{X}} \phi(\mathbf{x})\pi(\mathbf{x})d\mathbf{x}. \tag{2.1}$$

A Monte Carlo estimate of this quantity is given by

$$\hat{I}_{MC} \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N} \phi(\mathbf{x}^{(i)})$$

where $\{\mathbf{x}^{(i)}\}_{i=1}^N$ are i.i.d. samples from π.

Clearly, we need samples from π in order to compute this estimate. In practice, however, we often cannot sample from π directly. There are two general classes of methods for dealing with this. The first are importance sampling methods, described in Section 1.2.2.2, where we weight samples from a proposal distribution in order to estimate an expectation of the form (2.1). For many problems, however, it is difficult to come up with a proposal distribution that provides estimates with sufficiently small variance for practical use.

The second general class of methods are MCMC methods, described in Section 1.2.3, in which we construct an ergodic π-stationary Markov chain and use the samples generated by this chain to estimate (2.1). The major issue with MCMC methods is that their convergence rate can be prohibitively slow in some applications.

There are many ways to parallelize sampling methods that are not the focus of this work. For example, naive importance sampling, like classical Monte Carlo, is intrinsically parallel. Therefore, in applications where we have access to a good importance density we can get linear speedup with the number of processors available. Similarly, in cases where MCMC converges rapidly we can parallelize the estimation of (2.1) by running separate chains on each processor. While these situations are hoped for, they are not particularly interesting from a parallel architecture standpoint because they can run equally well in a distributed system. Finally, we are not concerned with problems for which the computation of individual MCMC moves or importance weights are very expensive but themselves parallelizable. While the increased availability of parallel architectures will almost certainly be of help in such cases, the focus here is on potential speedups by parallelizing general sampling methods. An example of recent work in this area can be found in Suchard and Rambaut (2009), in which speedup is obtained by parallelizing the evaluation of individual likelihoods.

Much work in recent years has gone into dealing with the large constants in the variance of importance sampling estimates and slow convergence rates in MCMC and it is in such 'advanced' Monte Carlo methods that we direct our interest. One such class of advanced Monte Carlo methodology can be broadly classified as population-based, and includes both population-based MCMC and SMC, described in Sections 1.2.3.3 and 1.2.4. Such methods involve the generation of a population of samples from appropriately defined distributions that interact in some way and can be used to estimate expectations such as (2.1). These methods are ideal candidates for the investigation of speedup on a many-core architecture because while they are parallelizable, they are not trivially so and stand to benefit enormously from many-core architectures. In particular, the interactions between samples at various times make distributed computing architectures inappropriate for some applications using these methods. In contrast, an important property of many-core architectures is that the existence of shared memory significantly diminishes the run-time cost of core to core communication, which in this context can be interpreted loosely as particle or sample interaction.

2.3.1 Population-Based MCMC

Parallelization of population-based MCMC schemes involves few constraints on the general design of algorithms in this class. From Section 1.2.3.3, an algorithm requires the specification of the hybrid kernel K via M, $\pi_{1:M-1}$, $K_1^{(1:M)}$, R, $\mathcal{I}_{1:R}$ and $K_2^{(1:R)}$ and the way in which the constituent kernels are mixed and cycled. Here, $M-1$ is the number of auxiliary distributions, $\pi_{1:M-1}$ defines their densities, $K_1^{(1:M)}$ defines the component-wise kernels and R, $\mathcal{I}_{1:R}$ and $K_2^{(1:R)}$ define the number and type of interaction kernels.

The application of each of the component-wise kernels $K_1^{(1:M)}$ to each variable is often trivially parallelizable. However, the interaction kernels can in theory be sequential in nature. For a parallel implementation, it is beneficial for the $\mathcal{I}_j$'s chosen at a particular time to be disjoint as this allows a sequence of interaction kernels to be run in parallel. Of course, this also implies that different $\mathcal{I}_j$'s should be chosen at different times since otherwise there will be no interaction between the disjoint subsets of variables. Furthermore, in a SIMD parallel architecture, it is desirable to have sets of simultaneously executing kernels be nearly identical algorithmically. A final consideration for parallelization is that while speedup is generally larger when more computational threads can be run in parallel, it is not always helpful to increase M arbitrarily as this can affect the convergence rate of the chain. However, in situations where a suitable choice of M is dwarfed by the number of computational threads available, one can always increase the number of chains with target π to produce more samples.

It is prudent to note that the parallelization must not affect the definition of the kernel, or rather that one should ensure that the parallel implementation corresponds to a π-invariant MCMC kernel. As an example, one might define a kernel in which each component-wise kernel is applied to each component once. A parallel implementation of this cycle of kernels must then ensure that each kernel is applied exactly once.

2.3.2 Sequential Monte Carlo

The parallelization of SMC algorithms is reasonably straightforward. With reference to Section 1.2.4, the importance sampling and weighting steps are trivially parallelizable as they depend only on the local state of each particle. The resampling step, in which some particles are replicated and others destroyed depending on their normalized importance weights, typically comprises the construction of an empirical cumulative distribution function for the particles based on their importance weights followed by sampling from this N times, where N is the fixed number of particles used throughout the computation. While neither of these tasks is trivially parallelizable, they can benefit moderately from parallelization. However, the bulk of the speedup will generally come from the parallelization of the evolution and weighting steps. Therefore, using criteria like effective sample size (Liu and Chen 1995) to avoid resampling at every time step can also improve speedup.

An important distinction between SMC samplers and population-based MCMC, which can generally be employed using the same definition of auxiliary distributions, is subtle but practically important. In contrast to population-based MCMC, which produces a population of samples according to $\bar{\pi}$ at stationarity at each time, an SMC sampler starts by approximating the first auxiliary distribution and then each subsequent distribution in turn, using the samples produced before. Therefore, the samples generated by population-based MCMC at each time each have different stationary distributions and the samples from a particular chain over time provide an empirical approximation of that chain's target distribution while in an SMC sampler, the weighted samples generated at each time approximate one auxiliary target distribution and the true target distribution is approximated at the last time step. This difference is further discussed in Section 2.4.1.3.

2.4 Canonical Examples

To demonstrate the types of speed increase one can attain by utilizing GPUs, we apply each method to a representative statistical problem. We use Bayesian inference

for a Gaussian mixture model as an application of the population-based MCMC and SMC samplers, while we use a factor stochastic volatility state-space model to gauge the speedup of our parallel SMC method. We ran our parallel code on a computer equipped with an NVIDIA 8800 GT GPU, a computer equipped with an NVIDIA GTX 280 GPU and we ran reference single-threaded code on a Xeon E5420 / 2.5 GHz processor. The resulting processing times and speedups are given in Tables 2.1 - 2.3.

The applications we discuss here are representative of the types of problems that these methods are commonly used to solve. In particular, while the distribution of mixture means given observations is only one example of a multimodal distribution, it can be thought of as a canonical distribution with multiple well-separated modes. Therefore, the ability to sample points from this distribution is indicative of the ability to sample points from a wide range of multimodal distributions. Similarly, performance of a latent variable sampler in dealing with observations from a factor stochastic volatility model is indicative of performance on observations from reasonably well-behaved but non-linear and non-Gaussian state-space models.

2.4.1 Mixture Modelling

Finite mixture models are a very popular class of statistical models as they provide a flexible way to model heterogeneous data (see, e.g., McLachlan and Peel 2000). Let $\mathbf{y} = y_{1:m}$ denote i.i.d. observations where $y_j \in \mathbb{R}$ for $j \in \{1, \ldots, m\}$. A univariate Gaussian mixture model with k components states that each observation is distributed according to the mixture density

$$p(y_j|\mu_{1:k}, \sigma_{1:k}, w_{1:k}) = \sum_{i=1}^{k} w_i f(y_j|\mu_i, \sigma_i)$$

where f denotes the density of the univariate normal distribution. The density of $\mathbf{y}$ is then equal to $\prod_{j=1}^{m} p(y_j|\mu_{1:k}, \sigma_{1:k}, w_{1:k})$.

For simplicity, we assume that k, $w_{1:k}$ and $\sigma_{1:k}$ are known and that the prior distribution on μ is uniform on the k-dimensional hypercube $[-10, 10]^k$. We set $k = 4$,

$\sigma_i = \sigma = 0.55$, $w_i = w = 1/k$ for $i \in \{1, \ldots, k\}$. We simulate $m = 100$ observations for $\boldsymbol{\mu} = \mu_{1:4} = (-3, 0, 3, 6)$. The resulting posterior distribution for $\boldsymbol{\mu}$ is given by

$$p(\boldsymbol{\mu}|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\mu})\mathbf{1}_{[-10,10]^4}(\boldsymbol{\mu})$$

The main computational challenge associated with Bayesian inference in finite mixture models is the nonidentifiability of the components. As we have used exchangeable priors for the parameters $\mu_{1:4}$, the posterior distribution $p(\boldsymbol{\mu}|\mathbf{y})$ is invariant to permutations in the labelling of the parameters. Hence this posterior admits $k! = 24$ symmetric modes.

Generating N samples from such a posterior is a popular method for determining the ability of samplers to explore a high-dimensional space with multiple well-separated modes, which should all be represented in the samples. Basic random-walk MCMC and importance sampling methods typically fail to provide a correct approximation of the posterior for practical values of N (Celeux et al. 2000). It should be noted that while it might not be necessary to sample from all the symmetric modes in the case of a mixture model, the successful traversal of all the modes suggests that the sampler would succeed in traversing non-symmetric modes in other distributions, so long as symmetry is not exploited by the sampler.

2.4.1.1 Population-Based MCMC

We select the auxiliary distributions $\pi_{1:M-1}$ following the parallel tempering methodology, i.e. given M we define $\pi_i(\mathbf{x}) \propto \pi(\mathbf{x})^{\beta_i}$ with $0 \leq \beta_1 < \cdots < \beta_M = 1$. This class of auxiliary distributions is motivated by the fact that MCMC converges more rapidly when the target distribution is flatter. For this problem, we use the cooling schedule $\beta_i = (i/M)^2$ and a standard $\mathcal{N}(\mathbf{0}, I_k)$ random walk Metropolis-Hastings kernel for the component-wise kernels.

For the interaction kernels, we use only the basic exchange move (Geyer 1991;

Hukushima and Nemoto 1996): chains i and j swap their values with probability

$$\min\left\{1, \frac{\pi_i(\mathbf{x}^{(j)})\pi_j(\mathbf{x}^{(i)})}{\pi_i(\mathbf{x}^{(i)})\pi_j(\mathbf{x}^{(j)})}\right\}.$$

Further, we allow exchanges to take place only between variables with adjacent indices and in such a way that all moves can be done in parallel. We use $R = M - 1$ and $\mathcal{I}_j = \{j, j+1\}$ for $j \in \{1, \ldots, M-1\}$. The kernels $K_2^{(j)}$ are defined via the exchange move above and $\mathcal{I}_j$. At a particular time, each with probability half, the cycle of interaction kernels is either $K_2^{(1)}K_2^{(3)}\cdots K_2^{(M-1)}$ or $K_2^{(2)}K_2^{(4)}\cdots K_2^{(M)}$, where we have assumed that M is even for simplicity. This ensures that all the interaction moves in one cycle can be performed in parallel and also that each variable $\mathbf{x}^{(i)}$ is indirectly in communication with every other variable. While use of permutation or crossover moves would be appropriate for this particular model, we felt that they would detract from the ability to generalize our results to the case where the likelihood is not invariant to permutations of the labels.

Table 2.1: Running times for the Population-Based MCMC Sampler for various numbers of chains M.

M	CPU (mins)	8800GT (secs)	Speedup	GTX280 (secs)	Speedup
8	0.0166	0.887	1.1	1.083	0.9
32	0.0656	0.904	4	1.098	4
128	0.262	0.923	17	1.100	14
512	1.04	1.041	60	1.235	51
2048	4.16	1.485	168	1.427	175
8192	16.64	4.325	230	2.323	430
32768	66.7	14.957	268	7.729	527
131072	270.3	58.226	279	28.349	572

To test the computational time required by our algorithms we allow the number of chains to vary but fix the number of points we wish to sample from the marginal density $\pi_M = \pi$ at 8192. As such, an increase in the number of chains leads to a proportional increase in the total number of points sampled. Processing times for our code are given in Table 2.1, in which one can see that using 131072 chains is impractical on the CPU but entirely reasonable using the GPU. Figure 2.5(a) shows the estimated posterior density $p(\mu_{1:2}|\mathbf{y})$ from a set of 2^{20} MCMC samples from π_M with $M = 32768$, which is nearly identical to the estimated marginal posterior densities of any other pair of components of $\boldsymbol{\mu}$. This marginal density has 12 well-separated modes in $\mathbb{R}^2$ but it is worth noting that the joint density $p(\mu_{1:4}|\mathbf{y})$ has 24

well-separated modes in $\mathbb{R}^4$. Figure 2.5(b) shows the number of points from each mode for various values of M. We also computed the average number of iterations taken for the samplers to traverse all modes for the different values of M. For $M = 1$ and $M = 2$, the sampler did not traverse all the modes at all, while for values of M between 4 and 32 the traversal time decreased from 80000 to 10000, after which it was unchanged with increases in M. These numbers should be compared to $24 \times H_{24} \approx 91$ - the expected number of samples required to cover every mode if one could sample independently from π - where H_i is the ith harmonic number.

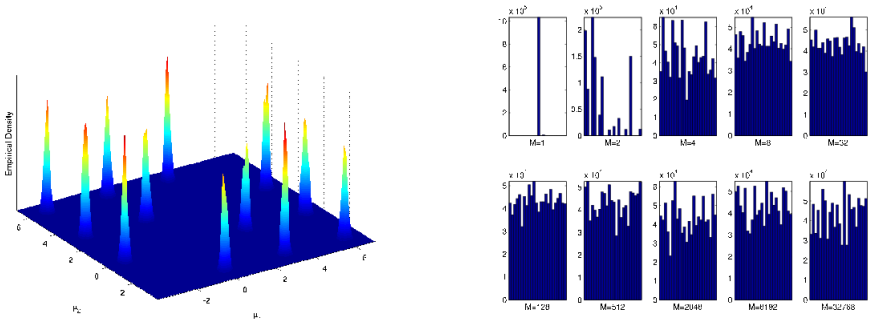


Figure 2.5: (a) Estimated marginal posterior density $p(\mu_{1:2}|\mathbf{y})$ from MCMC samples and (b) number of MCMC samples from each mode.

2.4.1.2 SMC Sampler

As with population-based MCMC, we use a tempering approach and the same cooling schedule, i.e. $\pi_t(\mathbf{x}) \propto \pi(\mathbf{x})^{\beta_t}$ with $\beta_t = (t/M)^2$ and $M = 200$. we use the uniform prior on the hypercube to generate the samples $\mathbf{x}_0^{(1:N)}$ and perform 10 MCMC steps with the standard $\mathcal{N}(\mathbf{0}, I_k)$ random walk Metropolis-Hastings kernel at every time step. We use the generic backwards kernel suggested in Crooks (1998), Neal (2001) and Del Moral et al. (2006) for the case where each kernel is π_t-stationary so that the unnormalized incremental importance weights are of the form $\pi_t(\mathbf{x}_{t-1})/\pi_{t-1}(\mathbf{x}_{t-1})$. Algorithm 1.8 in Section 1.2.4.2 provides further details.

We also ran the SMC sampler with no resampling at all, which for these settings corresponds to the annealed importance sampling (AIS) method proposed in Neal (2001); see also Crooks (1998) for a similar method in physics. The resulting samples were less successful at characterizing the multimodality of the posterior distribution.

Table 2.2: Running times for the Sequential Monte Carlo Sampler for various values of N.

N	CPU (mins)	8800GT (secs)	Speedup	GTX280 (secs)	Speedup
8192	4.44	1.192	223.5	0.597	446
16384	8.82	2.127	249	1.114	475
32768	17.7	3.995	266	2.114	502
65536	35.3	7.889	268	4.270	496
131072	70.6	15.671	270	8.075	525
262144	141	31.218	271	16.219	522

This is consistent with the numerical findings and theoretical results discussed in (Del Moral et al. 2006, Sections 3.5 and 4.2.3): when the MCMC kernels used at every time step mix reasonably well, it is beneficial to resample periodically. In addition, when ESS is used as a threshold for resampling in the SMC sampler, the resampling step takes very little time compared to the evolution and weighting of the particles simply because it happens so infrequently. As such, the running time of the SMC sampler compared to AIS is practically identical. Processing times for our code are given in Table 2.2. Figure 2.6(a) shows the estimated posterior density $p(\mu_{1:2}|\mathbf{y})$ from the SMC sampler with $N = 65536$. Figure 2.6(b) shows the number of points from each mode for various values of N.

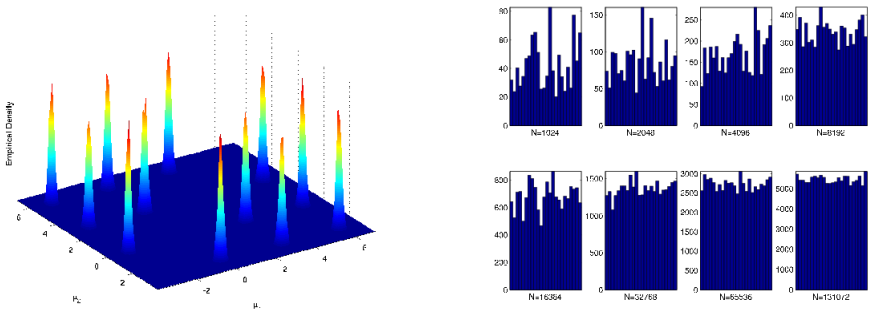


Figure 2.6: (a) Estimated marginal posterior density $p(\mu_{1:2}|\mathbf{y})$ from SMC samples and (b) effective number of SMC samples from each mode.

2.4.1.3 Comparison

While both methods are capable of exploring the posterior distribution for $\boldsymbol{\mu}$, there are important differences in how the methods make use of parallelization. In particular, the SMC sampler parallelizes across particles approximating the same auxiliary

distribution whilst the MCMC sampler parallelizes across auxiliary distributions at the same iteration. As such, to make full use of the graphics card the SMC sampler requires many particles while the MCMC sampler requires many auxiliary distributions. In most cases, however, one will be happy to use in excess of 8192 particles for SMC but one may not want to use in excess of 32768 auxiliary distributions. Indeed, for the application described above there seems to be no benefit in increasing the number of chains beyond 128, although this might also be due to the choice of cooling schedule and random walk variances. Furthermore, we utilized only the simplest information exchange and proposal moves in our samplers so as not to trivialize the problem. It should be noted, however, that there are situations in which a large number of intermediate temperatures are required for exchange acceptance probabilities to be greater than some preset value, for example when the dimension of the distribution of interest increases (Predescu et al. 2004). As mentioned in Section 2.4.1.1, we would like to emphasize that the use of very large numbers of chains is possible using these parallel methods with only a modest increase in computation time. An alternative scheme to more fully utilize available parallel resources could be to run many population-based MCMC algorithms in parallel that do not interact with each other, or to have multiple chains at the same temperature and utilize a more complex interaction operation.

The SMC sampler appears to be more efficient than the MCMC sampler for this problem. Indeed, with only 8192 particles the SMC sampler gives a reasonable representation of the posterior, taking only 597ms. The MCMC sampler requires around 2^{20} samples to give a reasonably uniform number of samples per mode, and this takes just over 2 minutes. In addition, although we have not done so here, it is possible with both the SMC and MCMC approaches to use the samples from the auxiliary distributions to estimate integrals of interest by computing appropriate importance weights. An interesting recent proposal on how to effectively combine estimates using such samples can be found in Gramacy et al. (2010).

For Bayesian inference in mixture models, there are many ways of dealing with the identifiability of the mixture parameters; Jasra et al. (2005) includes a review of these. It is worth mentioning that for this type of model, we can permute samples as a post-processing step or within an MCMC kernel so traversal of the modes can

be achieved trivially. The speedup of both methods is unaffected by the use of such mechanisms. In addition, the speedup is unaffected by increases in the number of observations since this only increases the amount of computation that each thread must do by a constant factor. Increasing the number of observations also has little effect on the difficulty for the sampler to move between modes since the modes are already well separated. Similarly, the speedup observed is robust to changes in the number of mixture components. The computation of each likelihood requires memory that is linear in the number of components, while the memory required per thread dictates the number of threads that can be run in parallel. However, as the number of components increases we usually have more observations, providing an opportunity to parallelize the computation of a single likelihood across multiple threads. This allows the amount of memory per thread to be significantly lower than if the complete likelihood was calculated by each thread and thus still allows many threads to be run in parallel.

2.4.2 Factor Stochastic Volatility

Many financial time series exhibit changing variance with time. A simple multivariate volatility model that allows us to capture the changing cross-covariance patterns of time series consists of using a dynamic latent factor model. In such models, all the variances and covariances are modelled through a low dimensional stochastic volatility structure driven by common factors (Liu and West 2000; Pitt and Shephard 1999). We consider here a factor stochastic volatility model most similar to that proposed in Liu and West (2000):

$$\begin{aligned}
\mathbf{y}_t|\mathbf{f}_t &\sim \mathcal{N}(\mathbf{B}\mathbf{f}_t, \mathbf{\Psi}) \\
\mathbf{f}_t|\mathbf{x}_t &\sim \mathcal{N}(\mathbf{0}, \mathbf{H}_t) \\
\mathbf{x}_t &\sim \mathcal{N}(\mathbf{\Phi}\mathbf{x}_{t-1}, \mathbf{U})
\end{aligned}$$

where $\mathbf{\Psi} \stackrel{\text{def}}{=} \text{diag}(\psi_1, \ldots, \psi_M)$, $\mathbf{H}_t \stackrel{\text{def}}{=} \text{diag}(\exp(\mathbf{x}_t))$ and $\mathbf{\Phi} \stackrel{\text{def}}{=} \text{diag}(\phi_1, \ldots, \phi_K)$.

Here, $\mathbf{f}_t$ is K-dimensional, $\mathbf{y}_t$ is M-dimensional and $\mathbf{B}$ is an $M \times K$ factor loading matrix with zero entries above the diagonal for reasons of identifiability. The latent

variable at each time step t is the K-dimensional vector $\mathbf{x}_t$. The likelihood of the data, $\mathbf{y}_t$, given $\mathbf{x}_t$ is Gaussian with

$$\mathbf{y}_t|\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{B}\mathbf{H}_t\mathbf{B}^T + \mathbf{\Psi})$$

We generate data for times $t = 1, \ldots, T = 200$, $M = 5$, $K = 3$, $\mathbf{x}_0 = \mathbf{0}, \psi_i = 0.5, i \in \{1, \ldots, M\}$, $\phi_i = 0.9, i \in \{1, \ldots, K\}$,

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & 0.5 & 1 \\ 0.2 & 0.6 & 0.3 \\ 0.8 & 0.7 & 0.5 \end{pmatrix} \text{ and } \mathbf{U} = \begin{pmatrix} 0.5 & 0.2 & 0.1 \\ 0.2 & 0.5 & 0.2 \\ 0.1 & 0.2 & 0.5 \end{pmatrix}$$

This is a simple example of a multivariate, non-linear and non-Gaussian state-space model for which particle filters are commonly employed to sample from the posterior $p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$. Processing times for our code are given in Table 2.3.

Table 2.3: Running time (in seconds) for the Sequential Monte Carlo method for various values of N.

N	CPU	8800GT	Speedup	GTX280	Speedup
8192	2.167	0.263	8	0.082	26
16384	4.325	0.493	9	0.144	30
32768	8.543	0.921	9	0.249	34
65536	17.425	1.775	10	0.465	37
131072	34.8	3.486	10	.929	37

The speedups obtained in this application are considerably less than for the mixture model example. This can be explained by lower arithmetic intensity, higher space complexity in each thread and increased resampling rate as compared to the SMC sampler example above. The mixture model likelihood calculation contains a compute-intensive product-sum operation involving 104 values whilst the factor stochastic volatility likelihood consists mainly of matrix operations. In the latter case, the speedup is independent of T but not the dimension of the observations since the amount of memory required per thread increases quadratically in the dimension of each observation. For example, we attained a speedup of 80 on the GTX

280 when running a particle filter for a multivariate stochastic volatility model with $M = K = 2$. The frequency of resampling is an issue with respect to speedup because it can typically only attain around 10 to 20 fold speedup for practical values of N, mainly due to the parallel scan operation. This potentially gives rise to tradeoffs in speedup between the transition and weighting steps and the time between resampling steps for some models, since more sophisticated proposal distributions that parallelize less cleanly might reduce the resampling rate. This type of performance, however, still provides considerable speedup and may be more representative of the type of speedup practitioners can expect in general.

2.4.3 Floating Point Precision

For all three algorithms discussed above, we ran identical algorithms with the same random numbers on the CPU using double precision floating point numbers and the resulting estimates of expectations of interest were affected by an order of magnitude less than the Monte Carlo variance of the estimates. The actual paths sampled, of course, were different due to the sensitivity of all of the algorithms to perturbations but this did not affect the ability of the samples to approximate the target distribution.

2.5 Discussion

The speedups for the population-based MCMC algorithm and the SMC sampler are tremendous. In particular, the evaluation of $p(\mathbf{y}|\boldsymbol{\mu})$ for the mixture-modelling application has high arithmetic intensity since it consists of a product-sum operation with 400 Gaussian log-likelihood evaluations involving only 104 values. In fact, because of the low register and memory requirements, so many threads can be run concurrently that SIMD calculation of this likelihood can be sped up by 500 times on the 8800 GT and 800 times on the GTX 280. However, the speedup attained for the standard SMC algorithm may be more representative of the kinds of gains one can expect in most applications with only reasonable arithmetic intensity. Even

so, speedups of 10 to 35 make many problems tractable that previously were not by reducing a week's worth of computation to a few hours. For example, estimation of static parameters in state-space models or the use of SMC proposals within MCMC can require thousands of runs, so a speedup of this scale can substantially reduce the computation time of such approaches (see, e.g., Andrieu et al. 2010). It is worth noting also that we can expect speedups in the vicinity of 500 with SMC if few resampling steps are required and each weighting step has small space complexity and moderate time complexity.

The GTX 280 GPU outperforms the 8800 GT GPU by a factor of about 2 in all situations in which the GPU is used to capacity. This is the case in all but the population-based MCMC algorithm, in which the number of threads is determined by the number of auxiliary distributions. The reason for this is simple: the algorithms presented are register-bound on the inputs given, in that the number of registers required by each thread is the critical quantity that bounds the number of threads that can be run concurrently. The GTX 280 has twice the number of registers per multiprocessor and more than twice the multiprocessors compared to the 8800 GT. Hence, one could expect more speedup on many-core chips with even more registers. In fact, further improvements could be made using multiple cards with large amounts of memory, configurations of which are now available in NVIDIA's Tesla line. These Tesla 'personal supercomputers' comprise 3 or more high-performance GPUs, each with 4GB of memory and a CPU with at least as much memory as the GPUs' combined. It is also possible to design algorithms that are memory-bound, though we have not encountered this in the context of Monte Carlo simulation. It is certainly possible that both register and memory limitations can affect the parallelizability of the mentioned algorithms when facing very high-dimensional problems. However, in such cases it is possible that alternative uses of many-core architecture can provide speedup in these situations.

The acceleration of the Monte Carlo methods discussed here have practical benefits not only to computation time but also to energy efficiency. A general purpose CPU allocates extra circuits and hence power to flow control and caching which is unnecessary for the types of computation described here. As such, reasonable decreases in power consumption can be realized by using specialized many-core

architectures like SIMD instead.

It should be noted that while we have used CUDA to implement the parallel components of algorithms, the results are not necessarily specific to this framework or to GPUs. It is expected that the many-core processor market will grow and there will be a variety of different devices and architectures to take advantage of. However, the SIMD parallelization technique and the sacrifice of caching and flow control for arithmetic processing is unlikely to disappear, particularly because when it is well-suited to a problem it will nearly always deliver considerable speedup. In addition, GPUs are affordable, off-the-shelf components that can be easily installed on a personal computer. Of particular interest is an emerging framework, the Open Computing Language (OpenCL), which provides a uniform programming environment for developers that enables them to write portable code for a variety of parallel devices, including GPUs and CPUs. For users who would like to see moderate speedup with very little effort, there is work being done to develop libraries that will take existing code and automatically generate code that will run on a GPU. An example of this is the Jacket engine for MATLAB code, created by Accelereyes.

The speedups attainable with many-core architectures have broad implications in the design, analysis and application of SMC and population-based MCMC methods. With respect to SMC, it allows more particles to be used for the same or even less computation time, which can make these samplers viable where they previously were not. When faced with designing a population-based MCMC sampler, the results expectedly show that there is little cost associated with increasing the number of auxiliary distributions until the GPU reaches the critical limit of threads it can run concurrently. After this, there is a doubling in the computation time when the number of chains is doubled. In our application, this does not occur until we have around 4096 auxiliary distributions. One might notice that this number is far larger than the number of processors on the GPU. This is due to the fact that even with many processors, significant speedup can be attained by having a full pipeline of instructions on each processor to hide the relatively slow memory reads and writes. Of course, we can expect this application-specific number to decrease when dealing with higher-dimensional distributions or those whose density evaluations require more registers or memory. Nevertheless, practitioners have more freedom to increase

the number of auxiliary distributions to achieve a faster rate of convergence as the computation time associated with each step is not as closely tied to this value as it is on a single-core processor. In both SMC and MCMC, it is also clear from this case-study that it is beneficial for each thread to use as few registers as possible since this determines the number of threads that can be run simultaneously. This may be of interest to the methodology community since it creates a space-time tradeoff that might be exploited in some applications.

A consequence of the space-time tradeoff mentioned above is that methods which require large numbers of registers per thread are not necessarily suitable for parallelization using GPUs. For example, operations on large, dense matrices that are unique to each thread can restrict the number of threads that can run in parallel and hence dramatically affect potential speedup. In cases where data is shared across threads, however, this is not an issue. For example, a mixture model with large amounts of data does not affect the number of registers required whilst increasing the number of components increases the number of registers required only linearly. In contrast, increasing the number of observed assets in a factor stochastic volatility model leads to a quadratic increase in the number of registers required, substantially affecting scalability in this regard. An increase in the number of observations itself has no effect on speedup. In principle, it is not the size of the data that matters but the space complexity of the algorithm in each thread that dictates how scalable the parallelization is.

The parallelization of the advanced Monte Carlo methods described here opens up challenges for both practitioners and for algorithm designers. There are already an abundance of statistical problems that are being solved computationally and technological advances, if taken advantage of by the community, can serve to make previously impractical solutions eminently reasonable and motivate the development of new methods. Having demonstrated that population-based methods can make use of many-core architectures, the following section explores in greater detail the analogous nature of SMC and population-based MCMC methods, both to clarify their shared features as well as their differences.

2.6 The Future of Population-Based Methods

The investigation into the types of gains in computational power that can be achieved with parallelizable algorithms on many-core architectures provide insight into which types of methodology in computational statistics will be best able to take advantage of 21st Century computing machines. While population-based algorithms are not the only types of Monte Carlo algorithms that will benefit, the results provide encouragement for the methodology community to invest time and energy in the improvement of such algorithms. Examples outside of population-based methods for Monte Carlo integration that will benefit from parallel architectures include classical Monte Carlo and standard importance sampling since they are embarrassingly parallel in nature. However, such approaches can often be improved by some limited form of particle, or sample, interaction, as in SMC. A more interesting example is in standard MCMC schemes where sampling from the proposal, or calculation of the acceptance ratio involves computation that can be parallelized. This last example is clearly application-specific but provides fertile ground for research into the design of such parallelized algorithms as well as the specification of suitably complex models that are computationally tractable in particular application areas.

This last section of this chapter contains some insights into the analogous nature of population-based MCMC and SMC methodologies, and motivates future research into their improvement. The main purpose of this material is to demonstrate that population-based MCMC and SMC methods are to a large extent analogous, despite important differences in the types of particle interactions they use within their respective populations. Since the publication of Del Moral et al. (2006), SMC samplers have been adopted for inference in a range of static problems that were previously largely approached using MCMC. This is a welcome addition to the suite of methodology available to statisticians, as the types of distributions we wish to sample from grow more and more complex. It is clear, in addition, that the benefits of population-based methods in a broader sense arise from principled interactions between the auxiliary random variables, or particles. We turn our attention to an application domain in which SMC methodology is dominant, for good reason, but for which population-based MCMC is applicable.

2.6.1 Population-Based MCMC in State-Space Models

SMC methodology is particularly suitable for inference in state-space models, although this suitability is especially pronounced in the case where the static parameters of the model are known. Using this terminology, the parameter θ in Section 1.1.2.2 is a static parameter whereas the latent variables $x_{0:T}$ are referred to as dynamic parameters.

2.6.1.1 Particle Filters as SMC Samplers

The first step in obtaining a population-based MCMC algorithm for state-space models involves the identification of the SMC sampler associated with the classical particle filter. Recall that in classical SMC, the target distributions are generally thought of as being of increasing dimension, i.e. we define $\pi_t(x_{0:t}) = p(x_{0:t}|y_{1:t})$ for $t = 0, \ldots, T$. However, we can think equivalently of defining a distribution on $\mathcal{X}^{T+1}$, $\tilde{\pi}_t$ via

$$\tilde{\pi}_t(x_{0:T}) = \pi_t(x_{0:t})\psi_t(x_{t+1:T})$$

where the definition of ψ_t is essentially arbitrary and of no consequence. The sequence of densities on $\mathcal{X}^{T+1}$ that we wish the SMC sampler to target is then $\{\tilde{\pi}_t\}_{t=0}^T$.

All that remains is to show what choices of kernels K_t and L_{t-1} correspond to the standard SMC algorithm. We will define, as usual, $\gamma_t(x_{0:t}) = p(x_{0:t}, y_{1:t}) \propto \pi_t(x_{0:t})$ and $\tilde{\gamma}_t(x_{0:T}) = \gamma_t(x_{0:t})\psi_t(x_{t+1:T})$. Assume that $x_{0:T}$ is a particle produced at time $t-1$ with weight w_{t-1} and we are proposing a particle $z_{0:T}$ at time t. Then, the choices

$$K_t(x_{0:T}, z_{0:T}) = \delta_{x_{0:t-1}}(z_{0:t-1})q_t(z_t|z_{t-1})\psi_t(z_{t+1:T})$$

and

$$L_{t-1}(z_{0:T}, x_{0:T}) = \delta_{z_{0:t-1}}(x_{0:t-1})\psi_{t-1}(x_{t:T})$$

imply the importance weight satisfies

$$\begin{aligned} w_t &= w_{t-1} \frac{\tilde{\gamma}_t(z_{0:T}) L_{t-1}(z_{0:T}, x_{0:T})}{\tilde{\gamma}_{t-1}(x_{0:T}) K_t(x_{0:T}, z_{0:T})} \\ &= w_{t-1} \frac{\gamma_t(z_{0:t})}{\gamma_{t-1}(x_{0:t-1}) q_t(z_t | x_{t-1})} \end{aligned}$$

which is equivalent to the weighting procedure using a particle filter. In particular, the usual state-space model setting has $\frac{\gamma_t(z_{0:t})}{\gamma_{t-1}(x_{0:t})} = p(x_t, y_t | x_{t-1})$.

Note that the choice of ψ_t's is arbitrary and plays no part in a practical implementation of the algorithm, since there is no need to actually produce the samples according to ψ_t when proposing $z_{0:T} \sim K_t(x_{0:T}, \cdot)$. However, for the sake of intuition, or perhaps for some purpose in inference, one could define these to be specific densities, as well as allowing ψ_t to define a conditional density given $x_{0:t}$. For example one could define $\psi_t(x_{t+1:T} | x_{0:t}) = p(x_{t+1:T} | x_t)$ so that $\tilde{\pi}_t(x_{0:T}) = p(x_{0:T} | y_{1:t})$. We now turn our attention to a population-based MCMC algorithm that uses the same set of auxiliary distributions.

2.6.1.2 Population-Based MCMC Approach

In the population-based MCMC (PB-MCMC) setup, we use the same definition of $\tilde{\pi}_t$ and so we want to define a Markov transition kernel for which $\bar{\pi}(x_{0:T}^{(0:T)}) = \prod_{t=0}^{T} \tilde{\pi}_t(x_{0:T}^{(t)})$ is the density of its equilibrium distribution. We focus first on the interaction kernel, which occurs between particles $x_{0:T}^{(t-1)}$ and $x_{0:T}^{(t)}$ where $t \in \{1, \ldots, T\}$.

We define a Metropolis-Hastings exchange move in which first we sample $z_{0:T}^{(t-1)} \sim K_t(x_{0:T}^{(t-1)}, \cdot)$ and $z_{0:T}^{(t)} \sim L_{t-1}(x_{0:T}^{(t)}, \cdot)$ and then attempt the move $(x_{0:T}^{(t-1)}, x_{0:T}^{(t)}) \to$

$(z_{0:T}^{(t)}, z_{0:T}^{(t-1)})$. The acceptance probability is given by

$$\begin{aligned}\alpha &= \min\left\{1, \frac{\tilde{\gamma}_t(z_{0:T}^{(t-1)})\tilde{\gamma}_{t-1}(z_{0:T}^{(t)}))K_t(z_{0:T}^{(t)}, x_{0:T}^{(t)})L_{t-1}(z_{0:T}^{(t-1)}, x_{0:T}^{(t-1)})}{\tilde{\gamma}_{t-1}(x_{0:T}^{(t-1)})\tilde{\gamma}_t(x_{0:T}^{(t)})K_t(x_{0:T}^{(t-1)}, z_{0:T}^{(t-1)})L_{t-1}(x_{0:T}^{(t)}, z_{0:T}^{(t)})}\right\} \\ &= \min\left\{1, \frac{\gamma_t(z_{0:T}^{(t-1)})\gamma_{t-1}(z_{0:T}^{(t)}))}{\gamma_{t-1}(x_{0:T}^{(t-1)})\gamma_t(x_{0:T}^{(t)})}\frac{q_t(x_t^{(t)}|x_{t-1}^{(t)})}{q_t(z_t^{(t-1)}|z_{t-1}^{(t-1)})}\right\} \\ &= \min\left\{1, \frac{p(z_t^{(t-1)}, y_t|x_{t-1}^{(t-1)})}{p(x_t^{(t)}, y_t|x_{t-1}^{(t)})}\frac{q_t(x_t^{(t)}|x_{t-1}^{(t)})}{q_t(z_t^{(t-1)}|x_{t-1}^{(t-1)})}\right\}\end{aligned}$$

If, for example, $q_t(x_t|x_{t-1}) = p(x_t|x_{t-1})$ then this is analogous to the bootstrap filter in SMC and the term not equal to 1 in the acceptance probability reduces to $p(y_t|z_t^{(t-1)})/p(y_t|x_t^{(t)})$. If, on the other hand, $q_t(x_t|x_{t-1}) = p(x_t|x_{t-1}, y_t)$, i.e. we have chosen the locally optimal SMC proposal, then the term not equal to 1 in the acceptance probability reduces to $p(y_t|x_{t-1}^{(t-1)})/p(y_t|x_{t-1}^{(t)})$ and can be computed before the proposal is sampled.

In addition to the above Metropolis-Hastings exchange move, one can also use kernels that operate on only one of the auxiliary variables. To ensure irreducibility in the state-space models context, assuming $q_t(\cdot|x_{t-1})$ is positive on the support of $p(\cdot|x_{t-1}, y_t)$, it is sufficient to include additionally only a kernel that moves $x_0^{(0)}$ whose equilibrium distribution is $\tilde{\pi}_0$. In many cases, the density $p(x_0)$ can be sampled from exactly.

2.6.1.3 Static Parameter Estimation

We now consider the problem of static parameter estimation in state-space models. It is straightforward to extend the density of each variable to include a static parameter θ, i.e. we now define $\gamma_t(\theta, x_{0:T}) = p(\theta)p(x_{0:T}, y_{1:T}|\theta)$ where $p(\theta)$ denotes the prior density of θ, and similarly extend $\tilde{\gamma}_t$. Now assume that for the exchange step performed, θ is kept fixed in both K_t and L_{t-1}, i.e.

$$K_t((\theta, x_{0:T}), (\theta', z_{0:T})) = \delta_\theta(\theta')\delta_{x_{0:t-1}}(z_{0:t-1})q_t(z_t|z_{t-1}, \theta)\psi_t(z_{t+1:T})$$

and

$$L_{t-1}((\theta', z_{0:T}), (\theta, x_{0:T})) = \delta_{\theta'}(\theta')\delta_{z_{0:t-1}}(x_{0:t-1})\psi_{t-1}(x_{t:T})$$

In this case, the probability of an exchange move is given by

$$\alpha = \min\left\{1, \frac{p(z_t^{(t-1)}, y_t | x_{t-1}^{(t-1)}, \theta^{(t-1)})}{p(x_t^{(t)}, y_t | x_{t-1}^{(t)}, \theta^{(t)})} \frac{q_t(x_t^{(t)} | x_{t-1}^{(t)}, \theta^{(t)})}{q_t(z_t^{(t-1)} | x_{t-1}^{(t-1)}, \theta^{(t-1)})}\right\}$$

which, crucially, does not require the re-evaluation of γ_t or γ_{t-1} for a conflicting value of θ. As such, if we additionally include in our MCMC scheme kernels that operate on individual values of θ, then one can use this PB-MCMC approach to obtain samples from $p(\theta|y_{1:T})$.

2.6.1.4 Computational Issues

There are a great number of possibilities for the implementation of a PB-MCMC algorithm. The state space of the chain is essentially $\mathcal{X}^{(T+1)(T+2)/2}$ since the auxiliary variables defined by the densities $\{\psi_t\}_{t=0}^T$ are not involved in the algorithm. As such, if one was to update every relevant variable at every iteration of the MCMC scheme, this would involve a $O(T^2)$ cost per iteration, which is prohibitively expensive. Each exchange move, on the other hand takes $O(1)$ time, so it is feasible to perform $O(T)$ exchange moves per iteration. For the computations to be easily parallelizable, however, it is recommended that each variable is only involved in one exchange move per iteration. In parallel, for example, a scheme in which one performs an exchange move for each every adjacent pair in a specified order is considerably more costly than a scheme in which an exchange move is performed for $O(T/2)$ sets of non-overlapping pairs. This, of course, has an impact on how quickly variables can interact indirectly.

The publication of Andrieu et al. (2010), which allows for the use of SMC methods to make proposals within an MCMC algorithm, represents a major breakthrough in our ability to perform offline static parameter estimation in state-space models. Indeed, the algorithms presented therein provide a gold standard by which we can measure the effectiveness of alternative approaches to parameter estimation in this context.

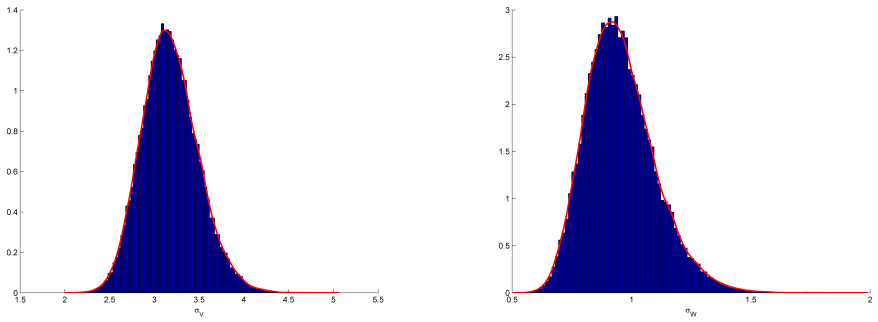
2.6.1.5 Results

We focus our attention on a popular toy example of a non-linear state-space model (Gordon et al. 1993; Kitagawa 1996):

$$X_t = \frac{X_{t-1}}{2} + 25\frac{X_{t-1}}{1+X_{t-1}^2} + 8\cos(1.2t) + V_t$$
$$Y_t = \frac{X_t^2}{20} + W_t$$

where $V_t \sim N(0, \sigma_V^2)$, $W_t \sim N(0, \sigma_W^2)$ and $X_0 \sim N(0, \sigma_V^2)$ and we adopt a prior on $\theta = (\sigma_V^2, \sigma_W^2)$ where $\sigma_V^2 \sim IG(1,1)$ and $\sigma_W^2 \sim IG(1,1)$, where IG denotes the inverse-gamma density. We let $T = 100$ and simulated $y_{1:T}$ according to this model, using $\theta = (10, 1)$.

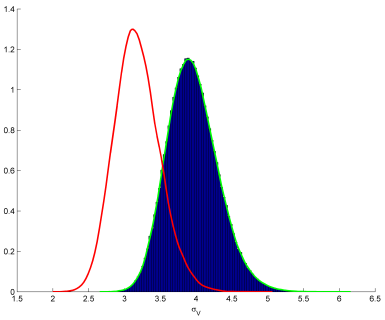
A total of four algorithms are run on this example, a standard MCMC sampler, a minimalist PB-MCMC sampler, a PB-MCMC sampler that for each subchain, additionally attempts to move a single component selected uniformly at random and a particle Gibbs sampler that uses 500 particles. Each algorithm uses a Gibbs step to update its value(s) of θ at every iteration. The initial value of θ for each sampler is chosen to be $(4, 4)$.



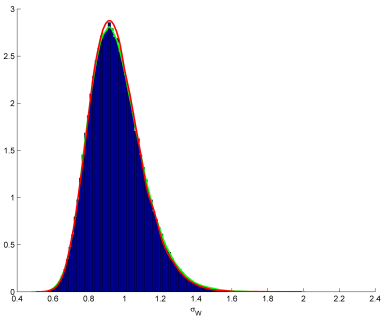
(a) Histogram/density estimate for σ_V (b) Histogram/density estimate for σ_W

Figure 2.7: Particle Gibbs histograms

Output of the simulations can be seen in Figures 2.7-2.11. In running the simulations, it became clear that the particle Gibbs sampler converges almost immediately to the stationary distribution of θ. This is due to the fact that for this model and with $T = 100$, the particle filter produces almost perfect samples from $p(x_{0:T}|\theta)$. For comparison, its density estimates are overlaid in red on the plots of density es-

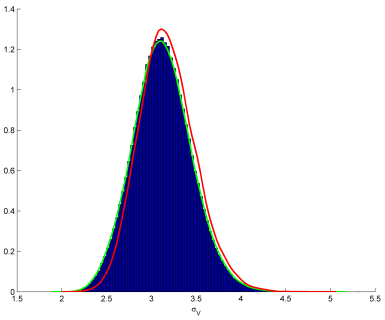


(a) Histogram/density estimate for σ_V

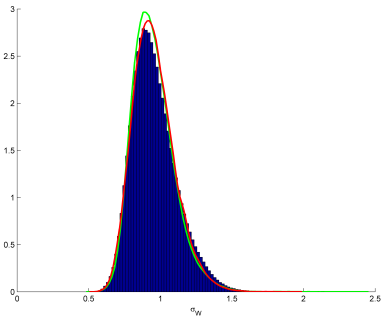


(b) Histogram/density estimate for σ_W

Figure 2.8: Standard MCMC histograms

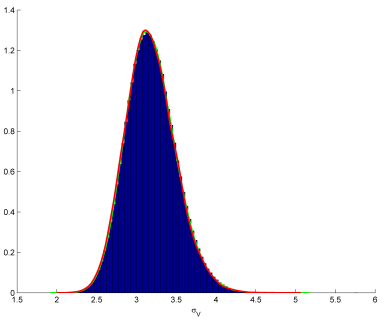


(a) Histogram/density estimate for σ_V

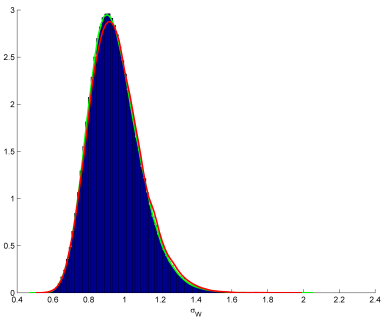


(b) Histogram/density estimate for σ_W

Figure 2.9: Minimalist PB-MCMC histograms

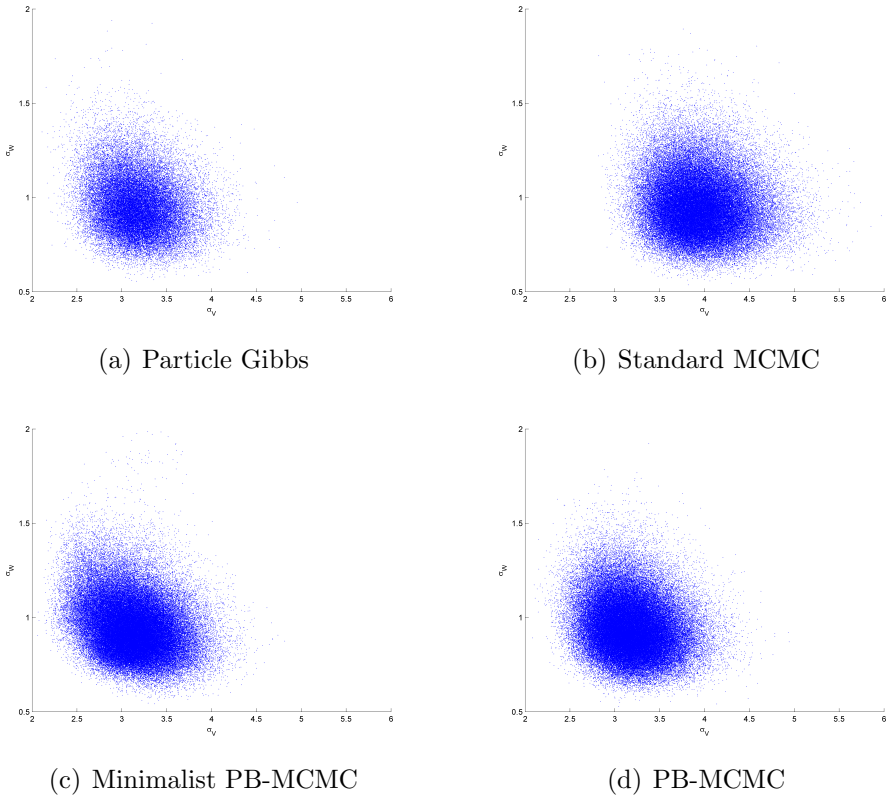


(a) Histogram/density estimate for σ_V



(b) Histogram/density estimate for σ_W

Figure 2.10: PB-MCMC histograms



(a) Particle Gibbs

(b) Standard MCMC

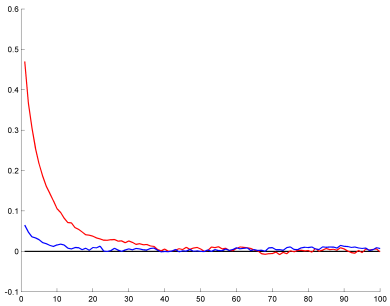
(c) Minimalist PB-MCMC

(d) PB-MCMC

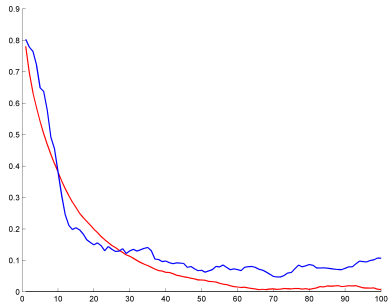
Figure 2.11: Scatter plots

timates for the other samplers. For the standard MCMC sampler, one can see from Figure 2.8 that the chain has still not converged after 500 million iterations. The reason for this is that moving one component at a time leads to the sampler getting stuck in a local mode of $x_{0:T}|\theta$ with low actual posterior density and subsequently the variance σ_V^2 is overestimated. For the minimalist PB-MCMC scheme, there are indications that the chain is converging after 2 million iterations, but there is still some discrepancy between the samples from the particle Gibbs sampler and those produced by this scheme. Upon further inspection of the samples produced, it can be seen that it takes on the order of thousands of iterations for a sample generated in subchain 0 to make it to subchain T while a sample from subchain T takes hundreds of thousands of iterations to be removed completely by the sampler. The PB-MCMC scheme which additionally updates one component selected at random in the chain performs much better, but it is worth noting that this additional update requires the evaluation of the transition density, which is unnecessary for the particle Gibbs algorithm and the minimalist PB-MCMC scheme. In addition, it converges considerably more slowly than the particle Gibbs sampler.

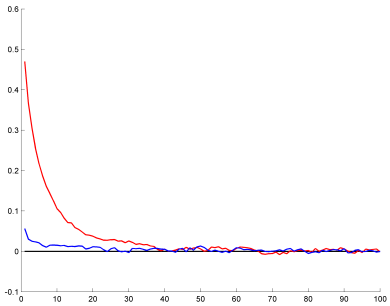
The autocorrelation plots in Figure 2.12 use lags in increments of 500 for the PB-MCMC methods to compensate for the fact that the particle Gibbs uses $N = 500$ particles and each simulation requires $O(NT)$ time, whereas the PB-MCMC methods require only $O(T)$ time. For the PB-MCMC with additional local moves, this comparison is somewhat unfair as the local moves require additional time to compute so despite the similar autocorrelation plot, the PB-MCMC method is more expensive by a factor of about 2. In addition, the PB-MCMC methods require the sampling of new values of θ for each chain, whereas only one sample of θ is required in a particle Gibbs implementation. In situations in which one cannot compute the sufficient statistics of θ given the latent variables and the data, the cost of updating θ can be even more expensive. Autocorrelation plots for the standard MCMC method are not presented because the MCMC kernel has clearly not converged.



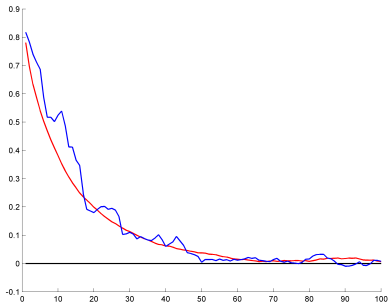
(a) Minimalist PB-MCMC method: σ_V



(b) Minimalist PB-MCMC method: σ_W



(c) PB-MCMC method: σ_V



(d) PB-MCMC method: σ_W

Figure 2.12: Autocorrelation plots for the PB-MCMC methods (blue) along with those of the particle Gibbs method (red). The x-axis denotes lags in increments of 500 for the PB-MCMC methods compared to lags of 1 for the particle Gibbs method.

2.6.2 Discussion

The purpose of this section has been to clarify the relationship between PB-MCMC and SMC methods, and show that they can generally be used to solve the same problems, albeit with different degrees of success. In population-based methods in general, open methodological questions surround the types of particle interactions that can be introduced to imbue samplers with good properties.

The PB-MCMC method still leaves a lot to be desired, despite performing reasonably well in some situations. The resampling step in SMC as a particle interaction step appears difficult to mimic in a PB-MCMC framework. On the other hand, some MCMC kernels with robust properties, such as the normal kernel coupler of Warnes (2001) and related methods can be chosen only periodically whereas a similar method involving marginal resampling (Klaas et al. 2005) can induce an $O(N^2)$ cost that is often too expensive in practice.

Chapter 3

Hierarchical Priors for Variable Selection

3.1 Introduction

There has been recent interest in sparse estimates for coefficients in regression and classification problems, with this problem often termed variable selection in the literature. To this end, a variety of approaches have been proposed in the statistics, machine learning and signal processing literature. Many of the computationally tractable approaches are the solutions of penalized optimization problems associated with regularization of the coefficients in likelihood optimization. Although not truly Bayesian approaches, often they can be interpreted as maximum *a posteriori* (MAP) estimates associated with the posterior density of the coefficients where the prior induces the regularization used in the optimization routine.

The problem is usually formulated as follows. One has observed data $\mathbf{y} = y_{1:n} \in \mathbb{R}^n$ and a design matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{R}^{n \times p}$, where one models each observation y_i as conditionally independent given $\mathbf{x}_i$ and some parameter $\boldsymbol{\beta}$. One defines $\boldsymbol{\beta} \in \mathbb{R}^p$ and a likelihood function $f(y_i|\mathbf{x}_i, \boldsymbol{\beta})$, noting that the likelihood of $\mathbf{y}$ given $(\mathbf{X}, \boldsymbol{\beta})$ is then $f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) = \prod_{i=1}^n f(y_i|\mathbf{x}_i, \boldsymbol{\beta})$. In order to perform inference on $\boldsymbol{\beta}$, one additionally defines a prior density $p(\cdot)$ such that the posterior density of $\boldsymbol{\beta}$ is given

by

$$p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}) = \frac{f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta})p(\boldsymbol{\beta})}{\int_{\mathbb{R}^p} f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta})p(\boldsymbol{\beta})d\boldsymbol{\beta}} \tag{3.1}$$

The parameter $\boldsymbol{\beta}$ is usually defined in such a way that when $\beta_j = 0$ for some $j \in \{1, \ldots, p\}$ then the value of $\mathbf{x}_{i,j}$ does not affect the conditional distribution of y_i for all $i \in \{1, \ldots, n\}$. For example, in a linear regression setting where $f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta})$ is a normal density with mean $\mathbf{X}\boldsymbol{\beta}$ and known variance $\delta^2 I$, then $\beta_j = 0$ implies that covariate j is irrelevant.

A popular family of penalized optimization approaches compute estimates as solutions to convex minimization problems involving the negative log-likelihood of the data given $\boldsymbol{\beta}$ and ℓ_q penalization on the coefficients with $0 \leq q \leq 1$. When q is in this range, the solutions are sparse for large enough values of the penalty parameters. When $q \geq 1$, the penalty function is additionally convex, a property that has made the choice of $q = 1$ particularly suitable when the negative log-likelihood is convex as this leads to a unique global minimum for the objective function.

Since Tibshirani (1996), it has become popular practice to use ℓ_1-regularization on each component of $\boldsymbol{\beta}$. However, use of identical penalization on each coefficient, e.g. $\lambda \sum_{j=1}^{p} |\beta_j|$ can lead to unacceptable bias in the resulting estimates (Fan and Li 2001), which has motivated use of sparsity-inducing non-convex penalties despite the increased difficulty in computing the resulting estimates. In particular, coefficients get shrunk towards zero even when there is overwhelming evidence in the likelihood that they are non-zero. This has motivated research into "adaptive" methods (Zou 2006; Zou and Li 2008) in the statistics literature and iteratively reweighted methods (Candès et al. 2008; Chartrand and Yin 2008) in the signal processing literature.

In this chapter, a hierarchical family of priors for $\boldsymbol{\beta}$ is proposed that amounts marginally to a sparsity-inducing, non-convex penalty in MAP estimation. Further, the specific hierarchy gives rise to an expectation-maximization (EM) algorithm (Dempster et al. 1977) that is essentially an iteratively reweighted ℓ_q-minimization algorithm. In one case, the algorithm corresponds to the iteratively reweighted ℓ_1-minimization algorithm and has been independently suggested in both Garrigues (2009) and Cevher (2009). Our hierarchical formulation of the prior, however, allows users to incorporate prior information about different coefficients and allows

flexibility in grouping variables together. For example, the framework gives immediately an adaptive version of the group LASSO algorithm proposed in Yuan and Lin (2006).

This family of priors is also suitable for exploratory Bayesian analysis on $\boldsymbol{\beta} \in \mathbb{R}^p$, where the hyperparameter controlling the scale of the prior is allowed to vary, akin to LASSO path plots. In this setting, the posterior for $\boldsymbol{\beta}$ becomes more sparse as this scale parameter decreases, giving data owners some measure of the relative importance of different predictors. While computationally demanding, this task is naturally amenable to SMC samplers methodology implemented on a many-core architecture.

It is worth noting that the methodology of this chapter is not proposed as a solution to the problem of Bayesian variable selection, introduced briefly in Section 1.1.2.3. In particular, one can see that there is no definition of a likelihood function and prior density for each subset $\gamma \in \{0,1\}^p$ of possible covariates, nor even a prior on each model defined by γ. As such, the quasi-Bayesian MAP estimation approach and the exploratory Bayesian analysis of the full model are not attempting to solve this problem, but instead to provide solutions to less computationally demanding problems.

3.1.1 Penalized Optimization & Quasi-Bayesian Methods

Given a prior distribution for $\boldsymbol{\beta}$, we are interested in inference on $\boldsymbol{\beta}$ conditional on data $(\mathbf{y}, \mathbf{X})$ in a general regression setting. While a Bayesian approach would usually suggest sampling from the posterior density (3.1), we focus here on MAP (point) estimates of $\boldsymbol{\beta}$ since these are computationally cheaper, especially when $f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta})$ and $p(\boldsymbol{\beta})$ are concave. MAP estimates are computed by solving the optimization problem

$$\hat{\boldsymbol{\beta}}_{MAP} = \arg\max_{\boldsymbol{\beta}} f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) p(\boldsymbol{\beta})$$

or, equivalently

$$\hat{\boldsymbol{\beta}}_{MAP} = \arg\max_{\boldsymbol{\beta}} \log f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) + \log p(\boldsymbol{\beta}) \tag{3.2}$$

The $\log p(\boldsymbol{\beta})$ term can be thought of as a penalization term in the optimization of the log-likelihood of the data $\log f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta})$. The use of a MAP estimate as a quasi-Bayesian solution to the problem of variable selection is motivated by the observation that for certain types of priors, and depending on the values of their hyperparameters, this estimate will have some components of $\boldsymbol{\beta}$ set exactly to 0. As such, those non-zero components define a model that can be viewed as a point estimate of the model that generated the data. There are, of course, serious theoretical issues surrounding the use of such an estimate, some of which are discussed in Section 3.2.4.

3.2 A Hierarchical Prior

We propose a hierarchical approach to constructing priors for $\boldsymbol{\beta}$. At the lowest level, we give each element β_j of $\boldsymbol{\beta}$ an independent normal prior with mean 0 and variance σ_j^2, i.e. $p(\boldsymbol{\beta}|\sigma_{1:p}^2) = \prod_{j=1}^{p} p(\beta_j|\sigma_j^2)$ where $\beta_j|\sigma_j^2 \sim N(0, \sigma_j^2)$. If we leave σ_j^2 for $j \in \{1, \ldots, p\}$ as hyperparameters, computing the resulting MAP estimate corresponds to ℓ_2-penalized optimization of the log-likelihood.

If instead we model each σ_j^2 as being drawn from an exponential distribution with mean $2\tau_j^2$ we obtain a double-exponential distribution for β_j after σ_j^2 has been integrated out, i.e.

$$p(\beta_j|\tau_j) = \frac{1}{2\tau_j} \exp(-\frac{|\beta_j|}{\tau_j})$$

Computing the MAP estimate associated with this prior corresponds to ℓ_1-penalized optimization and the solution itself is identical to the LASSO estimate when f is a multivariate Gaussian with mean $\mathbf{X}\boldsymbol{\beta}$. This prior has become popular in recent years for variable selection since it induces sparsity in $\hat{\boldsymbol{\beta}}_{MAP}$ for small enough values of τ_j (Tibshirani 2011).

3.2.1 The Hierarchical Adaptive LASSO Prior

We propose adding another level of hierarchy to the prior by having separate random variables τ_j for each $j \in \{1, \ldots, p\}$ and placing an inverse-gamma prior on each τ_j. Indeed, if we let $\tau_j \sim IG(a_j, b_j)$ we obtain

$$p(\beta_j | a_j, b_j) = \frac{a_j}{2b_j} \left(\frac{|\beta_j|}{b_j} + 1 \right)^{-(a_j+1)} \tag{3.3}$$

after integrating out τ_j, which we call the hierarchical adaptive LASSO (HAL) prior since one can compute MAP estimates using this prior with a type of adaptive LASSO algorithm, as will be seen in Section 3.2.1.1. This is a density from the family of generalized t-distribution priors (McDonald and Newey 1988). Computing the MAP estimate associated with this prior corresponds to logarithmic penalization of the log-likelihood. From a Bayesian modelling perspective, the introduction of a distribution over the τ_j is a natural way to resolve the issue of believing *a priori* that there are differences in the sizes of the components of $\boldsymbol{\beta}$ such that the β_j's cannot be modelled as being *a priori* i.i.d. samples from a single Laplace distribution.

3.2.1.1 MAP Estimation via Expectation-Maximization

The optimization problem associated with the HAL prior is not concave. However, one can find local modes of the posterior using the EM algorithm with the $\boldsymbol{\tau} = \tau_{1:p}$ as latent variables. Indeed, each iteration of EM takes the form

$$\boldsymbol{\beta}^{(t+1)} = \arg\max_{\boldsymbol{\beta}} \log f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) + \int \log[p(\boldsymbol{\beta}|\boldsymbol{\tau})] p(\boldsymbol{\tau}|\boldsymbol{\beta}^{(t)}, a, b) d\boldsymbol{\tau}.$$

The conjugacy of the inverse-gamma distribution with respect to the Laplace distribution gives

$$\tau_j | \beta_j^{(t)}, a_j, b_j \sim IG(a_j + 1, b_j + |\beta_j|)$$

and with $p(\boldsymbol{\beta}|\boldsymbol{\tau}) = \prod_{j=1}^{p} p(\beta_j|\tau_j) = \prod_{j=1}^{p} 1/(2\tau_j) \exp(-|\beta_j|/\tau_j)$ yields

$$\boldsymbol{\beta}^{(t+1)} = \arg\max_{\boldsymbol{\beta}} \log f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) - \sum_{j=1}^{p} |\beta_j| \int \frac{1}{\tau_j} p(\tau_j | \beta_j^{(t)}, a_j, b_j) d\tau_j$$

where the expectation of $1/\tau_j$ given $\tau_j \sim IG(a_j+1, b_j+|\beta_j^{(t)}|)$ is $(a_j+1)/(b_j+|\beta_j^{(t)}|)$.

As such, one can find a local mode of the posterior $p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \boldsymbol{\beta})$ by starting at some point $\boldsymbol{\beta}^{(0)}$ and then iteratively solving

$$\boldsymbol{\beta}^{(t+1)} = \arg\max_{\boldsymbol{\beta}} \log f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) - \sum_{j=1}^{p} w_j^{(t)} |\beta_j| \tag{3.4}$$

where

$$w_j^{(t)} = \frac{a_j + 1}{b_j + |\beta_j^{(t)}|}.$$

It is clear that for large enough values of a_j and small enough values of b_j that the MAP estimates obtained by the EM algorithm are sparse. In fact, any posterior mode with this prior corresponds to a weighted LASSO solution, which is sparse when the penalization defined by $\{(a_j, b_j)\}_{j=1}^p$ is large enough. In the rest of this chapter, this algorithm will be referred to as the hierarchical adaptive LASSO (HAL).

3.2.1.2 Oracle Properties

In the penalized optimization literature, some estimators are justified at least partially by their possession of the oracle property: that for appropriate parameter choices, the method performs just as well as an oracle procedure in terms of selecting the correct predictors and estimating the nonzero coefficients correctly asymptotically in n when the likelihood is Gaussian. Other related properties include asymptotic unbiasedness, sparsity and continuity in the data (Fan and Li 2001). Penalization schemes with the oracle property include the smoothly clipped absolute deviation method (Fan and Li 2001), the adaptive LASSO (Zou 2006) and the one-step local linear approximation method (Zou and Li 2008).

For the HAL prior, the MAP estimate obtained using this prior has also been examined in (Cevher 2009; Armagan et al. 2011), all of which derive the expectation-maximisation algorithm for finding a local mode of the posterior. If $(a_j, b_j) = (a, b)$

for all $j \in \{1, \ldots, p\}$, it is straightforward to establish that the estimate is asymptotically unbiased, sparse when $b < 2\sqrt{(a+1)}$ and continuous in the data when $b = \sqrt{(a+1)}$ following the conditions laid out in Fan and Li (2001). Armagan et al. (2011) shows that this scheme satisfies the oracle property if simultaneously $a \to \infty$, $a/\sqrt{n} \to 0$ and $b\sqrt{n} \to C < \infty$ for some constant C.

It is worth remarking that the oracle property requires the prior on $\boldsymbol{\beta}$ to depend on the number of observations n, to manage the tradeoff between sparsity and consistency. This is in conflict with conventional Bayesian analysis, where priors represent beliefs about the data generating mechanism irrespective of the number of observations that will be obtained. In addition, given an actual application the values of n and p may not be in the realm of asymptotic relevance for this property to hold.

3.2.2 Extensions

The HAL prior is only one possible choice that can be obtained using a general hierarchical approach. We consider here extensions to obtain the family of generalized t-distribution priors in McDonald and Newey (1988) as well as priors that have a grouping effect through the sharing of parameters in the hierarchy.

3.2.2.1 Generalized t-Distribution Priors

One can model β_j more generally as coming from an exponential power distribution instead of a Laplace distribution. In this case, we can write

$$p(\beta_j|\eta_j, q) = \frac{1}{2\eta_j^{1/q}\Gamma(1+1/q)} \exp\left(-\frac{|\beta_j|^q}{\eta_j}\right).$$

With an inverse-gamma prior on η_j, which enjoys conjugacy with respect to the exponential power distribution, we obtain

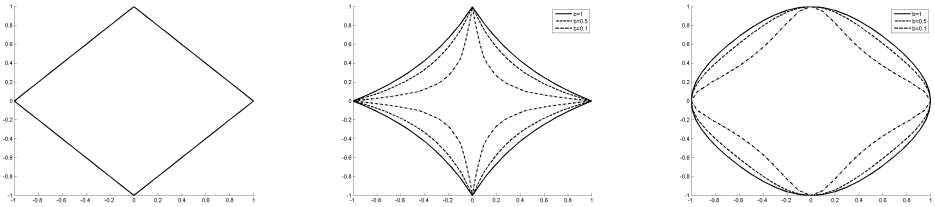
$$p(\beta_j|a_j, b_j, q) = \frac{q}{2\mathrm{B}(a_j, 1/q)b_j^{1/q}} \left(\frac{|\beta_j|^q}{b_j} + 1\right)^{-(a_j+1/q)},$$

which is equivalent to the generalized t-distribution of McDonald and Newey (1988) if one substitutes b_j with $a_j c_j^q$. Use of this prior results in the same scheme given by (3.4) but where the weights are given by

$$w_j^{(t)} = \frac{a_j + 1/q}{b_j + |\beta_j^{(t)}|^q}.$$

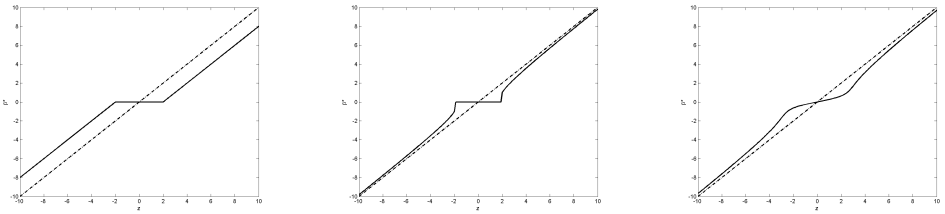
The use of an exponential power prior can be motivated hierarchically as a scale mixture of normal distributions for $q \in [1, 2)$ (West 1987) or as a scale mixture of uniform distributions for $q \in (1, \infty)$ (Walker and Gutiérrez-Peña 1999). For $q \in (0, 1)$ this distribution is still defined but it does not have the same interpretation as when $q \geq 1$ and additionally has a non-concave density which complicates computation of posterior modes. The choice $q = 2$ corresponds to a normal distribution and one can obtain a student t-distribution with ν degrees of freedom after marginalizing out η by choosing $a_j = \nu/2$ and $b_j = \nu$. This choice leads to a hierarchical adaptive ℓ_2-regularized method that may be suitable for problems in which prediction instead of variable selection is important.

Contour plots of the negative log density of the joint prior for two variables are given in Figure 3.1 and thresholding plots associated with the priors are given in Figure 3.2. The contour plots show graphically how the LASSO and HAL approaches give sparse solutions whilst the hierarchical adaptive ridge (HAR) prior, corresponding to $q = 2$, gives non-sparse solutions. The thresholding plots show that whilst the LASSO significantly biases even large coefficients, the HAL and HAR do not.



(a) LASSO (b) HAL (c) HAR

Figure 3.1: Two-dimensional contour plots of the penalties, i.e. the negative log density, associated with the priors.



(a) LASSO (b) HAL (c) HAR

Figure 3.2: Threshold plots associated with the priors.

3.2.2.2 The Hierarchical Adaptive Group LASSO

The hierarchical framework allows us to group variables together by making them dependent on a shared variable higher up in the hierarchy. For example, letting $g : \{1, \ldots, p\} \to \{1, \ldots, K\}$ be a function mapping variables to one of K groups and n_i be the number of variables in group i we can use the following model:

$$\begin{aligned}
\beta_j|\sigma^2_{g(j)} &\sim N(0, \sigma^2_{g(j)}), \text{ for } j \in \{1, \ldots, p\} \\
\sigma^2_i|\tau_i &\sim G\left(\frac{n_i+1}{2}, 2\tau_i^2\right), \text{ for } i \in \{1, \ldots, K\} \\
\tau_i|a_i, b_i &\sim IG(a_i, b_i), \text{ for } i \in \{1, \ldots, K\}
\end{aligned}$$

With $G_i = \{j : g(j) = i\}$, this gives

$$p(\beta_{G_i}|\tau_i) = \frac{(2\tau_i)^{-n_i}\pi^{-(n_i-1)/2}}{\Gamma((n_i+1)/2)} \exp(-\frac{\sqrt{\sum_{j\in G_i}|\beta_j|^2}}{\tau_i})$$

and so $\tau_i|\beta_{G_i}, a_i, b_i \sim IG(a_i + n_i, b_i + \sqrt{\sum_{j\in G_i}|\beta_j|^2})$. The corresponding iterative procedure is then

$$\boldsymbol{\beta}^{(t+1)} = \arg\max_{\boldsymbol{\beta}} \log f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) - \sum_{i=1}^{K} w_i^{(t+1)}||\beta_{G_i}||_2$$

where

$$w_i^{(t+1)} = \frac{a_i + n_i}{||\beta^{(t)}_{G_j}||_2 + b_i}.$$

The marginal prior on β_{G_i} has the density

$$p(\beta_{G_i}|a_i, b_i) = \frac{(2b_i)^{-n_i}\pi^{-(n_i-1)/2}\Gamma(n_i + a_i)}{\Gamma((n_i+1)/2)\Gamma(a_i)} \left(\frac{||\beta_{G_i}^{(t)}||_2}{b_i} + 1 \right)^{(-a_i-n_i)}$$

but this density is never evaluated in the EM algorithm.

A related problem to grouped variable selection is known as multi-task learning within the machine learning literature, where one wants to solve for $\boldsymbol{\theta} = \{\boldsymbol{\beta}^{(i)}\}_{i=1}^{L}$ in a variety of L related regression models. One approach is to solve the optimization problem

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{L} \log f_i(\mathbf{y}_i|\mathbf{X}_i, \boldsymbol{\beta}^{(i)}) + \sum_{j=1}^{p} \lambda_j ||\boldsymbol{\beta}_j||_2$$

where $\boldsymbol{\beta}_j = (\beta_j^{(1)}, \ldots, \beta_j^{(L)}) \in \mathbb{R}^L$ (Obozinski et al. 2006). This type of regularization can be derived using the same hierarchical prior used in the group LASSO where the coefficients relating to the same covariate are 'grouped' together to promote sparsity across the individual $\boldsymbol{\beta}$ estimates, i.e. a covariate is selected in all the related models or in none of the models. As such, an adaptive version of this multi-task learning approach follows the same form as the hierarchical adaptive group LASSO.

3.2.2.3 Matrix Priors

For the purpose of covariance matrix estimation, ℓ_1-regularization has been used on entries of the precision matrix Ω of a Gaussian graphical model (Yuan and Lin 2007; Friedman et al. 2008). This corresponds to MAP estimation using Laplace priors on each Ω_{ij} for $i \leq j$. We can incorporate this type of prior within our framework by placing inverse Gamma priors on the scale parameters of each Laplace distribution. We have

$$p(\Omega_{ij}|\tau_{ij}) = \frac{1}{2\tau_{ij}} \exp(-\frac{|\Omega_{ij}|}{\tau_{ij}})$$

with $p(\Omega|\tau) = \prod_{i=1}^{p} \prod_{j=i}^{p} p(\Omega_{ij}|\tau_{ij})$ and $\tau_{ij} \sim IG(a_{ij}, b_{ij})$. Note that in this formulation the prior on Ω is non-zero for non-positive-definite values. This allows us to specify

$$p(\boldsymbol{\tau}|\Omega, A, B) = \prod_{i=1}^{p} \prod_{j=i}^{p} IG(\tau_{ij}; a_{ij} + 1, b_{ij} + |\Omega_{ij}|).$$

One can define the likelihood of the observed data Y, $p(Y|\Omega)$ to be zero if Ω is not symmetric positive-definite. In this case, the posterior and hence the MAP estimate are equivalent to the case where the prior takes the form

$$p(\Omega|A,B) = \frac{\mathbf{1}_{\mathcal{P}}(\Omega)p(\Omega|A,B)}{\int \mathbf{1}_{\mathcal{P}}(\Omega)p(\Omega|A,B)d\Omega}$$

since in both cases we have

$$p(\Omega|Y,A,B) = \frac{\mathbf{1}_{\mathcal{P}}(\Omega)p(Y|\Omega)p(\Omega|A,B)}{\int \mathbf{1}_{\mathcal{P}}(\Omega)p(Y|\Omega)p(\Omega|A,B)d\Omega}$$

where $\mathcal{P}$ is the set of symmetric positive-definite matrices. Note, however, that the positive-definite prior cannot be used to derive the EM algorithm central to our methodology since $p(\boldsymbol{\tau}|\Omega,A,B)$ is no longer a product of inverse-gamma densities.

3.2.2.4 The Hierarchical LASSO

In some cases, one might be interested in having $\eta_j = \eta$ for all $j \in \{1,\ldots,p\}$ with $\eta \sim IG(a,b)$. In this case, one obtains a prior on $\boldsymbol{\beta}$ of the form

$$p(\boldsymbol{\beta}|a,b,q) = \frac{\Gamma(a+p/q)}{2^p\Gamma(a)\Gamma(1+1/q)^p b^{p/q}}\left(\frac{\sum_{j=1}^p |\beta_j|^q}{b}+1\right)^{-a-p/q} \tag{3.5}$$

which leads to the iterative procedure

$$\boldsymbol{\beta}^{(t+1)} = \arg\max_{\boldsymbol{\beta}} \log f(\mathbf{y}|\mathbf{X},\boldsymbol{\beta}) - w^{(t+1)}\sum_{j=1}^p |\beta_j|^q$$

where

$$w^{(t+1)} = \frac{a+p/q}{b+\sum_{j=1}^p |\beta_j^{(t)}|^q}.$$

In fact, a more general prior can be constructed by considering groupings of the coefficients such that $\beta_j \sim EP(\eta_i,q)$ for all $j \in G_i$, where G_i is again the set of indices of coefficients in group i. A prior constructed in this fashion leads to the

iterative procedure

$$\boldsymbol{\beta}^{(t+1)} = \arg\max_{\boldsymbol{\beta}} \log f(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) - \sum_{i=1}^{K} w_i^{(t+1)} \sum_{j \in G_i} |\beta_j|^q$$

where

$$w_i^{(t+1)} = \frac{a_i + n_i/q}{b_i + \sum_{j \in G_i} |\beta_j^{(t)}|^q}.$$

3.2.2.5 Modifying The Hierarchy

The above examples are only a subset of the possible modifications to the hierarchy that are possible. Indeed, one of the benefits of a hierarchical approach is that one can flexibly group variables via the sharing of random variables. Graphical models for the exponential family generalization and the grouped variable generalization are given in Figure 3.3 along with a discussion of their relationships to existing methods in Section 3.3.7.

3.2.3 Tuning the Hyperparameters

Use of the proposed framework relies on appropriate settings of the hyperparameters. For distributions of non-negative Z with density

$$p(Z|\nu, b) = \frac{\nu - 1}{b} \left(\frac{Z}{b} + 1 \right)^{-\nu}$$

the moments of Z are given by

$$\mathbb{E}_p[Z^t] = \frac{b^t \Gamma(\nu - 1 - t)\Gamma(t + 1)}{\Gamma(\nu - 1)},$$

which allows one to pick hyperparameters that represent prior beliefs about the mean and variance of variables of interest, e.g. $|\beta_j|$ in the case of prior (3.3) or $(\sum_{j=1}^{p} |\beta_j|^q)^{1/q}$ in the case of prior (3.5). Focusing on the HAL prior, we note that in this case we have $\mathbb{E}[|\beta_j|] = b_j/(a_j - 1)$, for $a_j > 1$. An observation on a_j and b_j is that when one increases both values but keeps $\mathbb{E}[|\beta_j|]$ constant, the tendency

for the iterative scheme to set β_j to zero is reduced since w_j is upper-bounded by $(a_j + 1)/b_j$. This observation could be used in a 'tempered' optimization scheme as discussed in Section 3.2.4, noting in particular that as $a_j \to \infty$, the prior approaches a Laplace distribution and so the posterior approaches unimodality.

3.2.4 Issues with MAP Estimation

There are many criticisms of MAP estimates from a Bayesian perspective. The motivation for their use is primarily computational, since Bayesian variable selection methods tend to be prohibitively expensive when dealing with large data sets. Beyond the obvious problem of summarizing the posterior distribution over models with a point estimate, one problem is that MAP estimates are not Bayes estimators but instead a limit of Bayes estimators under the 0-1 loss function. While important, this issue is not addressed here. A perhaps more fundamental issue is that MAP estimates are not invariant under reparametrization. This issue can be rectified by finding the point that maximizes posterior density with the Jeffreys measure as the dominating measure (Jermyn 2005; Druilhet and Marin 2007), e.g. for a likelihood $f(x|\theta)$ and prior $p(\theta)$, the parametrization-invariant MAP is given by

$$\theta_{MAP} = \arg\max_{\theta} f(x|\theta)p(\theta)|I(\theta)|^{-1/2}$$

where $I(\theta)$ is the Fisher information associated with $f(x|\theta)$.

An important issue with the MAP estimates obtained from our methodology is that the posterior is multimodal and there is no guarantee that one will obtain the global mode of the posterior as opposed to a local one. However, this is true of almost all non-convex penalized optimization approaches. In Chartrand and Yin (2008), a suggestion is to start with high values of b_j and reduce the values of b_j once the algorithm has converged. In principle, this can be done with both a_j and b_j, noting that such an algorithm will still find a local mode of the posterior and this "tempering" of the posterior during optimization can affect which mode is chosen. We do not investigate this further but note that characterization of the modes obtained using such a process is an interesting open question. It is also worth noting that one-step

estimates can satisfy the oracle property regardless of whether or not the global mode has been found (Zou and Li 2008), so the inability to guarantee the recovery of the global mode is not necessarily problematic from an estimation standpoint, although this does not aid interpretation or provide theoretical justification from a Bayesian perspective.

A final remark is that one should be wary of characterizing the model associated with even the global mode of the posterior for $\boldsymbol{\beta}$ using a sparsity-inducing prior. In particular, this model is not necessarily the same as the MAP estimate of the model. Indeed, one has not defined the likelihood functions and priors for each subset of the covariates, or even a prior on each model, and so the posterior over models is undefined.

3.3 Related Approaches

The proposed approach, either in the hierarchical model or in the estimation step, is closely related to many approaches that have been suggested in the literature. One contribution of this work is therefore to provide a Bayesian interpretation of existing methods and a flexible framework in which one can incorporate different models.

3.3.1 Laplacian Scale Mixtures & Compressible Priors

The HAL prior has been proposed independently in both Garrigues (2009) and Cevher (2009). In the former, one obtains the same procedure from a majorization-minimization algorithm and in the latter from an EM algorithm. However, the framework presented in this chapter makes explicit the flexibility of the hierarchical modeling approach and generalizes this prior to exponential power families, situations with grouped variables and positive-definite matrices, and provides suggestions for choosing hyperparameters.

3.3.2 Weakly Informative Priors

With $\beta_j \sim N(0, \sigma_j^2)$ and $\sigma_j^2 \sim IG(a_j, b_j)$ one obtains marginally a t-distribution for $\beta_j | a_j, b_j$. This corresponds to the idea of using weakly informative priors as in Gelman et al. (2008). For the case where $\beta_j \sim \text{Laplace}(0, \tau)$ and $\tau \sim IG(a_j, b_j)$, i.e. the HAL prior, we can similarly think of the generalized t-distribution prior on β_j after marginalizing out τ as a weakly informative prior. In fact, one can think of all of the priors proposed using the hierarchical approach in this work as weakly informative.

3.3.3 Adaptive Methods

Within the statistical literature, the closest approach is perhaps the adaptive LASSO (Zou 2006), whose implementation corresponds to a single step of the exponential power family generalization of the HAL with $\boldsymbol{\beta}^{(0)}$ a root-n consistent estimator of β and $b_j \to 0$. As such, the adaptive LASSO estimator can be thought of as taking an initial estimate and returning an estimate with higher posterior density given a logarithmic prior. Our method, on the other hand, finds a local mode of the posterior.

Similarly, the benefit of a polynomial form for the prior density is related to the motivation for the smoothly clipped absolute deviation penalty (Fan and Li 2001). Indeed, the penalization induced by the HAL prior grows slowly so that large values of β_j are not unnecessarily biased while remaining sparse. The one-step estimation scheme of Zou and Li (2008) is also closely related, being iteratively reweighted optimization algorithms with a different penalization.

In fact, one can show that at each iteration of the EM algorithm using the HAL prior, $\boldsymbol{\beta}^{(t)}$ is a one-step sparse estimate, but with a different penalty to those considered in Zou and Li (2008). The one-step penalty function associated with this scheme is $f(|\beta_j|) = -\log p(|\beta_j||a_j, b_j)$, with the property that

$$f'(|\beta_j|) = \frac{a_j + 1}{|\beta_j| + 1},$$

which is exactly the form of the weights used in the EM algorithm.

3.3.4 Iteratively Reweighted ℓ_q-Minimization

The basic HAL algorithm is clearly similar to the iteratively reweighted-ℓ_1 approach proposed in Candès et al. (2008), which is identical except that the weights have the form

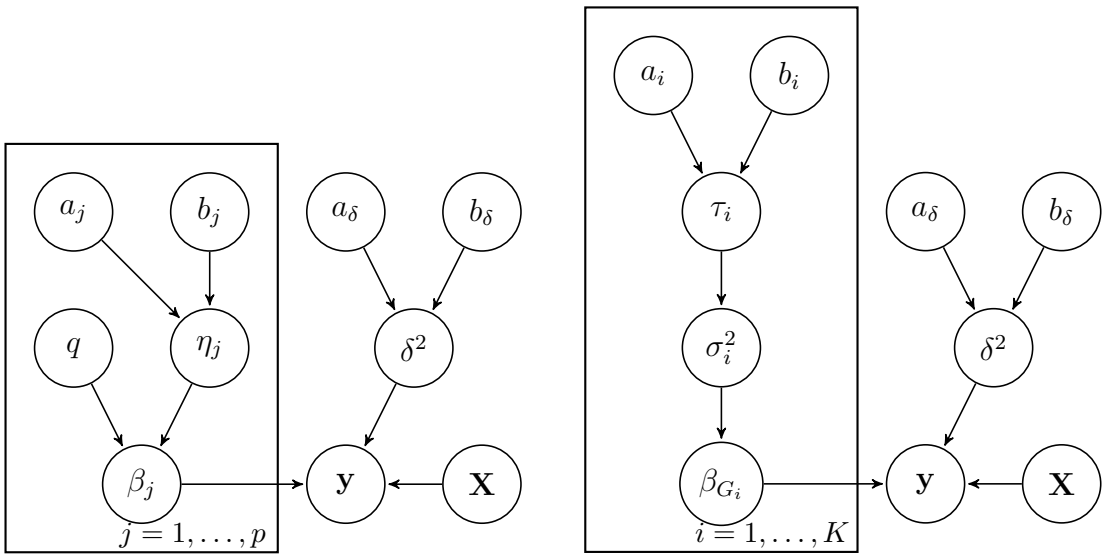
$$w_j^{(t)} = \frac{\lambda}{\epsilon + |\beta_j^{(t)}|}$$

which corresponds to the case where $a_j = \lambda - 1$ and $b_j = \epsilon$.

Similarly, the exponential-family generalization of the HAL algorithm is related to the family of approaches suggested in Chartrand and Yin (2008) for the various ℓ_q-penalization norms. As such, the hierarchical model for $\boldsymbol{\beta}$ gives an interpretation to the methods in the family of iteratively reweighted optimization solutions and, in particular, to the selection of additional parameters ϵ and λ.

3.3.5 Normal-Exponential-Gamma Priors

Our hierarchical prior differs from that suggested in Griffin and Brown (2007), where in that work an inverse gamma prior is placed on τ_j^2 as opposed to τ_j. This difference results in a posterior for $\boldsymbol{\beta}$ for which it is difficult to obtain MAP estimates (Griffin and Brown 2010b), although this problem can be alleviated somewhat by novel fast methods for computation of the parabolic cylinder function (Murphy 2012). The marginal prior is a member of the generalized hyperbolic family. This difference also appears in Park and Casella (2008), although in that work the full posterior is explored using MCMC and the parameter τ is shared amongst all coefficients.



(a) EP Hierarchy (b) Grouped Hierarchy

Figure 3.3: Graphical model for the exponential power and grouped variable hierarchies.

3.3.6 A Note on Improper Priors

Consider the exponential power density

$$p(\beta_j|\eta_j, q) = \frac{1}{2\eta_j^{1/q}\Gamma(1+1/q)} \exp\left(-\frac{|\beta_j|^q}{\eta_j}\right)$$

with the scale-invariant prior on η_j, $p(\eta_j) \propto 1/\eta_j$. The prior on β_j after marginalizing out η_j is then, regardless of q, improper with the form $p(\beta_j|q) \propto 1/|\beta_j|$. Since this is the same prior for $q = 1$, which we know will produce sparse $\boldsymbol{\beta}$ and for $q = 2$, which is the prior proposed in Figueiredo (2003), this explains why the prior in Figueiredo (2003) produces sparse results. However, it is worth noting that the posterior for $\boldsymbol{\beta}$ using this prior is improper with unbounded density at $\boldsymbol{\beta} = \mathbf{0}$.

3.3.7 Graphical Model

Figure 3.3 gives graphical models for the hierarchies corresponding to the exponential power (EP) generalization of section 3.2.2.1 and the adaptive group LASSO of section 3.2.2.2. These models allow us to visualize the flexibility of the framework

and the connections with related approaches. Indeed, for $q = 1$ one obtains the HAL prior or, by setting η_j to be a fixed hyperparameter, the LASSO prior. Similarly, for $q = 2$ one obtains hierarchical adaptive ridge regression or ridge regression, respectively. For the hierarchy with grouped variables, the similarity to the HAL hierarchy is clear, suggesting that application-specific hierarchies could be developed that lead to iteratively reweighted methods.

3.4 MAP Estimation Examples

3.4.1 Linear Regression

In linear regression, the likelihood of $\mathbf{y}$ given X and $\boldsymbol{\beta}$ is given by

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}, \mu, \delta^2) = \frac{1}{(2\pi\delta^2)^{n/2}} \exp\left\{-\frac{1}{2\delta^2}(\tilde{\mathbf{y}}_\mu - \mathbf{X}\boldsymbol{\beta})^T(\tilde{\mathbf{y}}_\mu - \mathbf{X}\boldsymbol{\beta})\right\}$$

where $\tilde{\mathbf{y}}_\mu = \mathbf{y} - \mu\mathbf{1}_n$. If X is standardized, we have $1_n^T X = 0$ and so we can put an improper prior on μ with $p(\mu) \propto 1$ and integrate it out so that

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}, \delta^2) = \frac{1}{(2\pi\delta^2)^{\frac{n-1}{2}}\sqrt{n}} \exp\left\{-\frac{1}{2\delta^2}(\tilde{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta})^T(\tilde{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta})\right\} \tag{3.6}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^n y_i$ and $\tilde{\mathbf{y}} = \mathbf{y} - \bar{y}1_n$.

3.4.1.1 Known Noise

If δ^2 is fixed, we proceed as expected. Note that in this case, the Jeffreys prior for $\boldsymbol{\beta}$ is a uniform improper prior so no adjustment needs to be made to make the MAP estimate parametrization invariant.

To test the method, we simulated data using $\boldsymbol{\beta} = (3, 1.5, 0, 0, 2, 0, 0, 0)^T$, $\delta^2 = 1$ and $X \sim N(0, \Sigma)$ with $\Sigma_{i,j} = 0.5^{|i-j|}$. We then ran 1000 repetitions of the HAL and the LASSO on this problem with various settings of (a, b) and τ respectively with the

results given in Tables 3.1-3.2.

Table 3.1: Results for the LASSO (linear regression, $\delta = 1$)

n	τ	avg. error	% correct	avg. false positives	avg. false negatives
40	0.2	0.4137	9.7	1.808	0.0
40	0.1	0.4817	36.7	0.89	0.0
40	0.02	1.6732	90.0	0.089	0.015
80	0.2	0.2872	2.8	2.519	0.0
80	0.1	0.2931	20.0	1.3510	0.0
80	0.02	0.8169	92.7	0.079	0.0

Table 3.2: Results for the HAL (linear regression, $\delta = 1$)

n	(a, b)	avg. error	% correct	avg. false positives	avg. false negatives
40	$(1, 0.1)$	0.3118	89.9	0.105	0.0
40	$(2, 0.1)$	0.3044	98.1	0.019	0.0
40	$(2, 0.05)$	0.3026	99.6	0.004	0.0
80	$(1, 0.1)$	0.2191	81.3	0.079	0.0
80	$(2, 0.1)$	0.2061	96.6	0.034	0.0
80	$(2, 0.05)$	0.2038	98.8	0.012	0.0

Both methods are capable of giving good results in this setting, which has a high signal-to-noise ratio. However, the reduction in average error is evident for the HAL, owing mainly to less penalization of the selected coefficients. We ran the same experiment but with $\delta = 3$, to test the algorithm with a lower signal-to-noise ratio with the results given in Tables 3.3-3.4. Again, the results for the HAL are typically superior to that for the LASSO. However, incorporating prior information leading to less penalization of β_2 and β_5 improves performance drastically. This type of prior information is likely to be necessary when we wish to include variables whose true coefficients are small.

Table 3.3: Results for the LASSO (linear regression, $\delta = 3$)

n	τ	avg. error	% correct	avg. false positives	avg. false negatives
40	1/6	1.9747	53.5	0.378	0.255
40	0.125	2.3952	43.0	0.207	0.580
40	0.125*	2.2117	93.9	0.005	0.057

* $(\tau_2, \tau_5) = (0.25, 0.25)$

Table 3.4: Results for the HAL (linear regression, $\delta = 3$)

n	(a, b)	avg. error	% correct	avg. false positives	avg. false negatives
40	$(2, 0.75)$	1.3407	56.2	0.274	0.352
40	$(2, 0.1)$	1.7198	28.0	0.064	0.831
40	$(2, 0.1)^*$	1.0224	95.9	0.004	0.038

* $(a_2, b_2, a_5, b_5) = (2, 2, 2, 2)$

3.4.1.2 Unknown Noise

If we model $\delta^2 \sim IG(a_\delta, b_\delta)$, we can find that MAP estimate associated with the posterior density $p(\boldsymbol{\beta}|\mathbf{y}, X)$, i.e. with δ^2 integrated out. To do so, we additionally include δ^2 as a latent variable in the EM algorithm, noting that conditional on $\boldsymbol{\beta}$, δ^2 and $\boldsymbol{\tau}$ are independent. Furthermore, we have $\delta^2|\boldsymbol{\beta}, X, \mathbf{y} \sim IG(a_\delta + (n-1)/2, b_\delta + 1/2(\tilde{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta})^T(\tilde{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta}))$ For the HAL, we iteratively solve

$$\boldsymbol{\beta}^{(t+1)} = \arg\max_{\boldsymbol{\beta}} -v_j^{(t)} \frac{1}{2}(\tilde{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta})^T(\tilde{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta}) - \sum_{j=1}^{p} w_j^{(t)}|\beta_j|$$

where

$$v_j^{(t)} = \frac{a_\delta + (n-1)/2}{b_\delta + 1/2(\tilde{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta}^{(t)})^T(\tilde{\mathbf{y}} - \mathbf{X}\boldsymbol{\beta}^{(t)})} \text{ and } w_j^{(t)} = \frac{a_j + 1}{b_j + |\beta_j^{(t)}|}$$

To test the method, we simulated data using the same as before but letting $\delta^2 \sim IG(a_\delta, b_\delta)$. We then ran 1000 repetitions of the HAL with various settings of $(a_\delta, b_\delta, a, b)$ with the results given in Table 3.5. There is clearly more difficulty in estimating the coefficients accurately when the variance of the observations is higher and there is again increased performance with good prior information.

Table 3.5: Results for the HAL (linear regression, random δ^2)

n	(a_δ, b_δ)	(a, b)	avg. error	% correct	avg. false pos.	avg. false neg.
40	(3,5)	$(2, 0.1)$	0.5509	90.5	0.040	0.070
40	(1,1)	$(2, 0.1)$	0.7302	79.8	0.058	0.265
40	(1,4)	$(2, 0.2)$	1.5046	53.0	0.085	0.742
40	(1,4)	$(2, 0.2)^*$	1.1865	78.0	0.047	0.318

* $(a_2, b_2, a_5, b_5) = (2, 2, 2, 2)$

3.4.1.3 Grouped Variable Selection

For grouped variable selection, we use $p = 32$ with groups of size 4. We let $\beta_{1:4} = (3, 1.5, 2, 0.5)'$, $\beta_{9:12} = (6, 3, 4, 1)'$, $\beta_{17:20} = (1.5, 0.75, 1, 0.25)'$ with all other components set to 0. The groupings of variables were given by $G_i = \{4i + k : k \in \{1, 2, 3, 4\}\}$. As with ungrouped variable selection, the hierarchical adaptive version of the group LASSO gives lower average errors and has a higher percentage of correct models chosen compared to the standard group LASSO.

Table 3.6: Results for the GLASSO (linear regression, $\delta = 3$)

n	τ	avg. error	% correct	avg. false positives	avg. false negatives
40	1/12	3.4738	65.4	0.580	1.012
40	0.1	3.1407	70.5	0.948	0.432

Table 3.7: Results for the GHAL (linear regression, $\delta = 3$)

n	(a, b)	avg. error	% correct	avg. false positives	avg. false negatives
40	$(2, 0.75)$	2.1267	89.6	0.328	0.144
40	$(2, 0.7)$	2.1205	91.1	0.232	0.176

3.4.2 Logistic Regression

In logistic regression with $y_i \in \{-1, 1\}$, one has

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) = \prod_{i=1}^{n} (1 + \exp(-y_i \boldsymbol{\beta}^T \mathbf{x}_i))^{-1}$$

so the log-likelihood is

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) = -\sum_{i=1}^{n} \log(1 + \exp(-y_i \boldsymbol{\beta}^T \mathbf{x}_i)). \tag{3.7}$$

The Jeffreys prior for this likelihood is given by $p(\boldsymbol{\beta}) \propto |X'VX|^{1/2}$, where V is a diagonal matrix with

$$v_{i,i} = \frac{\exp(-\boldsymbol{\beta}^T \mathbf{x}_i)}{[1 + \exp(-\boldsymbol{\beta}^T \mathbf{x}_i)]^2}.$$

As a result, the parametrization-invariant MAP estimate requires us to minimize

$$\boldsymbol{\beta}_{MAP} = \arg\min_{\beta} -\log f(\mathbf{X}|\mathbf{y}, \boldsymbol{\beta}) + \frac{1}{2}\log|X'VX| - \log p(\theta).$$

Unfortunately, while $-\frac{1}{2}\log|X'VX|$ is convex, $\frac{1}{2}\log|X'VX|$ is not so the resulting minimization problem is not convex. However, this does not seem to be a serious issue in our examples as the term $\log|X'VX|$ is relatively constant in the regions of high posterior density and so including the MAP correction has little effect on the results. To test the method, we simulated data using $\boldsymbol{\beta} = (3, 1.5, 0, 0, 2, 0, 0, 0)^T$ and $X \sim N(0, \Sigma)$ with $\Sigma_{i,j} = 0.5^{|i-j|}$ as with the linear regression simulations. We then ran 1000 repetitions of the HAL and the LASSO on this problem with various settings of (a, b) and τ respectively with the results given in Tables 3.8-3.9. An interesting result with this example is that for $(a, b) = (2, 0.1)$ except for $(a_2, b_2, a_5, b_5) = (2, 2, 2, 2)$, the HAL gave poor results due to the correlation of the predictors and the relatively high penalization of β_1. In this case, β_1 was excluded from the model associated with the MAP estimate in every simulation. Using additionally $(a_1, b_1) = (2, 0.5)$ led to a drastic improvement in the results, highlighting the importance the prior hyperparameters can have.

Table 3.8: Results for the LASSO (logistic regression)

n	τ	avg. error	% correct	avg. false positives	avg. false negatives
80	1/7.5	2.8559	62.1	0.387	0.098
80	0.1*	2.7212	93.9	0.008	0.053

* $(\tau_2, \tau_5) = (1, 1)$

Table 3.9: Results for the HAL (logistic regression)

n	(a, b)	avg. error	% correct	avg. false positives	avg. false negatives
80	$(2, 0.65)$	1.3736	65.4	0.33	0.114
80	$(2, 0.1)^*$	3.2084	0.0	0.00	1.000
80	$(2, 0.1)^\dagger$	1.1228	99.2	0.00	0.008

* $(a_2, b_2, a_5, b_5) = (2, 2, 2, 2)$
† $(a_1, b_1, a_2, b_2, a_5, b_5) = (2, 0.5, 2, 2, 2, 2)$

3.4.3 Gaussian Graphical Models

The log-likelihood for this model, after standardization, is

$$\log p(X|\Omega) = \frac{n}{2}\log|\Omega| - \frac{n}{2}\text{tr}(S\Omega)$$

where $S = 1/n\sum_{i=1}^{n} x_i^T \Omega x_i$.

Jeffreys prior for this likelihood is given by $p(\Omega) \propto |\Omega|^{(p+1)/2}$ so we can find the parametrization-invariant MAP using

$$\Omega^{(t+1)} = \arg\max_{\Omega} \frac{n-p-1}{2}\log|\Omega| - \frac{n}{2}\text{tr}(S\Omega) - \sum_{i=1}^{p}\sum_{j=i}^{p} w_{ij}^{(t)}|\Omega_{ij}|$$

where

$$w_{ij}^{(t)} = \frac{a_{ij}+1}{b_{ij}+|\Omega_{ij}^{(t)}|}.$$

In order for the likelihood with the MAP correction term to be concave, we require $n > p+1$ since $-\log\det$ is a convex function.

To test the method, we simulated data using

$$\mathbf{\Omega} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 1.5 & 0 & 0.2 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.3 & 0 & 0.2 & 0 & 0 \\ 0 & 0.2 & 0.3 & 2 & 0 & 0 & 0 & 1.5 \\ 0.5 & 0.8 & 0 & 0 & 1 & 0 & 0.5 & 0 \\ 0 & 0 & 0.2 & 0 & 0 & 0.5 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0.3 & 1.5 & 0 \\ 0 & 0 & 0 & 1.5 & 0 & 0 & 0 & 2 \end{pmatrix}$$

and again used 1000 repetitions of the procedure using the LASSO and the HAL with the results given in Tables 3.10-3.11. The HAL clearly has superior performance when using hyperparameters such that the average number of false positives and false negatives are roughly equal.

Table 3.10: Results for the LASSO (GGM)

n	τ	avg. error	% correct	avg. false positives	avg. false negatives
40	1/45	4.676	23.9	2.789	1.887
40	1/50	4.22	22.3	2.081	2.139

Table 3.11: Results for the HAL (GGM)

n	(a, b)	avg. error	% correct	avg. false positives	avg. false negatives
40	$(1, 0.075)$	2.594	65.4	1.304	1.290
40	$(2, 0.1)$	2.850	57.7	1.343	1.507

3.4.4 Discussion

We have proposed a MAP-based variable selection method using a hierarchical prior for $\boldsymbol{\beta}$ that works reasonably well in practice and brings together a variety of related approaches in the literature. In particular, the estimate itself corresponds to the solution of a non-convex penalized optimization problem, with properties similar to that in Fan and Li (2001), i.e. estimates of large coefficients tend to be penalized less than in standard ℓ_1-penalized optimization approaches, while still being sparse and continuous in the data for some choices of the hyperparameters. A possibly more important contribution is the interpretation the framework gives for various methods that have been proposed outside of Bayesian statistics, in particular for adaptive or one-step methods in the statistics literature and for iteratively reweighted methods in the machine learning and signal processing literature. This interpretation allows for manipulation of the hierarchy in application-specific ways.

A number of open questions remain when using this class of methodology for variable selection. One is how to resolve the issue of finding a global mode of the multimodal posterior. Another is assessing the utility of point estimates when there is little guarantee that the model corresponding to the MAP estimate has significant posterior mass from a Bayesian variable selection perspective. While serious concerns, the major contribution of this chapter thus far is in aiding Bayesian interpretation and generalization of increasingly popular penalized optimization methods amongst practitioners, without advocating their indiscriminate use.

In the next section, a sequence of HAL priors is used in a full Bayesian analysis of

a logistic regression problem where the primary motivation is not variable selection, but exploratory data analysis. As such, many of the concerns raised thus far are no longer applicable.

3.5 Sparsity Path Analysis

In this section, we explore the use of sparsity-inducing priors, and the HAL prior in particular, on regression coefficients to help understand the nature of association signal within "hit regions" of genome-wide association studies. We pay particular attention to graphical representations of posterior statistics obtained from sparsity path analysis (SPA) where we sweep over the setting of the scale (shrinkage / precision) parameter in the prior to explore the space of posterior models obtained over a range of complexities, from very sparse models with all coefficient distributions heavily concentrated around zero, to models with diffuse priors and coefficients distributed around their maximum likelihood estimates. The SPA plots are akin to LASSO plots of MAP estimates but they characterize the complete marginal posterior distributions of the coefficients plotted as a function of the precision of the prior. Generating posterior distributions over a range of prior precisions is computationally challenging but naturally amenable to SMC algorithms indexed on the scale parameter, which are especially computationally efficient when run on graphics cards.

3.5.1 Genome Wide Association Studies

Genome-wide association studies (GWAS) have presented a number of interesting challenges to statisticians (see, e.g., Balding 2006). Conventionally, GWAS use single marker univariate tests of association, testing marker by marker in order to highlight "hit regions" of the genome showing evidence of association signal. The motivation for this work concerns methods in Bayesian sparse multiple regression analysis to characterize and decompose the association signal within such regions spanning possibly hundreds of markers. Hit-regions typically cover loci in high linkage disequilib-

rium (LD) which leads to strong correlation between the markers making multiple regression analysis non-trivial. *A priori* we would expect only a small number of markers to be responsible for the association and hence priors that induce sparsity are an important tool to aid understanding of the genotype-phenotype dependence structure; an interesting question being whether the association signal is consistent with a single causal marker or due to multiple effects.

We restrict attention to case-control GWAS, i.e. binary phenotypes, these being by far the most prevalent although generalisations to other exponential family likelihoods or non-linear models is relatively straightforward.

In the context of GWAS there have been recent illustrations of the use of adaptive sparsity priors and MAP estimation (Hoggart et al. 2008), reviewed and compared in Ayers and Cordell (2010). In Hoggart et al. (2008) they use the Normal-Exponential-Gamma sparsity-prior of Griffin and Brown (2007) to obtain sparse MAP estimates from logistic regression. The use of continuous sparse priors can be contrasted with an alternative approach using two component mixture priors such as in George and McCulloch (1993) which adopt a distribution containing a spike at zero and another component with broad scale. Coefficients are then *a posteriori* classified into either state, relating to whether their corresponding variables (genotypes) are deemed predictively irrelevant or relevant. In the GWAS setting the two-component mixture prior has been explored by Wilson et al. (2010) and Fridley (2009). For model exploration the sparsity prior has some benefits in allowing the statisticians to visualize the regression analysis over a range of scales / model complexities.

We will focus here on the HAL prior of Section 3.2.1, a special case of the generalized t prior of McDonald and Newey (1988) and a full Bayesian analysis, due to its attractive features as a sparsity prior with a simple analytic form and interpretable parameters. In applications, the setting of the scale parameter in the prior is the key task that affects the sparsity of the posterior solution. We believe much is to be gained in exploring the continuum of posterior models formed by sweeping through the scale of the prior, possibly even over regions of low posterior probability, in an exploratory approach we call sparsity path analysis (SPA). This leads towards providing graphical representations of the posterior densities that

arise as we move from the most sparse models with all coefficient densities heavily concentrated around the origin through to models with diffuse priors and coefficients distributed around their maximum likelihood estimates. We stress that even though parts of this model space have low probability, they have utility in gaining a fuller understanding of the predictor-response (genotype-phenotype) dependence structure. SPA is an exploratory tool that is computationally challenging but highly suited to SMC algorithms indexed on the scale parameter and simulated in parallel on a many-core architecture.

The main interest of this section is to consider a full Bayesian analysis of logistic regression models with generalized t priors in the context of GWAS applications for which we develop SMC samplers and their GPU implementation. The GPU implementation is important in practice; in the GWAS analysis it reduces run-time from over five days to hours. We pay particular attention to developing graphical displays of summary statistics from the joint posterior distribution over a wide range of prior precisions, arguing that parts of the model space with low probability have high utility and are informative to the exploratory understanding of regression signal.

3.5.2 Application

3.5.2.1 Model

In Bayesian logistic regression analysis we are given a design matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ and binary phenotypes, or responses, $\mathbf{y} \in \{0,1\}^n$. We adopt a prior $p(\boldsymbol{\beta})$ on the p regression coefficients so that the posterior on $\boldsymbol{\beta}$ is given by (3.1). The likelihood can be written as

$$f(\mathbf{y}|\mathbf{X},\boldsymbol{\beta}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \tag{3.8}$$

$$p_i = \frac{1}{1+\exp(-\mathbf{x}_i\boldsymbol{\beta})} \tag{3.9}$$

We focus here on the HAL prior, but with $(a_j, b_j) = (a, b)$ for $j \in \{1, \ldots, p\}$ and where we define $b = ac$, so that

$$p(\beta_j|a, c) = \frac{1}{2c}\left(1 + \frac{|\beta_j|}{ac}\right)^{-(a+1)} \tag{3.10}$$

Since this is also the parametrization in McDonald and Newey (1988), we will also refer to this prior density as $Gt(a, c)$. We are interested with inferring and graphing the regression coefficients' marginal posterior distributions over a range of values for c for fixed a.

3.5.2.2 Data

In the development and testing of the methods in this work we made use of anonymous genotype data obtained from the Wellcome Trust Centre for Human Genetics (WTCHG), Oxford. The data consists of a subset of single-nucleotide polymorphisms (SNPs) from a genome-wide data set originally gathered in a case-control study to identify colorectal cancer risk alleles. The data set we consider consists of genotype data for 1859 subjects on 184 closely spaced SNPs from a hit-region on chromosome 18q. We standardize the columns of $\mathbf{X}$ to be mean zero and unit standard deviation. A plot of the correlation structure across the first 100 markers in the region is shown in Fig. 3.4. We can see the markers are in high LD with block-like strong correlation structure that makes multiple regression analysis challenging.

We generated pseudo-phenotype data, $\mathbf{y}$, by selecting at random five loci and generating coefficients, $\beta \sim N(0, 0.2^2)$, judged to be realistic signal sizes seen in GWAS (Stephens and Balding 2009). All other coefficients were set to zero. We then generated $\mathbf{y}$ according to (3.8) and (3.9). We considered two scenarios. The first when we construct a genotype matrix of 500 individuals using the five relevant markers and an additional 45 "null" markers whose corresponding coefficients were zero, i.e. a subset of the data from the study. The second data set uses all available markers and all subjects to create a (1859×184) design matrix. To clarify, all covariate data used in the following examples is real genotype data from chromosome 18q on individuals genotyped as part of the WTCHG study. However the phenotype

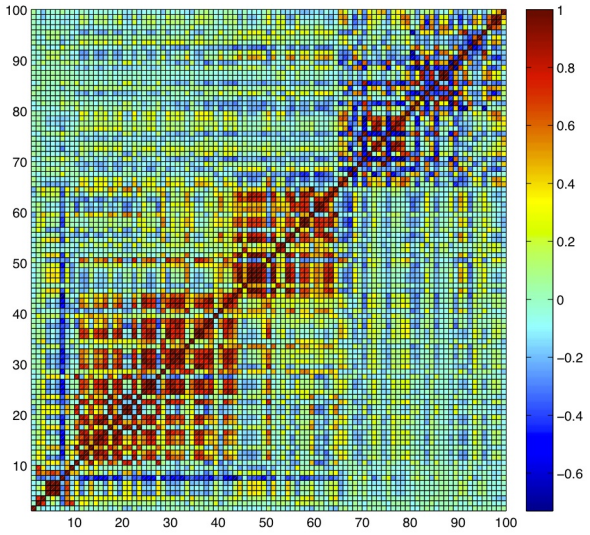


Figure 3.4: Plot of correlation structure across the first 100 adjacent genotype markers from chromosome 18q.

(response variable) $\mathbf{y}$ is simulated due to confidentiality issues.

3.5.3 Sequential Monte Carlo Approach

We propose to explore the posterior distributions of $\boldsymbol{\beta}|\mathbf{X}, \mathbf{y}, a, c$ with varying $c = b/a$ via Sequential Monte Carlo (SMC) sampler methodology (Del Moral et al. 2006). In particular, we let the target distribution at time t be $\pi_t(\boldsymbol{\beta}) = p(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y}, a, b_t/a)$ with $t = \{1, \ldots, T\}$ and some specified sequence $\{b_t\}_{t=1}^T$ where $b_i > b_{i+1}$ and b_T is small enough for the prior to dominate the likelihood with most coefficients' posterior distributions heavily concentrated around zero, and b_1 large enough for the prior to be diffuse and have limited impact on the likelihood. In order to compute posterior distributions over the range of $\{b_t\}_{t=1}^T$ we make use of sequential Monte Carlo samplers indexed on this sequence of scales.

It is important to note that we cannot evaluate

$$\pi_t(\boldsymbol{\beta}) = \frac{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta})p(\boldsymbol{\beta}|a, b_t)}{\int p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta})p(\boldsymbol{\beta}|a, b_t)d\boldsymbol{\beta}}$$

since we cannot evaluate the integral in the denominator, the marginal likelihood of

$\mathbf{y}$. However, we can evaluate $\gamma_t(\boldsymbol{\beta}) = Z_t \pi_t(\boldsymbol{\beta})$ where $Z_t = \int p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) p(\boldsymbol{\beta}|a, b_t) d\boldsymbol{\beta}$. Noting from Section 1.2.4.2 that SMC samplers provide a relative estimate of Z_t at each step of the algorithm, we can see immediately that the methodology will produce as a byproduct estimates of the marginal likelihood of $\mathbf{y}$ for each value of $b \in \{b_t\}_{t=1}^T$.

The algorithm used is given in Algorithm 1.8, where we make use of MCMC kernels within an SMC sampler. Details are given in Section 3.5.3.1 and general tuning guidelines are given in Section 3.5.3.2.

If we additionally give $\log b$ a uniform prior on $[\log b_T, \log b_1]$, these unnormalized marginal densities can also be used to weight the collections of particles at each time, allowing us to approximate the distribution of $\boldsymbol{\beta}$ with b marginalized out.

$$\tilde{\pi}_t^N(d\boldsymbol{\beta}) = \frac{1}{\sum_{t=1}^T \widehat{\frac{Z_t}{Z_1}}} \sum_{t=1}^T \frac{\widehat{Z_t}}{Z_1} \sum_{i=1}^N W_t^{(i)} \delta_{\boldsymbol{\beta}_t^{(i)}}(d\boldsymbol{\beta}) \tag{3.11}$$

Note that this is equivalent to systematic sampling of the values of b, and that an improper uniform prior for $\log b$ on $(-\infty, \infty)$ would induce an improper posterior for $\boldsymbol{\beta}$.

3.5.3.1 Further Details

For analysis of the data with $n = 500$ and $p = 50$, we use $N = 8192$ particles and we consider two settings of $a = 1$ and $a = 4$ with $T = 450$ and $T = 350$ respectively, with the lower T for $a = 4$ explained by the low marginal likelihood associated with small values of b when a is large. We set $b_t = 2(0.98)^{t-1}$ for $t \in \{1, \ldots, T\}$. Given the nature of the posterior, a simple p-dimensional random walk Metropolis-Hastings kernel does not seem to converge quickly. We found that using a cycle of kernels that each update one element of $\boldsymbol{\beta}$ using a random walk proposal with variance 0.25 mixed better for a given amount of computation than a p-dimensional random walk proposal, which is consistent with Neal and Roberts (2006). To ensure that our kernels mix fast enough we construct K_t by cycling this cycle of kernels 5 times. This is computationally demanding since each step of the algorithm consists of $5Np$

MCMC steps, each of which has a complexity in $O(n)$, since the calculation of the likelihood requires us to update $\mathbf{X}\boldsymbol{\beta}$ for a change in one element of $\boldsymbol{\beta}$. Resampling of the particles occurred when the effective sample size (ESS) (Liu and Chen 1995) was below $\frac{3}{4}N$.

In fact, computation on a 2.67GHz Intel Xeon requires approximately 20 hours of computation time with these settings and $T = 450$. However, we implemented the algorithm to run on an NVIDIA 8800 GT graphics card using the Compute Unified Device Architecture (CUDA) parallel computing architecture, using the SMC sampler framework in Chapter 2 to take advantage of the ability to compute the likelihood in parallel for many particles. On this hardware, the time to run the algorithm is approximately 30 minutes for $T = 450$, giving around a 40 fold speedup.

The initial particles for $t = 1$ are obtained by running an MCMC algorithm targeting π_1 for a long time and picking suitably thinned samples to initialize the particles. The weights of each particle start off equal at $1/N$. While not ideal, samples obtained via simple importance sampling have a very low effective sample size for reasonable computational power and, in any case, the SMC sampler methodology requires us to have a rapidly mixing MCMC kernel for our resulting estimates to be good.

The particle with the highest posterior density at each time is used as an initial value for the MAP algorithm described in Section 3.2.1.1. The density of the estimated MAP from this initial value is compared with the MAP obtained by using the previous MAP as an initial value and the estimate with the higher posterior density is chosen as the estimated MAP. This has the effect of both removing variables that the original algorithm has not removed and keeping variables in that the original algorithm has not, as the algorithm used to obtain the MAP is only guaranteed to locate a local mode of the posterior.

While the particular form of the weights of the particles makes it possible to adapt the selection of b_t to allow aggressive changes in b while maintaining a given ESS (Del Moral et al. 2011; Bornn et al. 2010), we found that this scheme had a poor effect on the quality of our samples. This is primarily because we discovered that our cycle length for K_t was not large enough to get the particles to approximate

π_t after a large change in b, despite a small change in ESS. Increasing the cycle length has an essentially linear effect on the computational resources required, so this appears to be a poor option for reducing the computational complexity of the method. In addition, having b_t on a logarithmic schedule enables us to use the simple approximation to the posterior of $\boldsymbol{\beta}|\mathbf{X}, \mathbf{y}, a$ given by (3.11).

For the larger experiment with $n = 1859, p = 184$ and $a = 4$, we use the same parameters except that we used the schedule $b_t = (0.98)^{t-1}$ with $T = 250$. This was sufficient to capture a wide range of model complexities. Nevertheless, the running time for this experiment on our GPU was 7.5 hours. This demonstrates the key practical benefit to using GPUs, as a similar run on a single CPU would take upwards of 5 days.

3.5.3.2 Tuning the SMC Sampler

Following exploratory analysis we found that $b_t = b_1(0.98)^{t-1}$ worked well, and that setting b_1 equal to one or two produced a diffuse enough prior when sample size was in the 100s or 1000s, as is the situation in GWAS.

In practice, one will have to experiment with the parameters of the SMC sampler in order to ensure satisfactory results. In particular, given an MCMC kernel K_t, one must decide how many cycles should be performed per time step to mix adequately in relation to how quickly one wishes to decrease b. In an ideal scenario, K_t produces essentially independent samples from π_t regardless of the current set of particles. This would allow one to set $\{b_t\}_{t=1}^T$ such that computational power is only spent in areas of b that are of interest. In practice, however, K_t will often only mix well locally and so small steps in b are desirable to ensure that the Monte Carlo error in the scheme does not dominate the results. It is difficult to characterize the relationship between the number of cycles of K_t to perform and how aggressively one can decrease b in general. However, in this context, one might not want to decrease b too quickly anyway, so that a smooth picture of the relationship between successive posteriors can be visualized.

The number of particles, N, is a crucial parameter that should be set at such a level that some fraction of N can provide a suitable approximation of each intermediate target density. In practice, the number of effective samples is lower than N and resampling when the ESS is low is an attempt to ensure that this number is controlled. It is standard practice to choose to resample particles when the ESS is below CN for some $C \geq 1/2$. We chose to resample the particles when the ESS fell below $3/4N$.

A recommended way to test whether the sampler is performing acceptably is to run an MCMC scheme with fixed $b = b_t$ for some intermediate value of t and compare the samples one obtains after convergence with those produced by the SMC sampler. This is possible in this context because the posterior is not particularly troublesome to sample from for fixed b with a sensible kernel.

Finally it is worth noting that if the aim was to infer a single distribution given a single setting of $c = b/a$, or with a prior on c, one could achieve better results via one long MCMC run. The benefit of the SMC sampler here is that it provides a unified method to obtain all of the exploratory statistics and estimates of the marginal likelihood in a single run.

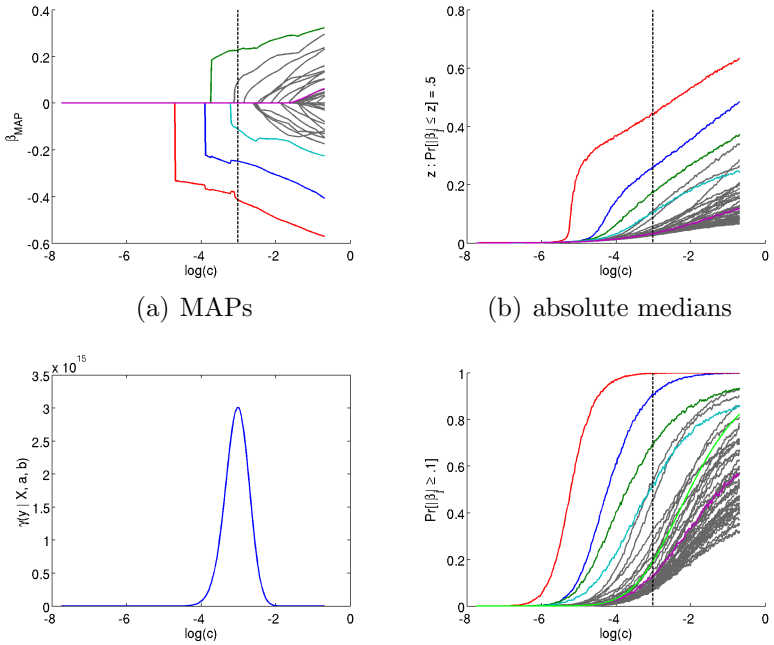
3.5.4 Results

Plots of LASSO MAP estimates as functions of the penalty parameter λ are a highly informative visual tool for statisticians to gain an understanding of the relative predictive importance of covariates over a range of model complexities (Hastie et al. 2009). However, MAP estimates are point estimates that fail to convey the full extent of information contained within the joint posterior distribution. One key aim of this report is to develop full Bayesian exploratory tools that utilize the joint posterior information.

In this section we describe graphical displays of summary statistics contained in SPA results by first considering the $n = 500$, $p = 50$ data set described above. The indices of the variables with non-zero $\boldsymbol{\beta}$'s are $\mathcal{I} = \{10, 14, 24, 31, 37\}$ with

corresponding $\boldsymbol{\beta}_{\mathcal{I}} = \{-0.2538, 0.4578, -0.1873, -0.1498, 0.0996\}$. For the $Gt(a, c)$ prior we found the $Gt(a = 1, c)$, or $Gt(a = 4, c)$, to be good default settings for the degrees of freedom parameter. We ran the SMC-GPU sampler described above and then generated a number of plots using the set of particles and their weights.

3.5.4.1 Path Plots



(a) MAPs (b) absolute medians

(c) posterior density (d) concentrations

Figure 3.5: SPA plots with $a = 4$; using 50 markers around 18q. The 5 non-zero coefficients are indicated by non-grey lines. The vertical dashed line indicates the mode of the marginal likelihood, $p(c|\mathbf{X}, \mathbf{y}, a)$, for c as shown in plot (c).

In Fig. 3.5 we plot four graphs of summary statistics obtained from the SMC sampler across a range of scales $\log c \in (-8, 0)$.

(a) MAP plots: In Fig. 3.5(a) we plot the MAP paths of all 50 coefficients moving from the most sparse models with $c = e^{-8}$ where all $\hat{\boldsymbol{\beta}}_{MAP} = 0$ through to $c = 1$ when all $\hat{\boldsymbol{\beta}}_{MAP} \neq 0$. The true non-zero coefficients are plotted in colors other than gray. One can observe that the MAP paths are non-smooth in $\log c$ such that the modes suddenly jump away from zero. This is a property of the non-log-concavity of the posterior distributions; such that the marginal posterior densities can be bimodal with one mode at $\beta_j = 0$ and another away

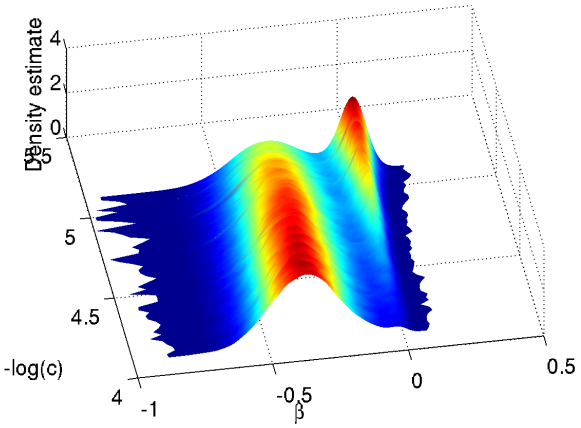
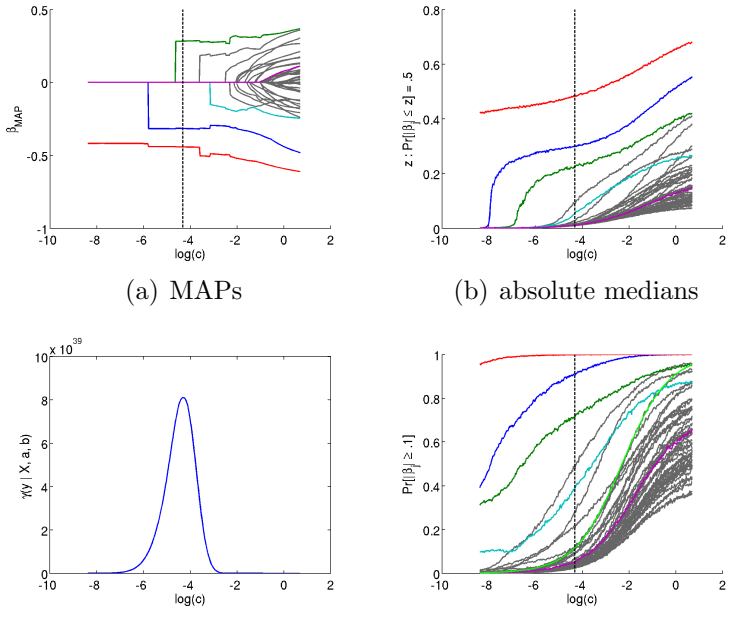


Figure 3.6: Plot of marginal posterior of β_{24} as a function of c, c.f. the red curve in Fig. 3.5(a). We can see the bimodality and hence the jumping of the MAP mode in Fig. 3.5(a) as $\log c$ changes from (-5, -4).



(a) MAPs (b) absolute medians

(c) posterior density (d) concentrations

Figure 3.7: SPA plots with $a = 1$; using 50 markers around 18q. This Figure should be compared with Fig. 3.5 using $a = 4$.

from zero. As c increases the density of the mode at zero decreases and at some point the global mode switches to the mode at $\beta_j \neq 0$. This can be seen clearly in Fig. 3.6 where we plot the marginal posterior distribution of β_{24} over the range $\log c \in (-5, -4)$; c.f. β_{24} is shown as the red curve in Fig. 3.5(a). We can see in Fig. 3.6 the non-concavity of the posterior density and how the global mode jumps from 0 to ≈ -0.4 as $\log c$ is around -4.75.

(b) Median plots: In Fig. 3.5(b) we plot the absolute value of the median of the marginal distribution of β_j's. This is a plot of $z_j(c)$ vs. $\log c$, for $j = 1, \ldots, p$, where,

$$z_j(c) = |\hat{F}^{-1}_{\beta_j}(0.5|c)|,$$

$F^{-1}_{\beta_j}(\cdot)$ is the inverse of the cumulative posterior distribution of β_j,

$$F_{\beta_j}(x|c) = \int_{-\infty}^{x} p(\beta_j|\mathbf{X}, \mathbf{y}, a, c) d\beta_j$$

and $p(\beta_j|\mathbf{X}, \mathbf{y}, a, c)$ is the marginal posterior distribution of β_j given c, i.e. $p(\beta_j|\mathbf{X}, \mathbf{y}, a, c) = \int_{\mathbb{R}^{p-1}} p(\beta_j, \beta_{-j}|\mathbf{X}, \mathbf{y}, a, c) d\beta_{-j}$ where index $\{-j\}$ indicates all indices other than the jth. The plot of absolute medians gives an indication of how quickly the posterior distributions are spreading outward from the origin as the precision of the prior decreases. By plotting on the absolute scale we are better able to compare the coefficients with one another and we also see that, unlike the MAPs, the medians are smooth functions of c.

(c) Posterior for scale c: In Fig. 3.5(c) we show the marginal posterior distribution $p(c|\mathbf{X}, \mathbf{y}, a)$,

$$p(c|\mathbf{X}, \mathbf{y}, a) = \int_{\mathbb{R}^p} p(c, \boldsymbol{\beta}|\mathbf{X}, \mathbf{y}, a) d\boldsymbol{\beta}.$$

The posterior on c graphs the relative evidence for particular prior scale. The mode of Fig. 3.5(c),

$$\tilde{c} = \arg\max_c p(c|\mathbf{X}, \mathbf{y}, a)$$

is indicated on plots (a),(b),(d) by a vertical dashed line.

(d) Concentration plots: In Fig. 3.5(d) we plot the concentration of the marginal posteriors of the β_j's around the origin, as well as the *a priori* concentration (light green). In particular, for user specified tolerance Δ, this is a plot of

$V(c)$ vs. $\log c$ where,

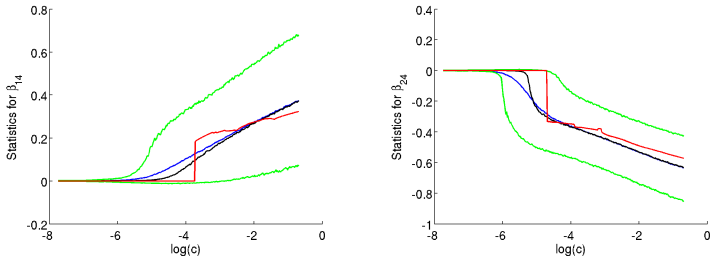
$$V(c) = 1 - \int_{-\Delta}^{\Delta} p(\beta_j|\mathbf{X}, \mathbf{y}, a, c)d\beta_j = 1 - \mathbb{P}[\beta_j \in (-\Delta, \Delta)|\mathbf{X}, \mathbf{y}, a, c]$$

This is a direct measure of the predictive relevance of the corresponding covariate (genotype). In Fig. 3.5(d) we have set $\Delta = 0.1$ although this is a user set parameter that can be specified according to the appropriate level below which a variable is deemed unimportant.

Taken together the SPA plots highlight the influential genotypes and the relative evidence for their predictive strength. We can also gain an understanding of the support in the data for differing values of $\log c$. Having generated the SPA plots it is interesting to compare the results of changing a from 4 to 1. In Fig. 3.7 we show plots corresponding to Fig. 3.5 but using $a = 1$. We can see that the $Gt(a = 1, c)$ induces less sparsity for a fixed value of c but that the mode of the posterior for c, $\tilde{c}$, is smaller so that the MAP estimates at this value are more sparse. Comparing the concentration plots Fig. 3.7(d) and Fig. 3.5(d) at the marginal mode of c we see that for $a = 1$ we see much greater dispersion in the concentration plot.

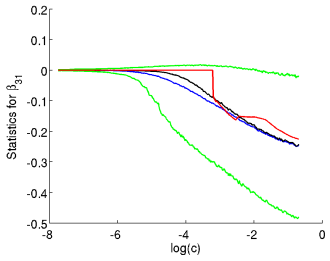
3.5.4.2 Individual Coefficient Plots

Examination of the plots in Fig. 3.5 may highlight to the statistician some interesting predictors to explore in greater detail. Individual coefficient plots of summary statistics can be produced to provide greater information on the posterior distributions. In Fig. 3.8 we show summary plots for four representative coefficients with their 90% credible intervals (green), median (black), mean (blue) and MAP (red). These are obtained from the set of weighted SMC particles. We can see that as expected the mean and median are smooth functions of the prior scale, while the MAP can exhibit the characteristic jumping for bimodal densities. In Fig. 3.9 we show corresponding density estimates. These coefficients were chosen to represent markers with strong association Fig. 3.8(a), weaker association Fig. 3.8(b),(c) and no association Fig. 3.8(d). We can see in the plots for Fig. 3.8(d) and Fig. 3.9(d) that for a null marker with no association signal the MAP appears to be much

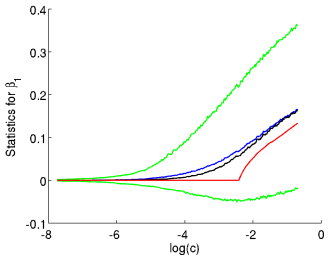


(a) $a = 4$, $j = 14$

(b) $a = 4$, $j = 24$

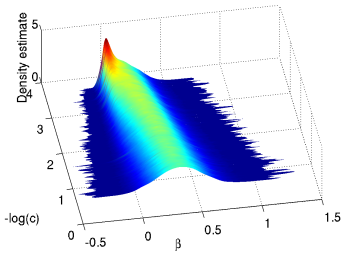


(c) $a = 4$, $j = 31$

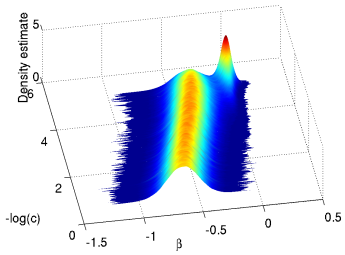


(d) $a = 4$, $j = 1$

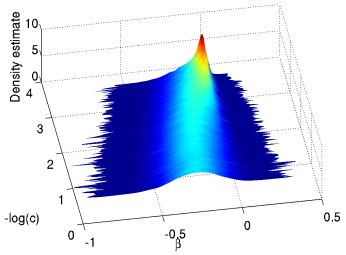
Figure 3.8: Stats for individual coefficients from Fig. 3.5 with $a = 4$. We plot 90% credible intervals (green), median (black), mean (blue) and MAP (red).



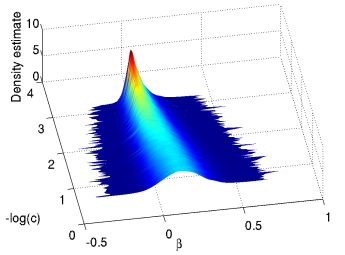
(a) $a = 4$, $j = 14$



(b) $a = 4$, $j = 24$

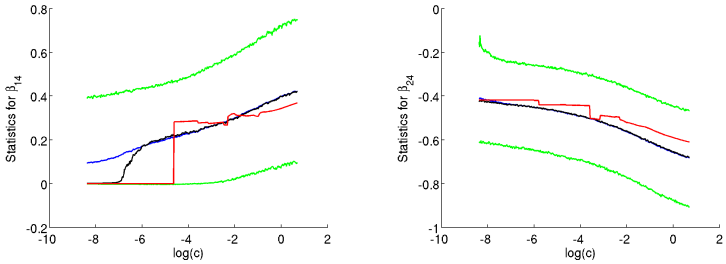


(c) $a = 4$, $j = 31$

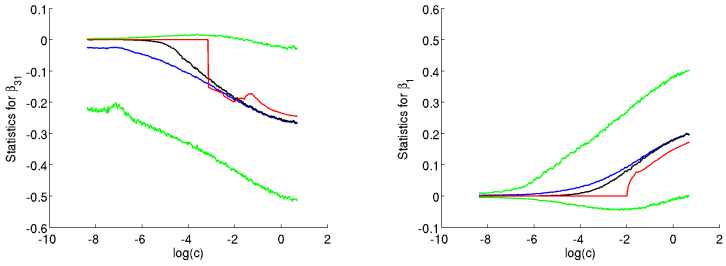


(d) $a = 4$, $j = 1$

Figure 3.9: Posterior density plots corresponding to Fig. 3.8, $a = 4$

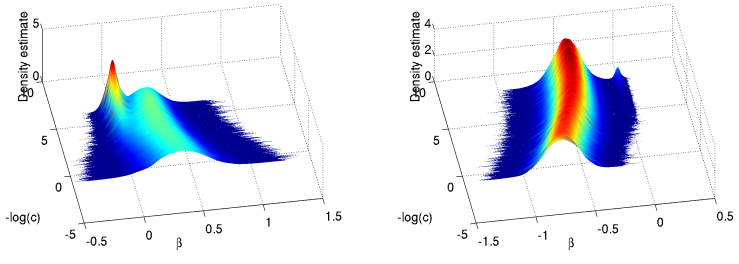


(a) $a = 1$, $j = 14$ (b) $a = 1$, $j = 24$

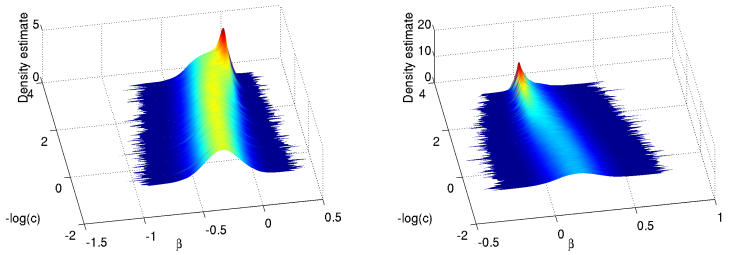


(c) $a = 1$, $j = 31$ (d) $a = 1$, $j = 1$

Figure 3.10: Stats for individual coefficients from Fig. 3.7 with $a = 1$. Compare with Fig. 3.8 where $a = 4$.



(a) $a = 1$, $j = 14$ (b) $a = 1$, $j = 24$



(c) $a = 1$, $j = 31$ (d) $a = 1$, $j = 1$

Figure 3.11: Posterior density plots corresponding to Fig. 3.10, $a = 1$

smoother in $\log c$. Equivalent plots but for $a = 1$ are shown in Fig. 3.10 and Fig. 3.11.

3.5.4.3 Marginal Plots

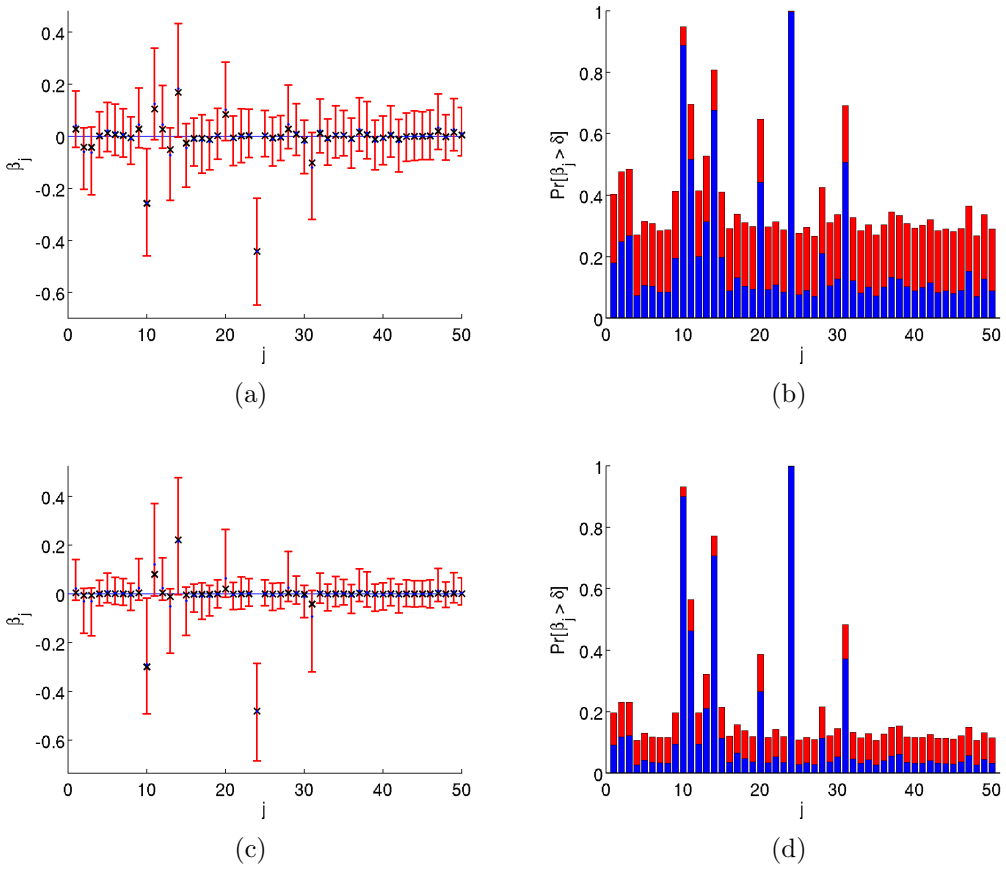


Figure 3.12: Marginal plots: (a) the summary stats from marginal posterior distributions showing medians (crosses), means (blue dots), and 90% credible intervals (bars) for $a = 4$; (b) the marginal concentrations $\Delta = 0.05$ red bars, $\Delta = 0.1$ (blue bars) for $a = 4$; (c) marginal posteriors as in (a) but with $a = 1$ prior; (d) marginal concentrations as in (b) but for $a = 1$, prior.

The SMC sampler also allows us to estimate the marginal posterior probability of $\boldsymbol{\beta}$, using (3.11), integrating over the uncertainty in c,

$$p(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y}, a) = \int_0^\infty p(\boldsymbol{\beta}, c|\mathbf{X}, \mathbf{y}, a)dc$$

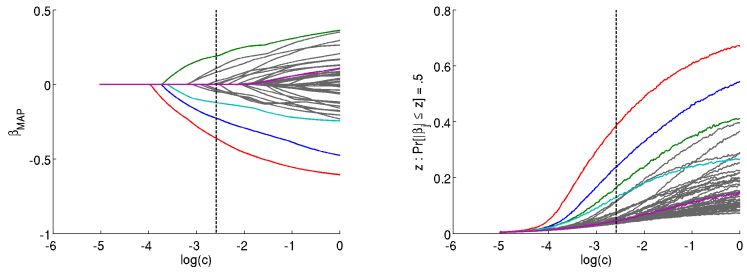
Moreover, we can also calculate the marginal posterior concentration away from

zero, for given tolerance Δ as,

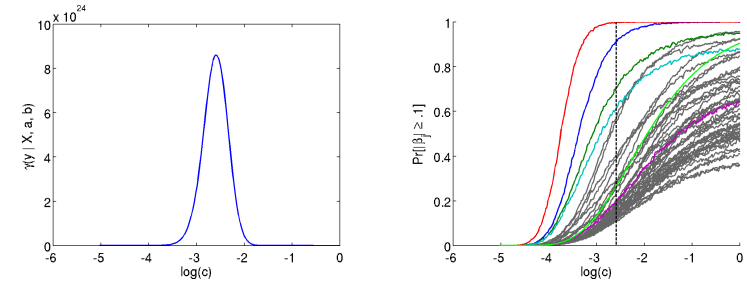
$$V = 1 - \int_0^{\infty} \int_{-\Delta}^{\Delta} p(\beta_j, c|\mathbf{X}, \mathbf{y}, a) d\beta_j dc$$

In Fig. 3.12 we plot summaries for the marginal posterior distributions of $\boldsymbol{\beta}$ as well as the marginal concentrations, for $a = 4$ and $a = 1$. We can see in Fig. 3.12 that the marginal plots provide a useful overview of the relative importance of each variable.

3.5.4.4 Comparison to the Double-Exponential Prior



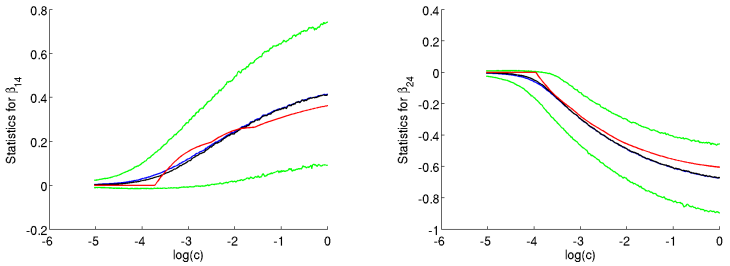
(a) MAPs (b) medians



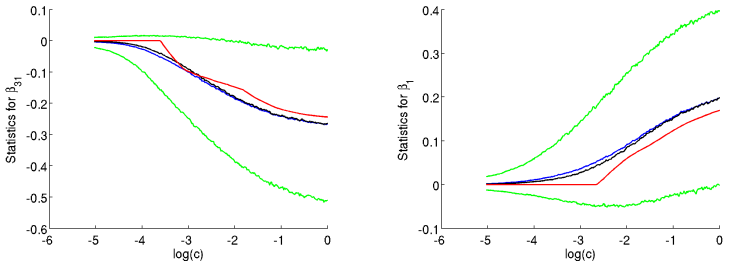
(c) marginal density (d) concentrations

Figure 3.13: SPA plots for the double-exponential prior, $Gt(a \to \infty, c)$.

It is informative to compare the results for the $Gt(a, c)$ prior above with that of the double-exponential prior $p(\beta_j) \propto \exp(-|\beta_j|/c)$ which is obtained as a generalized t prior with $q = 1$ as $a \to \infty$. In Fig. 3.13 we show SPA plots for this case. We can see the much smoother paths of the MAP compared with Fig 3.5. This can also be seen in the individual coefficient plots shown in Fig. 3.14 and Fig. 3.15. It is interesting to investigate in more detail the posterior density of β_{24} the coefficient with strongest evidence of association. This is shown in Fig. 3.16 and should be compared to Fig. 3.6 for $a = 4$. The concavity of the double-exponential prior causes

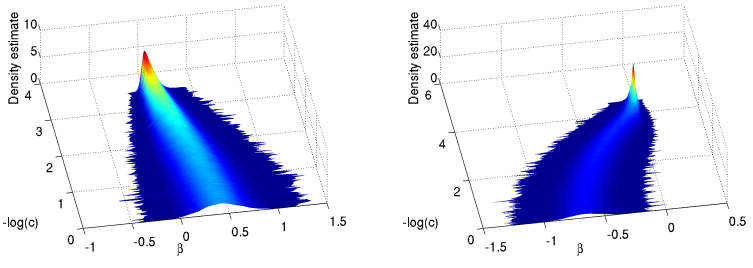


(a) double-exponential $j = 14$ (b) double-exponential $j = 24$

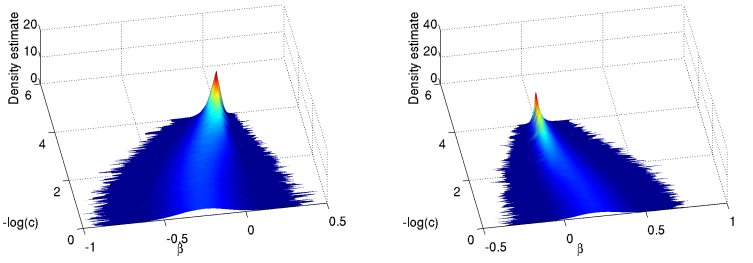


(c) double-exponential $j = 31$ (d) double-exponential $j = 1$

Figure 3.14: Stats for individual coefficients from Fig. 3.13 with double-exponential prior. Compare with Fig. 3.8 where $a = 4$.



(a) double-exponential $j = 14$ (b) double-exponential $j = 24$



(c) double-exponential $j = 31$ (d) double-exponential $j = 1$

Figure 3.15: Posterior density plots corresponding to Fig. 3.14, $a \to \infty$.

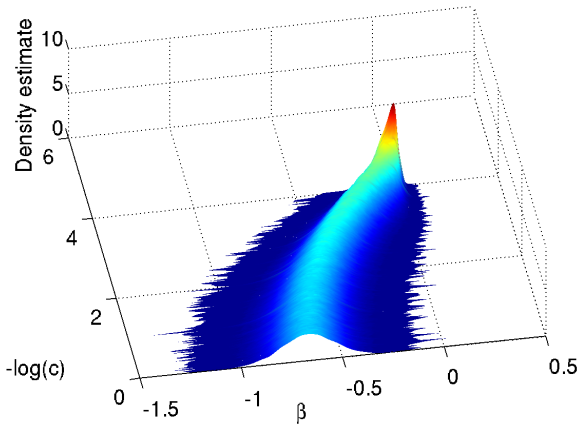


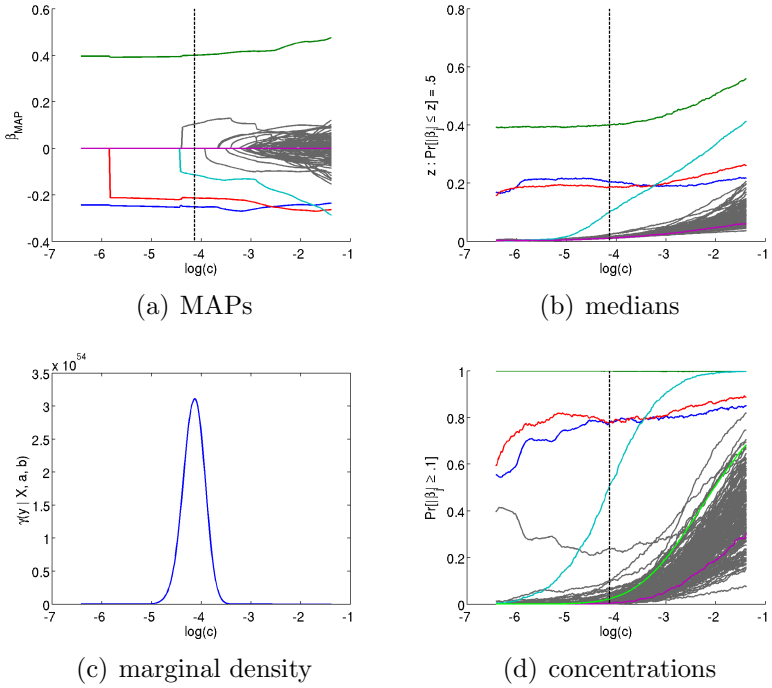
Figure 3.16: Plot of marginal posterior of β_{24} as a function of c for double-exponential prior $a \to \infty$. Under the double-exponential prior, $a \to \infty$, we see the posterior remains log-concave and hence the MAP is a smooth function of $\log c$; as compared to Fig. 3.6.

the posterior distribution to move smoothly towards being concentrated around 0, unlike the behaviour seen with the generalized t prior.

3.5.4.5 Large Data Set

We next analysed the larger data set with $n = 1859$ and $p = 184$. The indices of the non-zero coefficients are $\mathcal{I} = \{108, 22, 5, 117, 162\}$ with the same values retained for $\boldsymbol{\beta}_{\mathcal{I}} = \{-0.2538, 0.4578, -0.1873, -0.1498, 0.0996\}$. The SPA plots are shown in Fig. 3.17 with corresponding individual coefficient plots in Fig. 3.18 and Fig. 3.19. We can see in Fig. 3.17 that there is much stronger evidence that coefficients $\{108, 22, 5, 117\}$ are important. Interestingly variable 162 is still masked by other predictors. In comparison with Fig. 3.5(c) we observe that a smaller value of c is supported by the larger data set.

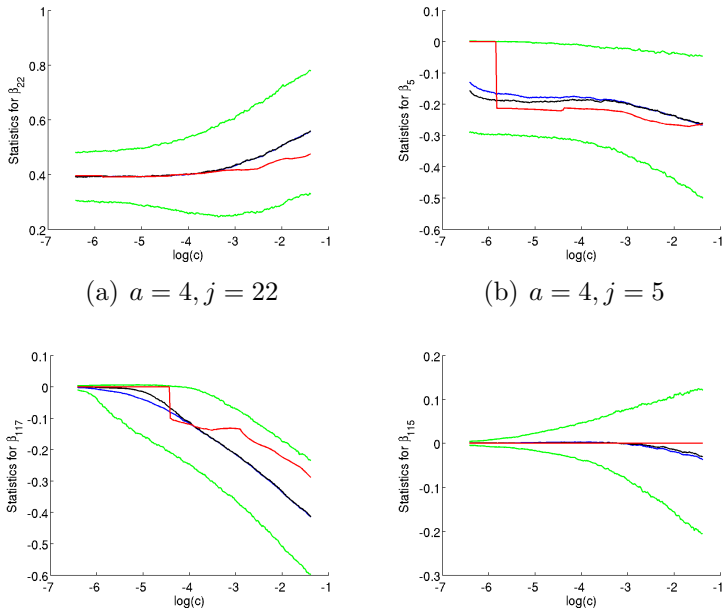
From Fig. 3.17(d) we can see that one null-coefficient, β_{107}, (with true $\beta_{107} = 0$) appears to have some evidence of predictive importance. In Fig. 3.20 we show summaries of the marginal distributions and concentration plots having integrated over the uncertainty in c. The null-coefficient SNP turns out to be situated adjacent to a predictive SNP $\beta_{108} = -0.2538$ shown as the blue line in Fig. 3.17(a), and the two markers are in high LD with correlation coefficient $C(X_{107}, X_{108}) = 0.87$,



(a) MAPs (b) medians

(c) marginal density (d) concentrations

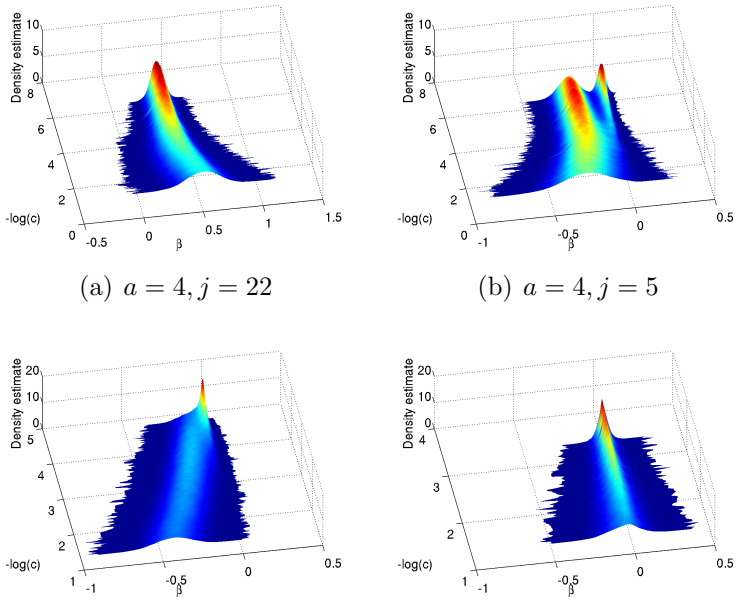
Figure 3.17: SPA plots using the larger data set $n = 1859$, $p = 184$.



(a) $a = 4, j = 22$ (b) $a = 4, j = 5$

(c) $a = 4, j = 117$ (d) $a = 4, j = 115$

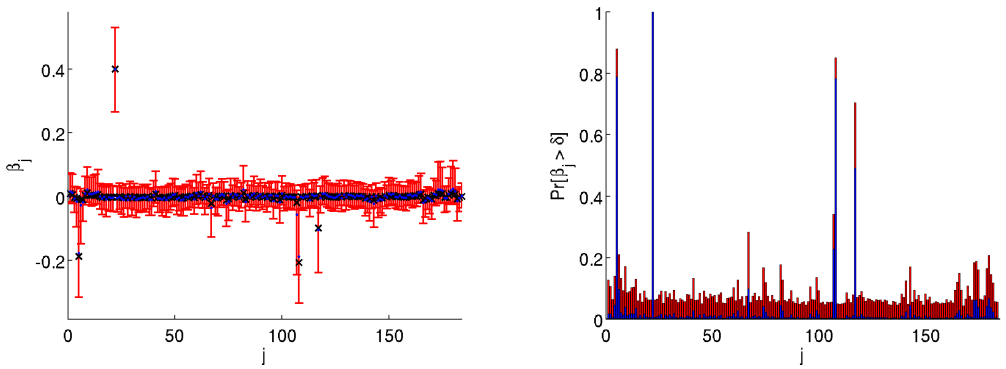
Figure 3.18: Summary statistics for posterior distributions for individual coefficients, 90% credible intervals (green), median (black), mean (blue) and MAP (red).



(a) $a = 4, j = 22$ (b) $a = 4, j = 5$

(c) $a = 4, j = 117$ (d) $a = 4, j = 115$

Figure 3.19: Posterior density plots of coefficients in Fig. 3.18



(a) (b)

Figure 3.20: Marginal plots: (a) the summary stats from marginal posterior distributions showing medians (crosses), means (blue dots), and 90% credible intervals (bars) for $a = 4$; (b) the marginal concentrations $\Delta = 0.05$ red bars, $\Delta = 0.1$ (blue bars) for $a = 4$

leaving some ambiguity in the source of the association signal. The plot of absolute medians Fig. 3.17(b) helps partly resolve this issue pointing to only four clear association signals when c is close to its marginal mode. Summary statistics for the posterior marginal distribution of β_{107} are shown in Fig. 3.21. Here we can see that the MAP (red line) holds at zero for most values of c, adding support that this is a "null" marker, although the credible intervals (green lines) show there is considerable uncertainty in the actual value. Note also that β_{107} is not related to the null-coefficient that is non-zero in the MAP estimate at the marginal mode of c. In this way we can see how complex association signals can be explored, and gain a better understanding of the source of such association signals.

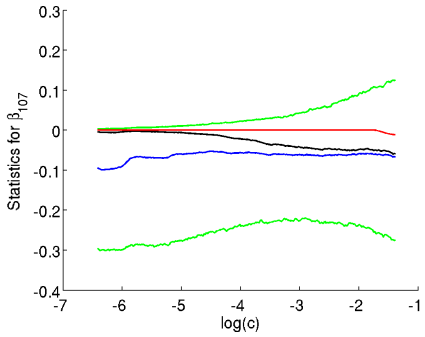


Figure 3.21: Statistics for individual coefficient β_{107}.

3.5.5 Discussion

In this section, an exploratory approach using a sparsity prior has been introduced that we call Bayesian sparsity path analysis to aid in the understanding of genetic association data. The approach involves graphical summaries of posterior densities of coefficients obtained by sweeping over a range of prior precisions, including values with low posterior probability, in order to informally characterize the space of models from the most sparse to the most complex.

The use of SMC methodology is ideally suited to this inference task, by defining an auxiliary distribution for each value of the scale of the prior. The resulting collection of weighted particles provides us with approximations of the posterior distributions associated with each scale in addition to estimates of the marginal likelihood and allows for more robust MAP estimation. The simulations are computationally de-

manding and would take days worth of run-time using conventional single-threaded CPU processing. To alleviate this we make use of many-core GPU parallel processing producing around a 20-40 fold improvement in run-times. This has real benefit in allowing for data analysis within a working day for the larger data set we considered.

The general methodology of SPA is not restricted to the $Gt(a, c)$ or HAL prior and can be implemented for other sparsity priors. Indeed, there have been a number of sparsity priors advocated recently in the literature. Some important contributions include the normal-gamma, normal-exponential-gamma (NEG) and the Horseshoe priors, (see Griffin and Brown 2010a; Caron and Doucet 2008; Griffin and Brown 2007; Carvalho et al. 2010). Like all these priors the $Gt(\cdot, \cdot)$ shares the properties of sparsity and adaptive shrinkage, allowing for less shrinkage on large coefficients, and it also has a scale-mixture representation which facilitates an efficient EM algorithm to obtain MAP estimates.

	$\log p(\beta_j)$
double-exponential	$-\lambda\lvert\beta_j\rvert + C$
normal-gamma	$(\frac{1}{2} - \lambda)\log\lvert\beta_j\rvert - \log K_{\lambda-1/2}\left(\frac{\lvert\beta_j\rvert}{\lambda}\right) + C$
NEG	$-\frac{\beta^2}{4\lambda^2} - \log D_{-2(\lambda+\frac{1}{2})}\left(\frac{\lvert\beta\rvert}{\lambda}\right) + C$
Horseshoe	$\log\left[\int_0^\infty (2\tau^2\lambda^2)^{-1/2}\exp\left(-\frac{\beta_j^2}{2\tau^2\lambda^2}\right)(1+\lambda)^{-2}d\lambda\right] + C$
$Gt(a, c)$	$-(a+1)\log\left(1 + \frac{\lvert\beta_j\rvert}{ac}\right) + C$

Table 3.12: Log densities for the double-exponential and various sparsity-priors, up to an additive prior-specific constant C. $K(\cdot)$ denotes a Bessel function of the third kind and $D_v(z)$ is the parabolic cylinder function. The Horseshoe prior does not have an analytic form.

Table 3.12 gives the log densities for the sparsity-priors mentioned above. An immediate benefit of the $Gt(a, c)$ density is that it has a simple analytic form compared with other sparsity-priors and is well known and understood by statisticians, which in turn aids explanation to data owners. The $Gt(a, c)$ density is also easy to compute relative to the NEG, normal-gamma and Horseshoe priors which all involve

non-standard computation, making them more challenging to implement and less amenable to fast parallel GPU simulation. Finally, the setting of the parameters of the generalized t is helped as statisticians are used to thinking about degrees of freedom and scale parameters of Student's densities. This could have an impact on the attractiveness of the prior to data owners.

Chapter 4

The Active Particle View of MCMC Algorithms

4.1 Introduction

In this chapter, a perspective on reversible MCMC kernels is developed that can be used to derive a large number of existing MCMC algorithms as well as provide insight into methodological opportunities in many situations. As usual, we are interested in sampling from a distribution with density $\pi : \mathcal{X} \to \mathbb{R}^+$ that can be computed pointwise up to a normalizing constant. The approach here is to explicitly denote the joint distribution of random variables produced within an MCMC kernel via an extended target density that admits π as a marginal density of what can be called an 'active particle'.

In some situations, the proposed framework can be viewed as a natural generalization of Metropolis-Hastings kernels to the case where one proposes multiple candidate points within a kernel, but this is perhaps superficial in that the framework does not introduce any novel operations. Instead, the extended target densities provide insight into the possibilities that can arise from the explicit inclusion of auxiliary variables in a joint density. This is particularly helpful in generalizing the arguments for detailed balance that have been proffered for specific algorithms.

An attempt is made to introduce the methodology in order of increasing complexity, when possible. The chapter begins with a presentation of classical MCMC algorithms, followed by extensions to involve an arbitrary but fixed number of proposals and generalizations of pseudo-marginal type algorithms. An extension that allows the number of proposals to become both random and dependent on existing proposals is then presented, before the applications to approximate Bayesian computation and Bayesian variable selection are discussed.

4.2 Active Particle View of MCMC

The standard perspective on reversible MCMC kernels is to consider the target density π and the proposal density q separately, and "accept" proposals from q in such a way that π-reversibility is guaranteed, as in Algorithm 1.5. An alternative perspective is to define an extended target density $\tilde{\pi}$, which admits π as a marginal and describes the conditional distribution of random variables used in the process of constructing a reversible MCMC chain. This approach provides a large degree of flexibility in interpreting the auxiliary random variables that are used in constructing π-stationary, reversible Markov chains.

4.2.1 Classical MCMC Algorithms

We begin by showing how Barker's method (Barker 1965) and the Metropolis-Hastings kernel (Hastings 1970) arise from this perspective. We first define random variables $X^{(1)}, X^{(2)} \in \mathcal{X}$ and $K \in \{1, 2\}$ and an extended target density

$$\tilde{\pi}(K = k, X^{(1:2)} = x^{(1:2)}) = \frac{1}{2}\pi(x^{(k)})q(x^{(\bar{k})}|x^{(k)}) \tag{4.1}$$

i.e., marginally K is uniform on $\{1, 2\}$ and conditional upon $K = k$, $X^{(k)}$ is distributed according to π and $X^{(\bar{k})}$ is distributed according to $q(\cdot|X^{(k)})$, where we adopt the notation $\bar{k} = 2$, if $k = 1$ and $\bar{k} = 1$, if $k = 2$.

4.2.1.1 Gibbs Sampling & Barker's Method

One way to sample according to (4.1) is entirely via Gibbs sampling. In this simple setting it seems that a deterministic cycle in which one alternates sampling $X^{(\bar{k})}|k, x^{(k)} \sim q(\cdot|x^{(k)})$ and then $K|x^{(1:2)}$ via

$$\begin{aligned}\tilde{\pi}(K = k|x^{(1:2)}) &= \frac{\tilde{\pi}(k, x^{(1:2)})}{\sum_{j=1}^{2} \tilde{\pi}(j, x^{(1:2)})} \\ &= \frac{\pi(x^{(k)})q(x^{(\bar{k})}|x^{(k)})}{\pi(x^{(1)})q(x^{(2)}|x^{(1)}) + \pi(x^{(2)})q(x^{(1)}|x^{(2)})}\end{aligned}$$

is sensible. The algorithm that results from such a scheme is Barker's method and is given in Algorithm 4.1.

Algorithm 4.1 Barker's Method

Given $(k_0, x_0^{(k_0)})$, at times $t = 0, 1, \ldots$

1. Sample $x_{t+1}^{(\bar{k}_t)} \sim q(\cdot|x_t^{(k_t)})$ and set $x_{t+1}^{(k_t)} = x_t^{(k_t)}$.
2. With probability $\frac{\tilde{\pi}(1,x^{(1:2)})}{\sum_{j=1}^{2} \tilde{\pi}(j,x^{(1:2)})}$, set $k_{t+1} = 1$. Otherwise, set $k_{t+1} = 2$.

4.2.1.2 Metropolis-Hastings

An alternative to Gibbs sampling K conditional on $x^{(1:2)}$ is to deterministically propose the swap move $(k, x^{(1:2)}) \to (\bar{k}, x^{(1:2)})$ and accept it with the Metropolis acceptance ratio:

$$\min\left\{1, \frac{\tilde{\pi}(\bar{k}, x^{(1:2)})}{\tilde{\pi}(k, x^{(1:2)})}\right\} = \min\left\{1, \frac{\pi(x^{(\bar{k})})q(x^{(k)}|x^{(\bar{k})})}{\pi(x^{(k)})q(x^{(\bar{k})}|x^{(k)})}\right\}$$

A cycle of Gibbs sampling $X^{(\bar{k})}|k, x^{(k)}$ followed by this deterministic swap attempt is exactly the Metropolis-Hastings algorithm, given in Algorithm 4.2 and which is equivalent to Algorithm 1.5.

Algorithm 4.2 Joint Distribution View of Metropolis-Hastings

Given $(k_0, x_0^{(k_0)})$, at times $t = 0, 1, \ldots$

1. Sample $x_{t+1}^{(\bar{k}_t)} \sim q(\cdot|x_t^{(k_t)})$ and set $x_{t+1}^{(k_t)} = x_t^{(k_t)}$.
2. With probability $\min\left\{1, \frac{\pi(x^{(\bar{k})})q(x^{(k)}|x^{(\bar{k})})}{\pi(x^{(k)})q(x^{(\bar{k})}|x^{(k)})}\right\}$, set $k_{t+1} = \bar{k}_t$. Otherwise, set $k_{t+1} = k_t$.

4.2.1.3 Reversible Jump Metropolis-Hastings

If $\mathcal{X} = \bigcup_{i\in\mathcal{I}}(\{i\} \times \mathbb{R}^{n_i})$ denotes a trans-dimensional space, the active particle view of MCMC presents a natural interpretation of the reversible jump methodology in Green (1995). Indeed, there is no requirement that $\mathbf{x}^{(k)} = (i^{(k)}, \mathbf{z}^{(k)})$ and $\mathbf{x}^{(\bar{k})} = (i^{(\bar{k})}, \mathbf{z}^{(\bar{k})})$ have $i^{(k)} = i^{(\bar{k})}$ and the presence of both variables in the joint density $\tilde{\pi}$ means that the dimension-matching requirement of Green (1995) is met automatically. The Jacobian that appears in that methodology arises within the density $q(\mathbf{x}^{(\bar{k})}|\mathbf{x}^{(k)})$ or $q(\mathbf{x}^{(k)}|\mathbf{x}^{(\bar{k})})$ when they encode proposals for some components via bijective, differentiable functions, i.e. a reversible change of variables is used.

4.3 Multiple Proposals

While we are still interested in sampling according to π, we adjust our focus to the use of more samples. This is largely motivated by both increased access to parallel computing and, for computational reasons explained in Section 1.3, an imperative to consider parallelizable algorithms. Using the active particle view of MCMC, this can be done via the joint density

$$\tilde{\pi}^N(k, x^{(1:N)}) = \frac{1}{N}\pi(x^{(k)})q(x^{(-k)}|k, x^{(k)}). \tag{4.2}$$

i.e., marginally K is uniform on $\{1, \ldots, N\}$ and conditional upon $K = k$, $X^{(k)}$ is distributed according to π and $X^{(-k)}$ is distributed according to $q(\cdot|k, x^{(k)})$, where we adopt the notation $-k = \{1, \ldots, N\} \setminus \{k\}$.

A possible form of q is where the variables $x^{(-k)}$ are conditionally independent and exchangeable given $x^{(k)}$, i.e. we have $q(x^{(-k)}|x^{(k)}) = \prod_{i=1,i\neq k}^{N} q(x^{(i)}|x^{(k)})$. For example, this could denote a random walk proposal centred at $x^{(k)}$ for each $i \in -k$. A further restriction could have each proposal independent of $x^{(k)}$, i.e. $q(x^{(-k)}|x^{(k)}) = \prod_{i=1,i\neq k}^{N} q(x^{(i)})$.

In the following, special attention will be paid to the computational resources required to make use of extra samples. In particular, we wish to avoid $O(N^2)$ complexity if possible since this can lead to severe scaling issues.

4.3.1 Barker's Method Generalization

We now consider Gibbs sampling $X^{(-k)}|k, x^{(k)}$ and $K|x^{(1:N)}$ as an analogue to Barker's method in this setting. The latter step requires the computation of the probability $\tilde{\pi}^N(K = l|x^{(1:N)})$, which is given by

$$\tilde{\pi}^N(l|x^{(1:N)}) = \frac{\pi(x^{(l)})q(x^{(-l)}|l, x^{(l)})}{\sum_{i=1}^{N} \pi(x^{(i)})q(x^{(-i)}|i, x^{(i)})}$$

Importantly, the computation of this probability takes $O(N^2)$ time due to the form of the denominator in general. One can consider, for example, $q(x^{(-i)}|x^{(i)}) = \prod_{j=1,j\neq i}^{N} q(x^{(j)}|x^{(i)})$.

In the special case where the joint density $q(x^{(1:N)})$ is defined and one can compute each marginal density $q_i(x^{(i)})$, we have instead

$$\tilde{\pi}^N(l|x^{(1:N)}) = \frac{\pi(x^{(l)})/q_l(x^{(l)})}{\sum_{i=1}^{N} \pi(x^{(i)})/q_i(x^{(i)})}$$

via division of both numerator and denominator by this joint density. As such, the computation takes $O(N)$ time in this case. An example of this special case is when one defines $q(x^{(1:N)}) = \prod_{i=1}^{N} q(x^{(i)})$ so that the proposal has the form $q(x^{(-k)}|x^{(k)}) = \prod_{i=1,i\neq k}^{N} q(x^{(i)})$. Note, however, that this special case precludes the use of many commonly used MCMC proposals, e.g. a random walk centred at $x^{(k)}$.

Another issue with the general scheme above is that $\tilde{\pi}^N(l|x^{(1:N)})$ depends on the density $q(x^{(-l)}|l, x^{(l)})$, which can be considerably lower than $q(x^{(-k)}|k, x^{(k)})$ for large N, e.g. when the proposal has the form $q(x^{(-k)}|x^{(k)}) = \prod_{j=1,j\neq k}^{N} q(x^{(j)}|x^{(k)})$.

4.3.2 Metropolis-Hastings Generalization

4.3.2.1 Uniform Swap

We now consider the analogous procedure but attempt a swap move instead of sampling K conditional upon $x^{(1:N)}$, say by picking an index l uniformly at random from $\{1, \ldots, N\} \setminus \{k\}$ and accepting the move $(k, x^{(1:N)}) \to (l, x^{(1:N)})$ with probability

$$\alpha((k, x^{(1:N)}), (l, x^{(1:N)})) = \min\left\{1, \frac{\pi(x^{(l)})q(x^{(-l)}|l, x^{(l)})}{\pi(x^{(k)})q(x^{(-k)}|k, x^{(k)})}\right\}.$$

While this is $O(N)$, this scheme doesn't make much sense since we pick l uniformly and it does not make good use of the information brought by the samples when N is large.

4.3.2.2 Weight Functions

One way to use all $N-1$ samples from q more effectively is to pick one of the samples to swap with a probability that we believe will encourage better mixing of the Markov chain. One can define any positive weight function $w(j, k, x^{(1:N)})$ which is then used to select a sample to perform a swap move. While still considering the target (4.2), we first perform a Gibbs step to sample $X^{(-k)}$ conditional upon k and $X^{(k)}$ and then attempt the swap move $(k, x^{(1:N)}) \to (l, x^{(1:N)})$ where l is a realization of a random variable L with probability mass function

$$\mathbb{P}[L = l] = \frac{w(l, k, x^{(1:N)})}{\sum_{j=1,j\neq k}^{N} w(j, k, x^{(1:N)})}$$

for $l \in \{1, \ldots, N\} \setminus \{k\}$ and 0 otherwise. The acceptance probability of this move is

$$\min\left\{1, \frac{\tilde{\pi}^N(l, x^{(1:N)})w(k, l, x^{(1:N)})}{\tilde{\pi}^N(k, x^{(1:N)})w(l, k, x^{(1:N)})} \frac{\sum_{i=1, i\neq k}^N w(i, k, x^{(1:N)})}{\sum_{i=1, i\neq l}^N w(i, l, x^{(1:N)})}\right\}.$$

Note that the use of a weight function is just a convenient way to describe the conditional distribution of L given $(k, x^{(1:N)})$.

In the special case where we have

$$w(l, k, x^{(1:N)}) = \pi(x^{(l)})q(x^{(-l)}|l, x^{(l)})\lambda(l, k, x^{(1:N)})$$

with $\lambda(l, k, x^{(1:N)}) = \lambda(k, l, x^{(1:N)})$ and $\lambda(l, k, x^{(1:N)}) > 0$ whenever $\tilde{\pi}^N(l, x^{(1:N)}) > 0$ then the acceptance probability of the swap move $(k, x^{(1:N)}) \to (l, x^{(1:N)})$ is

$$\min\left\{1, \frac{\sum_{i=1, i\neq k}^N w(i, k, x^{(1:N)})}{\sum_{i=1, i\neq l}^N w(i, l, x^{(1:N)})}\right\}.$$

The function λ is introduced to allow for a certain degree of flexibility in defining the weight function while retaining this form for the acceptance probability.

In the case where $w(l, k, x^{(1:N)}) = \pi(x^{(l)})q(x^{(-l)}|l, x^{(l)})$, i.e. we set $\lambda(l, k, x^{(1:N)}) = 1$ and we sample the swap index using

$$\mathbb{P}[L = l] = \frac{w(l, k, x^{(1:N)})}{\sum_{j=1}^N w(j, k, x^{(1:N)})}$$

for $l \in \{1, \ldots, N\}$ and 0 otherwise, then this corresponds to the Barker generalization and the acceptance probability is always 1. As such, Gibbs sampling of the index is just a special case of a move within the Metropolis-Hastings framework and so we will avoid special attention to this particular case. In addition, this type of move is inferior to a swap move that does not consider the degenerate swap $L = k$, in the sense of reducing the asymptotic variance of Monte Carlo estimates using the samples from the MCMC chain (see Peskun 1973; Tierney 1998).

This scheme is still not particularly good at this level of generality since the complexity of the move is typically in $O(N^2)$, with a particular exception being when the joint density $q(x^{(1:N)})$ is defined.

4.3.2.3 Multiple-Try Method Generalization

A possible way to avoid both the $O(N^2)$ computation associated with the acceptance probability in some cases and also its dependence on $q(x^{(-l)}|x^{(l)})$ despite the fact that $x^{(-k)}$ is generated according to $q(\cdot|x^{(k)})$ is to additionally sample new particles $x^{-(k,l)}$ for the purpose of attempting the swap move.

Given q, we denote by $q_{i|j}$ the marginal density of $x^{(i)}$ when $x^{(-j)}$ has density $q(x^{(-j)}|x^{(j)})$, that is $q_{i|j}(x^{(i)}|x^{(j)}) = \int_{\mathcal{X}^{N-2}} q(x^{(-j)}|x^{(j)})dx^{-(i,j)}$. Consider the weight function

$$w(j, k, x^{(1:N)}) = \pi(x^{(j)})q_{k|j}(x^{(k)}|x^{(j)})\lambda_{j,k}(x^{(j)}, x^{(k)}) \tag{4.3}$$

where $\lambda_{i,j}(x, y) = \lambda_{j,i}(y, x)$ and $\lambda_{i,j}(x, y) > 0$ when $\pi(x)q_{j|i}(y|x) > 0$. Note that use of this weight function requires being able to compute marginal densities of q. Let us also define another set of auxiliary random variables $z^{(1:N)}$ where $z^{(i)} \in \mathcal{X}$ for $i \in \{1, \ldots, N\}$.

If we sample $x^{(-k)} \sim q(\cdot|k, x^{(k)})$, l using this weight function, set $z^{(k,l)} = x^{(k,l)}$ and then sample $z^{-(k,l)} \sim q(\cdot|l, z^{(l)}, z^{(k)})$, where

$$q(z^{-(k,l)}|l, z^{(l)}, z^{(k)}) = \frac{q(z^{(-l)}|l, z^{(l)})}{q_{k|l}(z^{(k)}|z^{(l)})}$$

the Metropolis-Hastings move $(k, x^{(1:N)}) \to (l, z^{(1:N)})$ has acceptance probability

$$\min\left\{1, \frac{\sum_{j=1,j\neq k}^{N} w(j, k, x^{(1:N)})}{\sum_{j=1,j\neq l}^{N} w(j, l, z^{(1:N)})}\right\}.$$

In the special case where $q(x^{(-k)}|k, x^{(k)}) = \prod_{i=1,i\neq k}^{N} q_{i|k}(x^{(i)}|x^{(k)})$ and

$$q(z^{-(k,l)}|l, z^{(l)}, z^{(k)}) = \prod_{i=1,i\notin\{k,l\}}^{N} q_{i|l}(z^{(i)}|z^{(l)})$$

this runs in $O(N)$ time. If additionally we have that each $q_{i|j} = q$ is identical, the resulting scheme is the multiple-try Metropolis method of Liu et al. (2000).

One should note that the $x^{(-k)}$ are in fact the forward proposals, one of whom is proposed as the active particle, and the $z^{(-l)}$ are the backwards proposals used to calculate the probability of picking $z^{(k)} = x^{(k)}$ in the reverse move, which can be confusing notationally. This confusion can be cleared up by extending the target distribution, as is done in the next section. Another comment is that the multiple-try method corresponds to this particular choice of the weight functions and constraint on the proposal but these restrictions are not essential to the correctness of the algorithm. Indeed, the general acceptance probability for the move $(k, x^{(1:N)}) \to (l, z^{(1:N)})$ with any weight function is

$$\min\left\{1, \frac{\pi(z^{(l)})q_{k|l}(z^{(k)}|z^{(l)})w(k,l,z^{(1:N)})}{\pi(x^{(k)})q_{l|k}(x^{(l)}|x^{(k)})w(l,k,x^{(1:N)})}\frac{\sum_{j=1,j\neq k}^{N} w(j,k,x^{(1:N)})}{\sum_{j=1,j\neq l}^{N} w(j,l,z^{(1:N)})}\right\}.$$

Lifting the restrictions mentioned is particularly important, as we will see later, for situations in which we cannot evaluate the marginal conditionals $q_{j|k}(z^{(j)}|z^{(k)})$ and to take full advantage of the methodological opportunities this perspective brings to MCMC.

4.3.2.4 Alternative Formulation and Generalization

The use of a Metropolis-Hastings step above is in some ways antithetic to the purpose of the joint distribution view of MCMC since the variables $z^{-(k,l)}$ are not defined in the target density $\tilde{\pi}^N$. We can include them by further extending the target density

$$\begin{aligned}\tilde{\pi}^N(k,l,x^{(1:N)},z^{(1:N)}) = &\frac{1}{N}\pi(x^{(k)})q(x^{(-k)}|k,x^{(k)}) \\ &\times \frac{w(l,k,x^{(1:N)})}{\sum_{j\neq k} w(j,k,x^{(1:N)})}\delta_{x^{(k,l)}}(z^{(k,l)})q(z^{-(k,l)}|l,z^{(l)},z^{(k)})\end{aligned}$$

where the marginal distribution of k is uniform on $\{1,\ldots,N\}$, the conditional distribution of $x^{(k)}$ given k is π, and one can read from left to right the conditional densities of the remaining variables. The move $(k, x^{(1:N)}) \to (l, z^{(1:N)})$ described above then corresponds in this extended target to the swap move $(k, l, x^{(1:N)}, z^{(1:N)}) \to (l, k, z^{(1:N)}, x^{(1:N)})$ and has the same acceptance probability.

In the interest of clarity, it is helpful to relabel the variables in this extended target

density to make clear which are the forward proposals and which are the backward proposals. We choose $z^{(1:N)}$ to denote the forward proposals and $x^{(-k)}$ to denote the backward proposals. This gives the following extended target density

$$\begin{aligned}\tilde{\pi}^N(k, l, x^{(1:N)}, z^{(1:N)}) = &\frac{1}{N}\pi(x^{(k)})\delta_{x^{(k)}}(z^{(k)})q(z^{(-k)}|k, z^{(k)}) \\ &\times \frac{w(l, k, z^{(1:N)})}{\sum_{j=1,\neq k}^{N} w(j, k, z^{(1:N)})}\delta_{z^{(l)}}(x^{(l)})q(x^{-(k,l)}|l, x^{(l)}, x^{(k)}).\end{aligned}$$

The acceptance probability of the swap $(k, l, x^{(1:N)}, z^{(1:N)}) \rightarrow (l, k, z^{(1:N)}, x^{(1:N)})$ is then given by

$$\min\left\{1, \frac{\pi(z^{(l)})q_{k|l}(z^{(k)}|z^{(l)})w(k, l, x^{(1:N)})}{\pi(x^{(k)})q_{l|k}(x^{(l)}|x^{(k)})w(l, k, z^{(1:N)})}\frac{\sum_{j=1,j\neq k}^{N} w(j, k, z^{(1:N)})}{\sum_{j=1,j\neq l}^{N} w(j, l, x^{(1:N)})}\right\}.$$

While incorporating the additional backwards random variables introduced in Section 4.3.2.3, and the potential confusion arising from the distinct nature of $x^{(k)}$ and $x^{(-k)}$ before this relabelling, it is in some sense superfluous to require $z^{(k,l)} = x^{(k,l)}$ in the target density. In addition, it can obscure interpretation since $z^{(k)}$ is not a forward proposal and $x^{(l)}$ is not a backward proposal. A mild generalization, then, consists of introducing a closely related, generic proposal density q and weight function w and defining the extended target density as

$$\begin{aligned}\tilde{\pi}^N(k, l, x^{(1:N)}, z^{(1:N)}) = &\frac{1}{N}\pi(x^{(k)})q(z^{(1:N)}|k, x^{(k)}) \\ &\times \frac{w(l, k, x^{(k)}, z^{(1:N)})}{\sum_{j=1}^{N} w(j, k, x^{(k)}, z^{(1:N)})}q(x^{(-k)}|l, z^{(l)}, x^{(k)})\end{aligned} \tag{4.4}$$

where the weight function used to denote the conditional probability of l now depends on k, $x^{(k)}$ and N proposed variables $z^{(1:N)}$. In this target, we have N potentially distinct proposals instead of $N - 1$. The acceptance probability of the Metropolis swap $(k, l, x^{(1:N)}, z^{(1:N)}) \rightarrow (l, k, z^{(1:N)}, x^{(1:N)})$ is given by

$$\min\left\{1, \frac{\pi(z^{(l)})q_{k|l}(x^{(k)}|z^{(l)})w(k, l, z^{(l)}, x^{(1:N)})}{\pi(x^{(k)})q_{l|k}(z^{(l)}|x^{(k)})w(l, k, x^{(k)}, z^{(1:N)})}\frac{\sum_{j=1}^{N} w(j, k, x^{(k)}, z^{(1:N)})}{\sum_{j=1}^{N} w(j, l, z^{(l)}, x^{(1:N)})}\right\}$$

and the previous procedure can be seen as a cycle of kernels involving first Gibbs sampling $(z^{(1:N)}, l, x^{(-k)})$ conditional on $(k, x^{(k)})$ and then attempting this swap move.

In general, the Gibbs sampling step is considered as the application of a separate kernel and so the outcome of this step persists even if the swap move is rejected. Of course, the values of variables other than $(k, x^{(k)})$ do not affect the dynamics of the chain when such a cycle is used. The form of (4.4) will be built upon in the remainder of this chapter, due to its ease of interpretation.

The multiple-try Metropolis method (Liu et al. 2000) with this extended target corresponds to using $q(z^{(1:N)}|k, x^{(k)}) = \prod_{j=1}^{N} q_{j|k}(z^{(j)}|x^{(k)})$ and the weight function

$$w(j, k, x^{(k)}, z^{(1:N)}) = \pi(z^{(j)}) q_{k|j}(x^{(k)}|z^{(j)}) \lambda(z^{(j)}, x^{(k)})$$

for all $j \in \{1, \ldots, N\}$, but where $q_{j|k} = q$ are identical for all $k \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, N\} \setminus \{k\}$. Pseudo-code for the MTM algorithm is provided in Algorithm 4.3.

Algorithm 4.3 The Multiple-Try Method

Given $(k_0, x_0^{(k_0)})$, at times $t = 0, 1, \ldots$

1. Set $k = k_t$ and $x^{(k)} = x_t^{(k_t)}$.
2. For $i \in \{1, \ldots, N\}$, sample $z^{(i)} \sim q(\cdot|x^{(k)})$.
3. Sample $L = l$ with probability $\frac{\pi(z^{(l)}) q(x^{(k)}|z^{(l)}) \lambda(z^{(l)}, x^{(k)})}{\sum_{j=1}^{N} \pi(z^{(j)}) q(x^{(k)}|z^{(j)}) \lambda(z^{(j)}, x^{(k)})}$.
4. For $i \in \{1, \ldots, N\} \setminus \{k\}$, sample $x^{(i)} \sim q(\cdot|z^{(l)})$.
5. With probability

$$\min \left\{ 1, \frac{\sum_{j=1}^{N} \pi(z^{(j)}) q(x^{(k)}|z^{(j)}) \lambda(z^{(j)}, x^{(k)})}{\sum_{j=1}^{N} \pi(x^{(j)}) q(z^{(l)}|x^{(j)}) \lambda(x^{(j)}, z^{(l)})} \right\},$$

set $(k_{t+1}, l_{t+1}, x_{t+1}^{(1:N)}, z_{t+1}^{(1:N)}) = (l, k_t, z^{(1:N)}, x^{(1:N)})$.
Otherwise, set $(k_{t+1}, l_{t+1}, x_{t+1}^{(1:N)}, z_{t+1}^{(1:N)}) = (k_t, l, x^{(1:N)}, z^{(1:N)})$.

4.3.3 Remarks

At this point, some comments are in order. The derivation of the multiple-try method as a special case of the extension of the Metropolis-Hastings algorithm from $N = 2$, or $N = 1$ using the alternative representation above, to arbitrary N does not

only provide an alternative perspective on this methodology. The explicit definition of random variables proposed via the Gibbs step enables us to immediately verify that the Markov kernel this type of move defines is π-invariant without resorting to a proof that detailed balance holds via symmetry arguments as in Liu et al. (2000). This contribution is more obvious in light of the fact that any combination of proposal distribution q and weight function w with adequate support is theoretically valid, although the computational complexity of sampling l and computing the acceptance ratio are highly dependent on these choices.

In the material that follows, this framework will be further developed to accommodate adaptation of N, the introduction of additional parameters θ and sophisticated sequential proposals. The set of existing methods that are then special cases of this general framework include pseudo-marginal approaches (Beaumont 2003) and extensions (Andrieu and Roberts 2009) as well as delayed rejection samplers (Tierney and Mira 1999; Mira 2001). In the next chapter, we will see that particle MCMC methods (Andrieu et al. 2010), the stochastic potential switching method (Mak 2005) and auxiliary variable methods for dealing with doubly intractable distributions (Møller et al. 2006; Murray et al. 2006) can all be described from this perspective. In each of these cases, novel and substantive methodological innovations arise cleanly from the application of this general framework.

Related perspectives on MCMC kernels have appeared recently in Storvik (2011) and Neal (2011). The former uses a joint representation of random variables that admits π as a marginal, as is done here. The latter introduces the notion of an auxiliary space $\mathcal{Y}$ and kernels $K_{1:3}$ satisfying

$$\int_{\mathcal{Y}^2} \pi(x)K_1(x,y)K_2(y,y')K_3(y',x')dy'dy = \pi(x'). \tag{4.5}$$

In the context of this chapter, K_2 is a Markov kernel with equilibrium distribution given by $\tilde{\pi}^N$, a density on $\mathcal{Y}$, and the framework we present can be seen as providing a general way to construct kernels $K_{1:3}$ that satisfy this property in a way that both complements and extends other work. Informally, the variable x in (4.5) denotes the active particle, and y typically includes x as a component as well as other auxiliary variables. K_1 essentially samples y conditional upon x, and K_2 attempts a swap move, where y' is rearrangement of the components of y if the move is accepted.

This swap move includes an attempt to change the value of the active particle. The active particle associated with y' is then usually equivalent to x', so that K_3 involves discarding all other auxiliary variables. This represents only a subset of possible $K_{1:3}$ satisfying (4.5) and minor variations do occur in the material. However, (4.5) is perhaps too general to work with compared to the idea of cycling Gibbs steps and Metropolis swap moves, which provides a naturally constructive approach.

Both Storvik (2011) and Neal (2011) note that there are many possibilities for the definition of auxiliary variables on an extended space and provide examples of their use. This chapter, and the next, go into considerably more depth and explicitly detail more general strategies for the definition of these auxiliary variables and develop a novel strategy to allow the number of auxiliary variables itself to be a random variable. Again, it is worth noting that in all cases, the resulting MCMC kernels are just Metropolis-Hastings kernels on an extended space so the contribution of the material in this light is in detailing useful strategies for specifying how the space can be extended in ways that are advantageous computationally.

The developments presented here can be seen as falling under the general umbrella of population-based MCMC in that many of the auxiliary variables are defined as taking values in $\mathcal{X}$. However, there is a fundamental difference compared to most existing methods, an exception being Liu et al. (2000), in that only one of these variables is at stationarity distributed according to a density that one cannot sample according to easily. This distinction is crucial in understanding the motivation and development of the material in this chapter. The MCMC kernels developed, however, could be used within more traditional population-based methods.

It is important to note that the use of this extended target density does not preclude the use of other MCMC kernels. In particular, in the target density $x^{(k)}$ is, conditionally on k, distributed according to π. As such, this perspective gives insight into individual MCMC kernels that can still be cycled and mixed as per the algorithm designer's wishes, with auxiliary variables sampled as required when using a particular kernel. One should note that it is possible to define proposal densities and weight functions such that the resulting Markov kernel is reducible, and care should be taken to avoid such choices if this kernel is not cycled or mixed with other

kernels with equilibrium distribution π.

Finally, a crucial point that will be used later is that a significant detail has been omitted in the construction of the extended target thus far. This is the observation that the backward proposal need not be the same as the forward proposal and we will from this point on use the notation $\bar{q}$ to denote the backward proposal. We will also use the notation $\tilde{\pi}^N(l|k, x^{(k)}, z^{(1:N)})$ to denote the conditional probability of l, which has thus far been described as

$$\tilde{\pi}^N(l|k, x^{(k)}, z^{(1:N)}) = \frac{w(l, k, x^{(k)}, z^{(1:N)})}{\sum_{j=1}^N w(j, k, x^{(k)}, z^{(1:N)})}.$$

The extended target density can then be written

$$\tilde{\pi}^N(k, l, x^{(1:N)}, z^{(1:N)}) = \\ \frac{1}{N}\pi(x^{(k)})q(z^{(1:N)}|k, x^{(k)})\tilde{\pi}^N(l|k, x^{(k)}, z^{(1:N)})\bar{q}(x^{(-k)}|k, l, x^{(k)}, z^{(l)})$$

and the swap move $(k, l, x^{(1:N)}, z^{(1:N)}) \to (l, k, z^{(1:N)}, x^{(1:N)})$ has acceptance probability

$$\min\left\{1, \frac{\pi(z^{(l)})q(x^{(1:N)}|l, z^{(l)})\tilde{\pi}^N(k|l, z^{(l)}, x^{(1:N)})\bar{q}(z^{(-l)}|l, k, z^{(l)}, x^{(k)})}{\pi(x^{(k)})q(z^{(1:N)}|k, x^{(k)})\tilde{\pi}^N(l|k, x^{(k)}, z^{(1:N)})\bar{q}(x^{(-k)}|k, l, x^{(k)}, z^{(l)})}\right\}.$$

Of course, the relationship between q and $\bar{q}$ will affect the acceptance probability, and large disparities can lead to poor mixing.

Another possibility highlighted by this perspective is that one need not include all variables other than $(k, x^{(k)})$ in the Gibbs step before attempting a swap move, in contrast to how the material has been presented thus far. Indeed, any valid Gibbs step is possible, although one should ensure that the resulting Markov kernel is irreducible. An example of this, that has been utilized in some algorithms from another perspective, is that $x^{(-k)}$ need not be sampled if its conditional density under $\tilde{\pi}^N$ does not depend on l or $z^{(1:N)}$. In this case, one could sample only $(z^{(1:N)}, l) \sim \tilde{\pi}^N(\cdot|k, x^{(1:N)}) = \tilde{\pi}^N(\cdot|k, x^{(k)})$ in what can be described as a 'lazy' Gibbs step, before attempting a swap move. When such lazy steps are not performed, one can see that the values $(k, x^{(k)})$ define a Markov chain. However, if $x^{(-k)}$ is not sampled, then this is no longer true but $(k, x^{(1:N)})$ instead defines a Markov chain.

We can refer to $(k, x^{(k)})$ and $(k, x^{(1:N)})$, respectively, as collapsed Markov chains, or we can say that in these situations the Markov chain can be collapsed to $(k, x^{(k)})$ and $(k, x^{(1:N)})$.

There are many other possible extensions, which cannot be covered here exhaustively since the space of potential specific forms of the extended target is extremely large. In general, the primary ingredient is that $\tilde{\pi}^N(x^{(k)}|k) = \pi(x^{(k)})$, while the remaining auxiliary variables are defined conditionally upon $(k, x^{(k)})$ so as to preserve this property. One could, for example, define a target in which $z^{(1:N)}, x^{(1:N)}$ and even l are proposed in a dependent fashion, i.e.

$$\tilde{\pi}^N(k, l, x^{(1:N)}, z^{(1:N)}) = \frac{1}{N}\pi(x^{(k)})q(l, z^{(1:N)}, x^{(-k)}|k, x^{(k)})$$

with the resulting acceptance probability of a swap move being

$$\min\left\{1, \frac{\pi(z^{(l)})q(k, x^{(1:N)}, z^{(-l)}|l, z^{(l)})}{\pi(x^{(k)})q(l, z^{(1:N)}, x^{(-k)}|k, x^{(k)})}\right\}.$$

A final remark at this stage is that the marginal distribution of k under $\tilde{\pi}$ has been defined as uniform on $\{1, \ldots, N\}$, which is well-founded in many situations in which the marginal distribution of l is also uniform. However, this distribution is arbitrary from the perspective of ensuring that $\tilde{\pi}^N(x^{(k)}|k) = \pi(x^{(k)})$ and could be modified in order to speed convergence, particularly when the $z^{(1:N)}$ are not exchangeable.

4.4 Pseudo-Marginal Approaches

We now consider the case where the target density is defined on a space $\Theta \times \mathcal{X}$, and it is of interest to treat $\theta \in \Theta$ and $x \in \mathcal{X}$ differently in an MCMC scheme. This could be because $\pi(\theta) = \int_{\mathcal{X}} \pi(\theta, x)dx$ defines the actual distribution of interest but whose density cannot be evaluated, or because the dependencies between x and θ are such that the types of proposals one would like to use for each component are different. In the former case, one can define the marginal algorithm that cannot be implemented and the pseudo-marginal approaches in this section are in some

sense attempts to emulate this algorithm. This marginal algorithm is presented in Algorithm 4.4.

Algorithm 4.4 Marginal Algorithm

Given θ_0, at times $t = 0, 1, \ldots$

1. Set $\theta = \theta_t$ and sample $\theta' \sim g(\cdot|\theta)$.
2. With probability
$$\min\left\{1, \frac{\pi(\theta')g(\theta|\theta')}{\pi(\theta)g(\theta'|\theta)}\right\},$$
set $\theta_{t+1} = \theta'$.
Otherwise, set $\theta_{t+1} = \theta$.

We will focus on this former case, motivated by Beaumont (2003), which introduced the grouped-independence Metropolis-Hastings (GIMH) algorithm for MCMC exploration of $\pi(\theta)$ in cases when this density is not available computationally. In this setting, the variable x is a classic auxiliary variable such that the density $\pi(\theta, x)$ is available computationally and, as mentioned above, admits $\pi(\theta)$ as its marginal.

Consider the extended target density

$$\begin{aligned}\tilde{\pi}^N(k, l, \theta, \theta', x^{(1:N)}, z^{(1:N)}) = \frac{1}{N}\pi(\theta, x^{(k)})q(\theta', z^{(1:N)}|\theta, k, x^{(k)}) \\ \times \tilde{\pi}^N(l|\theta, \theta', k, x^{(k)}, z^{(1:N)})\bar{q}(x^{(-k)}|\theta, \theta', k, l, x^{(k)}, z^{(1:N)}) \quad (4.6)\end{aligned}$$

where

$$\tilde{\pi}^N(l|\theta, \theta', k, x^{(k)}, z^{(1:N)}) = \frac{w(l, (\theta, k, x^{(k)}), (\theta', z^{(1:N)}))}{\sum_{j=1}^N w(j, (\theta, k, x^{(k)}), (\theta', z^{(1:N)}))}.$$

One can then use a combination of Gibbs sampling and swap moves as above to construct a Markov chain whose invariant distribution admits π as a marginal distribution.

4.4.1 Grouped-Independence Type Samplers

One possible choice of $q(\theta', z^{(1:N)}|\theta, k, x^{(k)})$ is

$$q(\theta', z^{(1:N)}|\theta, k, x^{(k)}) = g(\theta'|\theta) \prod_{i=1}^{N} q(z^{(i)}|\theta')$$

with a natural choice in this context for $\bar{q}$ being

$$\bar{q}(x^{(-k)}|\theta, \theta', k, l, x^{(k)}, z^{(1:N)}) = \prod_{i=1, i\neq k}^{N} q(x^{(i)}|\theta).$$

If we define the conditional probability of l via

$$\tilde{\pi}^N(l|k, \theta, x^{(k)}, \theta', z^{(1:N)}) = \frac{\pi(\theta', z^{(l)})/q(z^{(l)}|\theta')}{\sum_{j=1}^{N} \pi(\theta', z^{(j)})/q(z^{(j)}|\theta')},$$

the acceptance probability of a swap move, after Gibbs sampling $(\theta', z^{(1:N)}, l, x^{(-k)}) \sim \tilde{\pi}^N(\cdot|k, x^{(k)})$, is given by

$$\min\left\{1, \frac{g(\theta|\theta')\sum_{j=1}^{N} \pi(\theta', z^{(j)})/q(z^{(j)}|\theta')}{g(\theta'|\theta)\sum_{j=1}^{N} \pi(\theta, x^{(j)})/q(x^{(j)}|\theta)}\right\}.$$

The algorithm corresponding to this scheme is given in Algorithm 4.5, which can be thought of as the multiple-try version of the grouped-independence Metropolis-Hastings (GIMH). It is important to note that this algorithm is, in fact, not the GIMH algorithm, which is given in Algorithm 4.6. In particular, step 5 of the algorithm highlights the difference, which is that in GIMH, the values of $x^{(-k)}$ do not change until a swap move is made. The inclusion of the variables k and l in the GIMH algorithm is for ease of interpretation only, since one can perform the algorithm without sampling l or keeping track of k. This is a departure from the algorithms presented so far in that the state of the Markov chain cannot be collapsed into only those values which are, in equilibrium, distributed according to π. The grouped-independence multiple-try (GIMT) algorithm can be seen as a special case of the multiple-try framework by noting that the conditional distribution of l corresponds to choosing the weight function

$$w(j, (\theta, k, x^{(k)}), (\theta', z^{(1:N)})) = \pi(\theta', z^{(j)})q(x^{(k)}|\theta)\lambda((\theta', z^{(j)}), (\theta, x^{(k)}))$$

with $\lambda((\theta', z^{(j)}), (\theta, x^{(k)})) = [q(z^{(j)}|\theta')q(x^{(k)}|\theta)]^{-1}$.

Algorithm 4.5 Grouped-Independence Multiple-Try

Given $(k_0, \theta_0, x_0^{(k_0)})$, at times $t = 0, 1, \ldots$

1. Set $k = k_t$ and $(\theta, x^{(k)}) = (\theta_t, x_t^{(k_t)})$.
2. Sample $\theta' \sim g(\cdot|\theta)$
3. For $i \in \{1, \ldots, N\}$, sample $z^{(i)} \sim q(\cdot|\theta')$.
4. Sample $L = l$ with probability $\frac{\pi(\theta', z^{(l)})/q(z^{(l)}|\theta')}{\sum_{j=1}^N \pi(\theta', z^{(j)})/q(z^{(j)}|\theta')}$.
5. For $i \in \{1, \ldots, N\} \setminus \{k\}$, sample $x^{(i)} \sim q(\cdot|\theta)$.
6. With probability

$$\min\left\{1, \frac{g(\theta|\theta')\sum_{j=1}^N \pi(\theta', z^{(j)})/q(z^{(j)}|\theta')}{g(\theta'|\theta)\sum_{j=1}^N \pi(\theta, x^{(j)})/q(x^{(j)}|\theta)}\right\},$$

set $(k_{t+1}, \theta_{t+1}, x_{t+1}^{(k_{t+1})}) = (l, \theta', z^{(l)})$.
Otherwise, set $(k_{t+1}, \theta_{t+1}, x_{t+1}^{(k_{t+1})}) = (k, \theta, x^{(k)})$.

That GIMH does not sample from $\bar{q}$ is, in fact, only possible because of the specific form of $\bar{q}$; specifically that it is independent of $(\theta', l, z^{(1:N)})$. As such, $x^{(-k)}$ remains distributed according to $\tilde{\pi}^N(\cdot|\theta, \theta', k, l, x^{(k)}, z^{(1:N)})$ until such a time as a swap occurs. Alternatively, one can see that the distribution of $(\theta', l, z^{(1:N)})$ conditional upon $(\theta, k, x^{(k)})$ is independent of $x^{(-k)}$ and so the GIMH algorithm corresponds to a valid Gibbs step only on the former variables. An alternative perspective is that the GIMH algorithm implements a scheme suggested by an extended target in which k, the index of the active particle, has been integrated out:

$$\tilde{\pi}^N(\theta, \theta', x^{(1:N)}, z^{(1:N)}) = \left[\frac{1}{N}\sum_{k=1}^N \pi(\theta, x^{(k)}) \prod_{j=1, j\neq k}^N q(x^{(j)}|\theta)\right] g(\theta'|\theta) \prod_{j=1}^N q(z^{(j)}|\theta').$$

The marginal density of $(\theta, x^{(1:N)})$ can be rewritten as

$$\tilde{\pi}^N(\theta, x^{(1:N)}) = \pi(\theta)\frac{1}{N}\sum_{k=1}^N \pi(x^{(k)}|\theta) \prod_{j=1, j\neq k}^N q(x^{(j)}|\theta)$$

to make clear that the marginal density of θ is π and also that the marginalization of k in this setup hides the fact that one of the particles of $x^{(1:N)}$ is, at equilibrium, distributed according to $\pi(\cdot|\theta)$.

Algorithm 4.6 Grouped-Independence Metropolis-Hastings

Given $(k_0, \theta_0, x_0^{(1:N)})$, at times $t = 0, 1, \ldots$

1. Set $k = k_t$ and $(\theta, x^{(k)}) = (\theta_t, x_t^{(k_t)})$.
2. Sample $\theta' \sim g(\cdot|\theta)$
3. For $i \in \{1, \ldots, N\}$, sample $z^{(i)} \sim q(\cdot|\theta')$.
4. Sample $L = l$ with probability $\frac{\pi(\theta', z^{(l)})/q(z^{(l)}|\theta')}{\sum_{j=1}^N \pi(\theta', z^{(j)})/q(z^{(j)}|\theta')}$.
5. Set $x^{(-k)} = x_t^{(-k)}$.
6. With probability

$$\min\left\{1, \frac{g(\theta|\theta')\sum_{j=1}^N \pi(\theta', z^{(j)})/q(z^{(j)}|\theta')}{g(\theta'|\theta)\sum_{j=1}^N \pi(\theta, x^{(j)})/q(x^{(j)}|\theta)}\right\},$$

set $(k_{t+1}, \theta_{t+1}, x_{t+1}^{(1:N)}) = (l, \theta', z^{(1:N)})$.
Otherwise, set $(k_{t+1}, \theta_{t+1}, x_{t+1}^{(1:N)}) = (k, \theta, x^{(1:N)})$.

A final note regarding the GIMT algorithm is that due to the specific form of the extended target, one can with cost $O(N)$ additionally perform a Gibbs sampling step for k given $(l, \theta, \theta', x^{(1:N)}, z^{(1:N)})$. The probability of each value of K is given by

$$\tilde{\pi}^N(K = i|l, \theta, \theta', x^{(1:N)}, z^{(1:N)}) = \frac{\pi(\theta, x^{(i)})/q(x^{(i)}|\theta)}{\sum_{j=1}^N \pi(\theta, x^{(j)})/q(x^{(j)}|\theta)}.$$

This is due to the significant lack of conditional dependencies between the variables. For general extended targets, attempting such a move will be both more costly, and have a low probability of changing the value of K.

The difference between the algorithms is not only cosmetic. The GIMH algorithm has a cheaper computational complexity, requiring only N simulations from $q(\cdot|\theta')$ and weight evaluations per iteration, whilst the multiple-try variant requires $2N-1$ such simulations and weight evaluations. A primary motivation for this type of pseudo-marginal algorithm is that the terms that appear in the acceptance probability are akin to an importance sampling estimate of $\pi(\theta)$. In particular, for the proposed θ' and $z^{(1:N)}$, we have

$$\mathbb{E}\left[\frac{1}{N}\sum_{j=1}^N \frac{\pi(\theta', z^{(j)})}{q(z^{(j)}|\theta')}\right] = \pi(\theta')$$

since $z^{(j)} \overset{\text{iid}}{\sim} q(\cdot|\theta')$ for $j = 1, \ldots, N$. However, for θ and $x^{(1:N)}$ this does not hold since the $x^{(1:N)}$ are no longer distributed according to q after being accepted at a previous iteration. For the GIMT algorithm, the $x^{(-k)}$ are rejuvenated by q at every step so while the expectation of the average of the weights is not $\pi(\theta')$, for large enough N, the bias contributed by $x^{(k)}$ can be diluted as long as the importance weights are bounded. For the GIMH algorithm, while the $x^{(-k)}$ are distributed according to q at equilibrium, they can take values such that the average of the weights is much higher than $\pi(\theta)$, and these values will persist until a swap move is finally accepted. From a more intuitive perspective, the regeneration of the $x^{(-k)}$ in the GIMT approach means that only the weight of $x^{(k)}$ can cause the chain to get 'stuck'.

4.4.2 Dependent Proposals

In Andrieu and Roberts (2009), a generalization of GIMH is presented that relaxes the requirement that $q(z^{(1:N)}|\theta') = \prod_{i=1}^{N} q(z^{(i)}|\theta')$. This relaxation is straightforward in the framework that has been developed thus far. In addition, it is clear that it is permissible to let q have the form $q(\theta', z^{(1:N)}|\theta, k, x^{(k)})$, and have $\bar{q}$ condition on $(\theta, \theta', k, l, x^{(k)}, z^{(1:N)})$. A generic scheme in this context is given in Algorithm 4.7, noting that $\tilde{\pi}^N(l, k, \theta', \theta, z^{(1:N)}, x^{(1:N)})$ is given in (4.6).

It may be pertinent in many situations to restrict what $\bar{q}$ conditions on in practice, to limit disparity between q and $\bar{q}$. One possible, simple, scheme is to retain the conditional independence of the $z^{(1:N)}$ and $x^{(-k)}$ variables but allow them to condition on $x^{(k)}$ and $z^{(l)}$ respectively and be nonexchangeable. This means we have

$$q(\theta', z^{(1:N)}|\theta, k, x^{(k)}) = g(\theta'|\theta, x^{(k)}) \prod_{i=1}^{N} q_i(z^{(i)}|\theta', x^{(k)})$$

and

$$\bar{q}(x^{(-k)}|\theta, k, x^{(k)}, z^{(l)}) = \prod_{i=1, i\neq k}^{N} q_i(x^{(i)}|\theta, z^{(l)}).$$

Algorithm 4.7 The Generic Pseudo-Marginal Approach

Given $(k_0, \theta_0, x_0^{(k_0)})$, at times $t = 0, 1, \ldots$

1. Set $k = k_t$ and $(\theta, x^{(k)}) = (\theta_t, x_t^{(k_t)})$.
2. Sample $\theta' \sim g(\cdot|\theta, k, x^{(k)})$
3. Sample $z^{(1:N)} \sim q(\cdot|\theta, \theta', k, x^{(k)})$.
4. Sample $L = l$ with probability $\tilde{\pi}^N(l|\theta, \theta', k, x^{(k)}, z^{(1:N)})$
5. Sample $x^{(-k)} \sim \bar{q}(\cdot|\theta, \theta', k, l, x^{(k)}, z^{(1:N)})$.
6. With probability

$$\min\left\{1, \frac{\tilde{\pi}^N(l, k, \theta', \theta, z^{(1:N)}, x^{(1:N)})}{\tilde{\pi}^N(k, l, \theta, \theta', x^{(1:N)}, z^{(1:N)})}\right\}$$

set $(k_{t+1}, \theta_{t+1}, x_{t+1}^{(k_{t+1})}) = (l, \theta', z^{(l)})$.
Otherwise, set $(k_{t+1}, \theta_{t+1}, x_{t+1}^{(k_{t+1})}) = (k, \theta, x^{(k)})$.

If we choose

$$\tilde{\pi}^N(l|x^{(k)}, \theta', z^{(1:N)}) = \frac{\pi(\theta', z^{(l)})/q_l(z^{(l)}|\theta', x^{(k)})}{\sum_{j=1}\pi(\theta', z^{(j)})/q_j(z^{(j)}|\theta', x^{(k)})}$$

then the acceptance ratio is given by

$$\min\left\{1, \frac{g(\theta|\theta', z^{(l)})\sum_{j=1}^N \pi(\theta', z^{(j)})/q_j(z^{(j)}|\theta', x^{(k)})}{g(\theta'|\theta, x^{(k)})\sum_{j=1}^N \pi(\theta, x^{(j)})/q_j(x^{(j)}|\theta, z^{(l)})}\right\}.$$

The numerator divided by N is an unbiased estimate of $\pi(\theta')$ and the denominator divided by N is a consistent estimate of $\pi(\theta)$ as long as the weights are well-behaved, keeping in the spirit of the pseudo-marginal approach. Furthermore, the conditioning on $x^{(k)}$ and $z^{(l)}$ allows the sampler to tune the proposals q_i. If, for example, $\pi(x|\theta)$ is unimodal, has thin tails and the proposal for θ' is such that the mean of $\pi(x|\theta')$ is close to that of $\pi(x|\theta)$, then centering $q_i(\cdot|\theta', x^{(k)})$ at $x^{(k)}$ can allow it to be similarly thin-tailed with well-behaved importance weights when the chain has reached equilibrium. Another key dependency that can be exploited is the ability to condition $g(\cdot|\theta, x^{(k)})$ on $x^{(k)}$. If one can sample according to $\pi(\theta|x)$, then the variable θ' could even be omitted from the extended target. More complex dependencies between variables will be explored in Section 5.2, where particle methods

are described. A description of this algorithm is given in Algorithm 4.8.

Algorithm 4.8 A Dependent Pseudo-Marginal Algorithm

Given $(k_0, \theta_0, x_0^{(k_0)})$, at times $t = 0, 1, \ldots$

1. Set $k = k_t$ and $(\theta, x^{(k)}) = (\theta_t, x_t^{(k_t)})$.
2. Sample $\theta' \sim g(\cdot|\theta, x^{(k)})$
3. For $i \in \{1, \ldots, N\}$, sample $z^{(i)} \sim q_i(\cdot|\theta', x^{(k)})$.
4. Sample $L = l$ with probability $\frac{\pi(\theta', z^{(l)})/q_l(z^{(l)}|\theta', x^{(k)})}{\sum_{j=1}^N \pi(\theta', z^{(j)})/q_j(z^{(j)}|\theta', x^{(k)})}$.
5. For $i \in \{1, \ldots, N\} \setminus \{k\}$, sample $x^{(i)} \sim q_i(\cdot|\theta, z^{(l)})$.
6. With probability

$$\min\left\{1, \frac{g(\theta|\theta', z^{(l)}) \sum_{j=1}^N \pi(\theta', z^{(j)})/q_j(z^{(j)}|\theta', x^{(k)})}{g(\theta'|\theta, x^{(k)}) \sum_{j=1}^N \pi(\theta, x^{(j)})/q_j(x^{(j)}|\theta, z^{(l)})}\right\},$$

set $(k_{t+1}, \theta_{t+1}, x_{t+1}^{(k_{t+1})}) = (l, \theta', z^{(l)})$.
Otherwise, set $(k_{t+1}, \theta_{t+1}, x_{t+1}^{(k_{t+1})}) = (k, \theta, x^{(k)})$.

As noted in Section 4.2.1.3, this methodology is readily applicable without modification when (θ, x) is trans-dimensional. Similarly, this framework can be used to justify the methodology in Dutta and Bhattacharya (2011).

4.5 Adapting N

The use of N auxiliary variables as a generalization of Metropolis-Hastings where $N = 1$ is a promising avenue for future methodological research. However, an obvious concern is how N should be selected. In some cases, this will be a matter of what computational resources are available as increasing N in some simple scenarios may not increase the rate of convergence of the Markov transition kernel such that $\rho_N \leq \rho_1^{2N-1}$, where ρ_j denotes the inverse of the rate of convergence for a geometrically ergodic kernel using j auxiliary variables and one notes that the number of samples required for $j = N$ is $2N - 1$. Parallel computing, of course, allows for a computational cost that is sublinear in N for small enough N and managing this tradeoff will likely be of great interest in the future. If $h(N)$ denotes the compu-

tational time taken for one kernel operation with N auxiliary variables and rate of convergence is the optimality criterion, one would like to find N that minimizes $\rho_N^{1/h(N)}$. Other criteria may be more suitable, such as the asymptotic variance of estimates of expectations of functions of variables distributed according to π.

While the answer of an optimal, fixed N for a given problem and computational resources may be available in some situations, there is also the possibility that N can be incorporated as a random variable in the extended target density. This possibility transforms the question of what the optimal, fixed N is into how to specify the distribution of this random variable, and in particular how to do this in such a way that is both amenable computationally and minimally restrictive. The motivation for this kind of scheme is that less computational power should be needed in some states of the chain than for others. In a model selection framework, for example, where θ encodes models of varying dimensions, one might expect a good value of N to be large when the dimension of the model is large.

We focus first on the adaptation of N in a pseudo-marginal type setting, before considering more standard examples. In this context, the primary motivation for this adaptation is to attempt to adequately approximate the marginal algorithm by increasing the value of N when necessary. It is worth clarifying that this approach falls outside the realm of what are usually called adaptive MCMC methods (see, e.g., Andrieu and Thoms 2008; Roberts and Rosenthal 2009), which typically focus on the global adaptation of parameters of time-inhomogeneous transition kernels in such a way that while a cycle of these kernels does not initially leave π invariant, a large enough cycle is indistinguishable from a cycle of some time-homogeneous kernel with equilibrium distribution π. We are interested here in a characteristically different approach, in that we would like the adaptation to occur in perpetuity and be defined within a transition kernel that by construction admits π as a marginal of its equilibrium distribution. In addition, such local adaptation does not preclude the use of global adaptation of some parameters that define the locally adaptive kernel.

4.5.1 Trial Runs

We are interested in defining an extended target density with N and N' as random variables. With N fixed, it was possible to define the marginal distribution of k, the index of the active particle, as uniform on $\{1, \ldots, N\}$. If N is a random variable, however, this definition cannot be made, at least until N is realized. One way around this problem, is to devote a special variable x^* to the active particle in the target density, whose index k is given a uniform distribution conditionally on N.

Consider the target

$$\begin{aligned}\tilde{\pi}(N, N', k, l, \theta, \theta', x^*, z^*, x^{(1:N)}, z^{(1:N')}) = \\ \pi(\theta, x^*) g(\theta'|\theta, x^*) q(N'|\theta', \theta, x^*) q(z^{(1:N')}|N', \theta', \theta, x^*) \tilde{\pi}(l|N', \theta', \theta, x^*, z^{(1:N')}) \\ \times \delta_{z^{(l)}}(z^*) q(N|\theta, \theta', z^*) \frac{1}{N} \delta_{x^*}(x^{(k)}) \bar{q}(x^{(-k)}|N, N', \theta, \theta', k, l, x^{(k)}, z^{(1:N')})\end{aligned} \tag{4.7}$$

where, in particular, the form of the proposal for N and N' is identical.

The acceptance probability of the swap move $(N, N', k, l, \theta, \theta', x^*, z^*, x^{(1:N)}, z^{(1:N')})$ $\rightarrow (N', N, l, k, \theta', \theta, z^*, x^*, z^{(1:N')}, x^{(1:N)})$ does not include the terms $q(N'|\theta', \theta, x^*)$ or $q(N|\theta, \theta', z^*)$ as they cancel in the acceptance ratio. For example, the dependent pseudo-marginal scheme of Algorithm 4.8 augmented with N and N' proposed conditionally in this fashion has the acceptance probability of the swap move given by

$$\min\left\{1, \frac{g(\theta|\theta', z^*) \frac{1}{N'} \sum_{j=1}^{N'} \pi(\theta', z^{(j)}) / q_j(z^{(j)}|\theta', x^*)}{g(\theta'|\theta, x^*) \frac{1}{N} \sum_{j=1}^{N} \pi(\theta, x^{(j)}) / q_j(x^{(j)}|\theta, z^*)}\right\}.$$

The terms $1/N$ and $1/N'$ divide the sum of N and N' weights, respectively, with all other terms remaining as before.

The importance of the cancellation of the probability mass functions for N and N' in the acceptance ratio is crucial in this setup because typically one wants to propose values of N and N' conditionally on the values of additional auxiliary random variables associated with θ and θ' respectively, whose purpose is ostensibly to garner information about $\pi(\theta)$ and $\pi(\theta')$ in order to inform the choice of N and N'. If these random variables were to be included in the extended target, their densities

would also cancel in the acceptance probability as they appear in the same positions in both the forwards and backwards proposals. The difficulty with allowing the proposal for N to condition on x^* arises because for the cancellation to occur as before, one would need the proposal for N' to depend on $z^* = z^{(l)}$, but l is defined conditionally upon N'.

The ability to condition N' on θ' and x^* gives some ability for the sampler to be adaptive. However, the inability to condition N on x^* and retain the cancellation of this probability is limiting to a certain extent. For example, one could simulate 'trial' particles $\tilde{z}_1, \tilde{z}_2, \ldots$ until the average of the weights associated with these trial satisfies some kind of stability criteria as a mechanism for selecting N'. The restriction in this scheme is more apparent when considering how N is proposed. In this case, N depends on z^* but not x^*. This indicates that N cannot be adapted to, in particular, the size of the weight associated with x^*, which might be desirable in order to limit its influence on the acceptance probability of a move.

It is also possible to have $N = N'$ by defining a distribution for $N = N'$ conditional on θ and θ', but not x^* and z^*. We will see in the next section that it is possible to define distributions for N and N' that are considerably more flexible.

4.5.2 A General Approach

The methodology above is not ideal for many reasons. First, the lack of dependence on x^* means it may not adapt 'well' to the actual state of the chain. Second, we envisage it being used mainly via trial simulations of random variables like the x's and z's, i.e. using similar types of proposals, with these samples then thrown away, which can be wasteful computationally.

To overcome this, we introduce auxiliary variables $u^{(1:N)}$ and $v^{(1:N')}$ and a function $\psi_{(N,k)}$ with $k \in \{1, \ldots, N\}$, that generates a permutation of its input, a vector of length N. We denote by $\psi^{-1}_{(N,k)}$ the inverse of $\psi_{(N,k)}$, which generates the permutation of its input satisfying

$$\psi^{-1}_{(N,k)}(\psi_{(N,k)}(y^{(1:N)})) = y^{(1:N)}.$$

Additionally, we require that if $u^{(1:N)} = \psi_{N,k}(x^{(1:N)})$, then $u^{(1)} = x^{(k)}$.

We can now define an extended target density of the form

$$\begin{aligned}\tilde{\pi}(N, N', k, l, \theta, \theta', u^{(1:N)}, v^{(1:N')}, x^{(1:N)}, z^{(1:N')}) = \\ \pi(\theta, u^{(1)}) g(\theta'|\theta, u^{(1)}) q(N', z^{(1:N')}|\theta, \theta', u^{(1)}) \tilde{\pi}(l|N', \theta, \theta', u^{(1)}, z^{(1:N')}) \\ \times \bar{q}(N, u^{(2:N)}|N', \theta, \theta', l, u^{(1)}, z^{(1:N')}) \tilde{\pi}(k|N, N', \theta, \theta', l, u^{(1:N)}, z^{(1:N')}) \\ \times \delta_{\psi_{(N',l)}(z^{(1:N')})}(v^{(1:N')}) \delta_{\psi^{-1}_{(N,k)}(u^{(1:N)})}(x^{(1:N)}).\end{aligned} \tag{4.8}$$

This target density may appear daunting at first and some of the conditional dependencies will be superfluous in most applications. An important property of this extended target density is that $u^{(1)}$ is marginally distributed according to π, and in particular, independent of N. This is necessary, as before, since K can usually only be defined conditionally upon N, which is now a random variable. The functions ψ and ψ^{-1} are used to map vectors of random variables to reordered vectors of those random variables in order that one can propose one of the N' samples $v^{(1)} = z^{(l)}$ to replace $u^{(1)}$ in a swap move without specifying a fixed value of N'. The need to include the function ψ may seem unnecessary at first, but it is a crucial component of the methodology since it allows for the specification of a backward proposal for the variables $z^{(1:N')}$ and a forward proposal for the variables $u^{(1:N)}$ to allow for the computation of the acceptance probability in a Metropolis swap move.

An example of a possible function $\psi_{N,k}$ is

$$\psi_{N,k}(y^{(1:N)}) = (y^{(k)}, y^{(1:k-1)}, y^{(k+1:N)})$$

where the corresponding $\psi^{-1}_{N,k}$ is defined as

$$\psi^{-1}_{N,k}(y^{(1:N)}) = (y^{(2:k-1)}, y^{(1)}, y^{(k+1:N)}).$$

Alternatively, an example with $\psi_{N,k} = \psi^{-1}_{N,k}$ is

$$\psi_{N,k}(y^{(1:N)}) = (y^{(k)}, y^{(2:k-1)}, y^{(1)}, y^{(k+1:N)}). \tag{4.9}$$

Due to the simple nature of switching the location of y_k and y_1 in this second choice,

it will be the default choice in the following material, but this function will generally be chosen in relation to the nature of the proposal densities in the extended target density.

As with previous extended target densities, reading (4.8) from left to right provides a suggested algorithm for performing a cycle of a Gibbs step and a Metropolis swap. In this case, it can be useful to distinguish between a density where N' and N are distinct random variables and the case $N = N'$, since in the latter case the proposal density for $N = N'$ will likely be defined jointly with both $u^{(2:N)}$ and $z^{(1:N)}$. In this latter case it can be helpful to write an alternative extended target as

$$\begin{aligned}\tilde{\pi}(N, k, l, \theta, \theta', u^{(1:N)}, v^{(1:N')}, x^{(1:N)}, z^{(1:N')}) = \\ \pi(\theta, u^{(1)}) g(\theta'|\theta, u^{(1)}) q(N, u^{(2:N)}, z^{(1:N)}|\theta, \theta', u^{(1)}) \tilde{\pi}(l|N, \theta, \theta', u^{(1:N)}, z^{(1:N)}) \\ \times \tilde{\pi}(k|N, \theta, \theta', l, u^{(1:N)}, z^{(1:N)}) \delta_{\psi_{(N,l)}(z^{(1:N)})}(v^{(1:N)}) \delta_{\psi^{-1}_{(N,k)}(u^{(1:N)})}(x^{(1:N)}).\end{aligned} \tag{4.10}$$

Two possible schemes are given in Algorithms 4.9 and 4.10, whose respective extended target densities are defined by (4.8) and (4.10). A further note is that these are but two decompositions of the extended target, and each suggest different general algorithms.

In many situations, q will be conditional only on θ' and $u^{(1)}$ while $\bar{q}$ will be conditional only on θ, $u^{(1)}$ and $z^{(l)}$. A simple, possible form for q is

$$\begin{aligned}q(N', z^{(1:N')}|\theta', u^{(1)}) = \beta_{N'}(z^{(1:N')}; \theta', u^{(1)}) q(z^{(N')}|\theta', u^{(1)}, z^{(1:N'-1)}) \\ \times \prod_{j=1}^{N'-1} [1 - \beta_j(z^{(1:j)}; \theta', u^{(1)})] q(z^{(j)}|\theta', u^{(1)}, z^{(1:j-1)})\end{aligned}$$

while a possible form for $\bar{q}$ is

$$\begin{aligned}\bar{q}(N, u^{(2:N)}|\theta, u^{(1)}, z^{(l)}) = \bar{\beta}_N(u^{(1:N)}; \theta, z^{(l)}) \bar{q}(u^{(N)}|\theta, z^{(l)}, u^{(1:N-1)}) \\ \times \prod_{j=1}^{N-1} [1 - \bar{\beta}_j(u^{(1:j)}; \theta, z^{(l)})] \bar{q}(u^{(j)}|\theta, z^{(l)}, u^{(1:j-1)})\end{aligned}$$

with the understanding that $\bar{q}(u^{(1)}|\theta, z^{(l)}, u^{(1)})$ is degenerate. The functions β_j and $\bar{\beta}_j$ encode the probability of stopping after obtaining j samples in the forward and

Algorithm 4.9 An Adaptive Algorithm with Distinct N, N'

Given $(\theta_0, u^{(1)})$, at times $t = 0, 1, \ldots$

1. Set $(\theta, u^{(1)}) = (\theta_t, u_t^{(1)})$.
2. Sample $\theta' \sim g(\cdot|\theta, u^{(1)})$.
3. Sample $N', z^{(1:N')} \sim q(\cdot|\theta, \theta', u^{(1)})$.
4. Sample $L = l$ with probability $\tilde{\pi}(l|N', \theta, \theta', u^{(1)}, z^{(1:N')})$.
5. Sample $N, u^{(2:N)} \sim \bar{q}(\cdot|N', \theta, \theta', l, u^{(1)}, z^{(1:N')})$.
6. Sample $K = k$ with probability $\tilde{\pi}(k|N, N', \theta, \theta', l, u^{(1:N)}, z^{(1:N')})$.
7. Set $v^{(1:N')} = \psi_{(N',l)}(z^{(1:N')})$ and $x^{(1:N)} = \psi^{-1}_{(N,k)}(u^{(1:N)})$.
8. With probability

$$\min\left\{1, \frac{\tilde{\pi}(N', N, l, k, \theta', \theta, v^{(1:N')}, u^{(1:N)}, z^{(1:N')}, x^{(1:N)})}{\tilde{\pi}(N, N', k, l, \theta, \theta', u^{(1:N)}, v^{(1:N')}, x^{(1:N)}, z^{(1:N')})}\right\},$$

set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta', v^{(1)})$.
Otherwise, set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta, u^{(1)})$.

Algorithm 4.10 An Adaptive Algorithm with Shared N

Given $(\theta_0, u^{(1)})$, at times $t = 0, 1, \ldots$

1. Set $(\theta, u^{(1)}) = (\theta_t, u_t^{(1)})$.
2. Sample $\theta' \sim g(\cdot|\theta, u^{(1)})$.
3. Sample $N, u^{(2:N)}, z^{(1:N)} \sim q(\cdot|\theta, \theta', u^{(1)})$.
4. Sample $L = l$ with probability $\tilde{\pi}(l|N, \theta, \theta', u^{(1:N)}, z^{(1:N)})$.
5. Sample $K = k$ with probability $\tilde{\pi}(k|N, \theta, \theta', l, u^{(1:N)}, z^{(1:N)})$.
6. Set $v^{(1:N)} = \psi_{(N,l)}(z^{(1:N)})$ and $x^{(1:N)} = \psi^{-1}_{(N,k)}(u^{(1:N)})$.
7. With probability

$$\min\left\{1, \frac{\tilde{\pi}(N, l, k, \theta', \theta, v^{(1:N)}, u^{(1:N)}, z^{(1:N)}, x^{(1:N)})}{\tilde{\pi}(N, k, l, \theta, \theta', u^{(1:N)}, v^{(1:N)}, x^{(1:N)}, z^{(1:N)})}\right\},$$

set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta', v^{(1)})$.
Otherwise, set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta, u^{(1)})$.

backward proposals respectively, and provide an opportunity for sampling to continue until some criteria is satisfied.

It is helpful to consider the form of the acceptance probability when choosing both q and $\bar{q}$ because discrepancies between these densities can have a large impact on the acceptance probability. In particular, the term

$$\rho = \frac{\bar{q}(N', v^{(2:N')}|\theta', v^{(1)}, u^{(1)})q(N, x^{(1:N)}|\theta, v^{(1)})}{\bar{q}(N, u^{(2:N)}|\theta, u^{(1)}, v^{(1)})q(N', z^{(1:N')}|\theta', u^{(1)})} \tag{4.11}$$

appears in this probability. One can decompose this term as $\rho = \rho_1\rho_2$ where

$$\rho_1 = \frac{\prod_{j=1}^{N} q(x^{(j)}|\theta, v^{(1)}, x^{(1:j-1)})}{\prod_{j=2}^{N} \bar{q}(u^{(j)}|\theta, v^{(1)}, u^{(1:j-1)})} \times \frac{\prod_{j=2}^{N'} \bar{q}(v^{(j)}|\theta', u^{(1)}, v^{(1:j-1)})}{\prod_{j=1}^{N'} q(z^{(j)}|\theta', u^{(1)}, z^{(1:j-1)})} \tag{4.12}$$

and

$$\rho_2 = \frac{\beta_N(x^{(1:N)}; \theta, v^{(1)}) \prod_{j=1}^{N-1}[1 - \beta_j(x^{(1:j)}; \theta, v^{(1)})]}{\bar{\beta}_N(u^{(1:N)}; \theta, v^{(1)}) \prod_{j=1}^{N-1}[1 - \bar{\beta}_j(u^{(1:j)}; \theta, v^{(1)})]} \tag{4.13}$$

$$\times \frac{\bar{\beta}_{N'}(v^{(1:N')}; \theta', u^{(1)}) \prod_{j=1}^{N'-1}[1 - \bar{\beta}_j(v^{(1:j)}; \theta', u^{(1)})]}{\beta_{N'}(z^{(1:N')}; \theta', u^{(1)}) \prod_{j=1}^{N'-1}[1 - \beta_j(z^{(1:j)}; \theta', u^{(1)})]}. \tag{4.14}$$

For example, when $\bar{q} = q$ and ψ is defined by (4.9), ρ_1 encodes the discrepancy between the proposals and ρ_2 encodes the discrepancy between N and N' when the positions $(1, l)$ and $(1, k)$ have been switched for the forwards $(z^{(1:N')}, v^{(1:N')})$ and backwards proposals $(x^{(1:N)}, u^{(1:N)})$ respectively.

In the special case where we have

$$\begin{aligned} q(N', z^{(1:N')}|\theta', u^{(1)}) = \beta_{N'}(z^{(1:N')}; \theta', u^{(1)})q(z^{(N')}|\theta', u^{(1)}) \\ \times \prod_{j=1}^{N'-1}[1 - \beta_j(z^{(1:j)}; \theta', u^{(1)})]q(z^{(j)}|\theta', u^{(1)}) \end{aligned}$$

and

$$\bar{q}(N, u^{(2:N)}|\theta, u^{(1)}, v^{(1)}) = \bar{\beta}_N(u^{(1:N)}; \theta, v^{(1)})q(u^{(N)}|\theta, v^{(1)}) \times \prod_{j=2}^{N-1}[1 - \bar{\beta}_j(u^{(1:j)}; \theta, v^{(1)})]q(u^{(j)}|\theta, v^{(1)}) \times [1 - \bar{\beta}_1(u^{(1)}; \theta, v^{(1)})],$$

then (4.12) can be simplified to

$$\rho_1 = \frac{q(u^{(1)}|\theta, v^{(1)})}{q(v^{(1)}|\theta', u^{(1)})}$$

where $v^{(1:N')} = \psi_{(N',l)}(z^{(1:N')})$ and $x^{(1:N)} = \psi^{-1}_{(N,k)}(u^{(1:N)})$ and ρ_2 is unchanged.

Possible options for the β_j's and $\bar{\beta}_j$'s are numerous, and it seems reasonable for these to be identical in some situations. Some possibilities include $\bar{\beta}_j = \beta_j = 0$ if $j < M$ for some M and if $j \geq M$ depending, e.g., on the average and variance of its arguments or the effective sample size. For blocks of variables at a time, one can have $\beta_j = 0$ if j is not in some set of permissible values of N, or stopping times. In the case of identifiable rare samples, as will be demonstrated in the examples of Section 4.6.1, one can make $\bar{\beta}_j$ and β_j depend on the observation of the number of such events.

4.5.2.1 An Adaptive Pseudo-Marginal Algorithm

An example of a pseudo-marginal algorithm with distinct, adaptive N and N' is given in Algorithm 4.11. In this scheme, we assume that $q(z^{(j)}|\theta', u^{(1)}, z^{(1:j-1)}) = q(z^{(j)}|\theta', u^{(1)})$ and $\bar{q}(u^{(j)}|\theta, v^{(1)}, u^{(1:j-1)}) = q(u^{(j)}|\theta, v^{(1)})$.

4.5.2.2 A Special Case

It is interesting to note that in some circumstances it is possible to specify β, $\bar{\beta}$ and the conditional distributions of L and K in such a way that the correction factor ρ_2

Algorithm 4.11 An Adaptive Dependent Pseudo-Marginal Algorithm

Given $(\theta_0, u_0^{(1)})$, at times $t = 0, 1, \ldots$

1. Set $(\theta, u^{(1)}) = (\theta_t, u_t^{(1)})$.
2. Sample $\theta' \sim g(\cdot|\theta, u^{(1)})$.
3. For $i = 1, \ldots$
   - Sample $z^{(i)} \sim q(\cdot|\theta', u^{(1)})$
   - With probability $\beta_i(z^{(1:i)}; \theta', u^{(1)})$, set $N' = i$ and go to step 4.
4. Sample $L = l$ with probability $\frac{\pi(\theta', z^{(l)})/q(z^{(l)}|\theta', u^{(1)})}{\sum_{j=1}^{N'} \pi(\theta', z^{(j)})/q(z^{(j)}|\theta', u^{(1)})}$.
5. For $i = 2, \ldots$
   - Sample $u^{(i)} \sim q(\cdot|\theta, z^{(l)})$
   - With probability $\bar{\beta}_i(u^{(1:i)}; \theta, z^{(l)})$, set $N = i$ and go to step 6.
6. Sample $K = k$ uniformly on $\{1, \ldots, N\}$.
7. Set $v^{(1:N')} = \psi_{(N',l)}(z^{(1:N')})$ and $x^{(1:N)} = \psi_{(N,k)}^{-1}(u^{(1:N)})$ and compute

$$\rho_2 = \frac{\beta_N(x^{(1:N)}; \theta, v^{(1)}) \prod_{j=1}^{N-1}[1 - \beta_j(x^{(1:j)}; \theta, v^{(1)})]}{\bar{\beta}_N(u^{(1:N)}; \theta, v^{(1)}) \prod_{j=1}^{N-1}[1 - \bar{\beta}_j(u^{(1:j)}; \theta, v^{(1)})]} \times \frac{\bar{\beta}_{N'}(v^{(1:N')}; \theta', u^{(1)}) \prod_{j=1}^{N'-1}[1 - \bar{\beta}_j(v^{(1:j)}; \theta', u^{(1)})]}{\beta_{N'}(z^{(1:N')}; \theta', u^{(1)}) \prod_{j=1}^{N'-1}[1 - \beta_j(z^{(1:j)}; \theta', u^{(1)})]}.$$

8. With probability

$$\min\left\{1, \rho_2 \frac{g(\theta|\theta', v^{(1)}) \frac{1}{N'} \sum_{j=1}^{N'} \pi(\theta', z^{(j)})/q(z^{(j)}|\theta', u^{(1)})}{g(\theta'|\theta, u^{(1)}) \frac{1}{N} \sum_{j=1}^{N} \pi(\theta, x^{(j)})/q(x^{(j)}|\theta, v^{(1)})}\right\},$$

set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta', v^{(1)})$.
Otherwise, set $(\theta_{t+1}, x_{t+1}^{(k_{t+1})}) = (\theta, u^{(1)})$.

is always 1.

Consider the case where $\bar{\beta}_j = \beta_j$ is symmetric, i.e. invariant to the order of its inputs. This implies that $\bar{\beta}_j(u^{(1:j)}; \theta, v^{(1)}) = \beta_j(x^{(1:j)}; \theta, v^{(1)})$ for all $j \geq k$ when ψ is given by (4.9). In order to ensure that additionally we have $\bar{\beta}_j(u^{(1:j)}; \theta, v^{(1)}) = \beta_j(x^{(1:j)}; \theta, v^{(1)})$ for $j < k$, let us consider the restriction $\beta_j(x^{(1:j)}; \theta, v^{(1)}) = \mathbf{1}_{[c,\infty)}(h_j(w^{(1:j)}))$ where $w^{(i)}$ is a function of $(x^{(i)}, \theta, v^{(1)})$ and c is a function of $(\theta, v^{(1)})$. If h_j is symmetric and additionally satisfies

$$h_{j+1}(w^{(1:j+1)}) \geq h_j(w^{(1:j)}) \tag{4.15}$$

for any possible value of $w^{(j+1)}$, then one will have $\bar{\beta}_j(u^{(1:j)}; \theta, v^{(1)}) = \beta_j(x^{(1:j)}; \theta, v^{(1)})$ for all $j \in \{1, \ldots, N\}$ as long as $k \neq N$. A simple example of an h_j satisfying these requirements is $h_j(w^{(1:j)}) = \sum_{i=1}^{j} w^{(i)}$, when each $w^{(i)} \geq 0$.

To see this, we have $\bar{\beta}_j(u^{(1:j)}; \theta, v^{(1)}) = \mathbf{1}_{[c,\infty)}(h_j(\bar{w}^{(1:j)}))$ and $\beta_j(x^{(1:j)}; \theta, v^{(1)}) = \mathbf{1}_{[c,\infty)}(h_j(w^{(1:j)}))$, where $\bar{w}^{(1:N)} = \psi_{N,k}^{-1}(w^{(1:N)})$. We have also that $h_N(w^{(1:N)}) \geq c$ but $h_{N-1}(w^{(1:N-1)}) < c$. This implies $h_N(\bar{w}^{(1:N)}) = h_N(w^{(1:N)}) \geq c$ but that for all $j < N$, $h_j(\bar{w}^{(1:j)}) \leq h_{N-1}(\bar{w}^{(1:N-1)}) = h_{N-1}(w^{(1:N-1)}) < c$, where the last equality holds as long as $k \neq N$. It immediately follows that $\bar{\beta}_j(u^{(1:j)}; \theta, v^{(1)}) = \beta_j(x^{(1:j)}; \theta, v^{(1)}) = 0$ for all $j \in \{1, \ldots, N-1\}$ as long as $k \neq N$ and that $\bar{\beta}_N(u^{(1:N)}; \theta, v^{(1)}) = \beta_N(x^{(1:N)}; \theta, v^{(1)}) = 1$. Analogous reasoning holds for showing that $\bar{\beta}_j(v^{(1:j)}; \theta', x^*) = \beta_j(z^{(1:j)}; \theta', x^*)$ for all $j \in \{1, \ldots, N'\}$ as long as $l \neq N'$.

Therefore, if we additionally have $\tilde{\pi}(L = N'|N', \theta, \theta', u^{(1)}, z^{(1:N')}) = 0$ and $\tilde{\pi}(K = N|N, N', \theta, \theta', l, u^{(1:N)}, z^{(1:N')}) = 0$ in (4.8), one will always have $\rho_2 = 1$. In this case, some care should be taken to ensure that the resulting kernel is irreducible. Indeed, having the conditional probability of $L = N'$ be 0 indicates that proposed values of $v^{(1)}$ cannot take values in $\{z : w(z, \theta', u^{(1)}) \geq c(\theta', u^{(1)})\}$, even if $\pi(\theta', z) > 0$ for some z in this set. In order to circumvent this issue, one could have $c(\theta', u^{(1)}) > \sup_z w(z, \theta', u^{(1)})$ or use a scheme in which $\bar{\beta}_j = \beta_j$, $\beta_j(x^{(1:j)}; \theta, v^{(1)}) = \mathbf{1}_{\{M, M+1, \ldots\}}(j)\mathbf{1}_{[c,\infty)}(h_j(w^{(1:j)}))$ and we have $\tilde{\pi}(L = N'|N', \theta, \theta', u^{(1)}, z^{(1:N')}) = 0$ if $N' \neq M$ and similarly $\tilde{\pi}(K = N|N, N', \theta, \theta', l, u^{(1:N)}, z^{(1:N')}) = 0$ if $N \neq M$. This essentially involves a minimum value of N and N' given by M. M can also be a function of $(\theta, v^{(1)})$ or $(\theta', u^{(1)})$, respectively, for the proposals of $(N, x^{(1:N)})$ or

$(N', z^{(1:N')})$.

Not all functions h_j we might want to use satisfy (4.15), even if they are symmetric. An example is the effective sample size.

The ability for us to ensure $\rho_2 = 1$ always does not imply that such schemes are necessarily better in practice. Indeed, one could alternatively attempt to use schemes that ensure ρ_2 is close to 1 or equal to 1 with high probability.

4.5.3 Standard Setting

The adaptation of N can serve a more general purpose, which may be clearer outside of the pseudo-marginal setting. In particular, larger values of N are not necessarily associated with closer approximations of a marginal density whose corresponding marginal algorithm one would like to emulate. We can write an extended target density in this case as

$$\begin{aligned}\tilde{\pi}(N, N', k, l, u^{(1:N)}, v^{(1:N')}, x^{(1:N)}, z^{(1:N')}) = \\ \pi(u^{(1)})q(N', z^{(1:N')}|u^{(1)})\tilde{\pi}(l|N', u^{(1)}, z^{(1:N')})\bar{q}(N, u^{(2:N)}|N', l, u^{(1)}, z^{(1:N')}) \\ \times \tilde{\pi}(k|N, N', l, u^{(1:N)}, z^{(1:N')})\delta_{\psi_{(N',l)}(z^{(1:N')})}(v^{(1:N')})\delta_{\psi^{-1}_{(N,k)}(u^{(1:N)})}(x^{(1:N)}).\end{aligned} \tag{4.16}$$

There are many possibilities that arise from the definition of N and N' as random variables. To give but one example of the versatility of the approach, we show how it can be used to derive the delayed rejection methodology introduced in Tierney and Mira (1999) and Mira (2001).

4.5.3.1 Delayed Rejection

Let $\tilde{y}^{(1:j)} = (y^{(j)}, \ldots, y^{(1)})$. Then the delayed rejection algorithm corresponds to the choices

$$q(N', z^{(1:N')}|u^{(1)}) = \delta_{u^{(1)}}(z^{(1)})\beta_{N'}(z^{(1:N')}) \left[\prod_{i=2}^{N'} q_i(z^{(i)}|z^{(1:i-1)})\right] \prod_{i=1}^{N'-1} [1 - \beta_i(z^{(1:i)})],$$

$\tilde{\pi}(l|N') = \delta_{N'}(l)$, $\bar{q}(N, u^{(2:N)}|N', z^{(1:N')}) = \delta_{N'}(N)\delta_{z^{(2:N)}}(u^{(2:N)})$, $\tilde{\pi}(k|N) = \delta_N(k)$ and $\psi_{N,N}(y^{(1:N)}) = \tilde{y}^{(1:N)}$, and for $j \geq 2$

$$\beta_j(z^{(1:j)}) = \min\left\{1, \frac{\pi(\tilde{z}^{(1)})[\prod_{i=2}^{j} q_i(\tilde{z}^{(i)}|\tilde{z}^{(1:i-1)})] \prod_{i=1}^{j-1}[1 - \beta_i(\tilde{z}^{(1:i)})]}{\pi(z^{(1)})[\prod_{i=2}^{j} q_i(z^{(i)}|z^{(1:i-1)})] \prod_{i=1}^{j-1}[1 - \beta_i(z^{(1:i)})]}\right\}, \tag{4.17}$$

while for $j = 1$, $\beta_1(z^{(1)}) = 0$.

Since $N' = N$ in all cases for this algorithm, we refer to both of these as N. Also, it should be clear that in this case $k = l = N$, $z^{(1:N)} = u^{(1:N)}$ and $v^{(1:N)} = x^{(1:N)} = \tilde{z}^{(1:N)}$. The acceptance probability of the move $(N, k, l, u^{(1:N)}, v^{(1:N)}, x^{(1:N)}, z^{(1:N)}) \to (N, k, l, u^{(1:N)}, v^{(1:N)}, x^{(1:N)}, z^{(1:N)})$ is given by

$$\min\left\{1, \frac{\pi(\tilde{z}^{(1)})\beta_N(\tilde{z}^{(1:N)})[\prod_{i=2}^{j} q_i(\tilde{z}^{(i)}|\tilde{z}^{(1:i-1)})] \prod_{i=1}^{j-1}[1 - \beta_i(\tilde{z}^{(1:i)})]}{\pi(z^{(1)})\beta_N(z^{(1:N)})[\prod_{i=2}^{j} q_i(z^{(i)}|z^{(1:i-1)})] \prod_{i=1}^{j-1}[1 - \beta_i(z^{(1:i)})]}\right\}. \tag{4.18}$$

Note that regardless of the form of β_j for $j < N$, if

$$\begin{aligned}\beta_N(z^{(1:N)}) &= \min\left\{1, \frac{\pi(\tilde{z}^{(1)})[\prod_{i=2}^{N} q_i(\tilde{z}^{(i)}|\tilde{z}^{(1:i-1)})] \prod_{i=1}^{N-1}[1 - \beta_i(\tilde{z}^{(1:i)})]}{\pi(z^{(1)})[\prod_{i=2}^{N} q_i(z^{(i)}|z^{(1:i-1)})] \prod_{i=1}^{N-1}[1 - \beta_i(z^{(1:i)})]}\right\} \\ &= \min\{1, c\},\end{aligned}$$

then (4.18) reduces to

$$\min\left\{1, c\frac{\min\{1, 1/c\}}{\min\{1, c\}}\right\} = 1, \text{ for any } c > 0.$$

Similarly, if $\beta_j(z^{(1:j)}) = 1$ for $j = M$ independent of the values of $z^{(1:j)}$, i.e. there is a maximum value of N given by M, then (4.18) reduces to $\beta_N(z^{(1:N)})$ given by (4.17)

when $N = M$. In Mira (2001), due to the rapidly increasing computational expense as N increases, and the observation that (4.17) does not necessarily define a finite stopping time, it is suggested that one stop at time j with a probability independent of $z^{(1:j)}$, p_j. This scheme can be viewed as a mixture of MCMC kernels where M is chosen randomly. Alternatively, one can include this probability of stopping via a modified $\beta'_j(z^{(1:j)}) = \beta_j(z^{(1:j)})(1 - p_j) + p_j$. The acceptance probability associated with this choice is

$$\min\left\{1, c\frac{\min\{1, 1/c\}(1 - p_N) + p_N}{\min\{1, c\}(1 - p_N) + p_N}\right\} \geq \min\{1, c\}$$

where c is defined as above. The inequality suggests that using this mixture of proposals instead of a mixture of kernels, which comes at essentially no extra cost, will provide lower asymptotic variance by the arguments in (Tierney 1998, Section 3). This can alternatively be seen as a special case of the result in (Tierney 1998, Section 4).

An observation is that the distributions of k, l and $u^{(2:N)}$ are degenerate in this formulation, and indeed that they can be defined more generally. Green and Mira (2001) details a scheme for allowing $z^{(1:N)} \neq u^{(1:N)}$ but our framework provides a more general approach, allowing also for the specification of non-degenerate distributions for k and l. Furthermore, it is not necessary that the condition for stopping be equivalent to the probability of acceptance once stopped since the acceptance probability associated with any set of computable β_j's can be computed explicitly.

4.6 Applications

This section contains discussion of some applications of the developed methodology to pseudo-marginal type problems. The application to MCMC kernels for approximate Bayesian computation provides a detailed discussion on the effect of the adaptation of N in this setting, where the highly restrictive scenario allows for an explicit analytic form for the expected acceptance probability of a move $\theta \to \theta'$ for one of the adaptive algorithms. The application to Bayesian variable selection uses a toy example whose purpose is to highlight the benefits of proposed methodology.

4.6.1 Approximate Bayesian Computation

Approximate Bayesian computation (ABC), mentioned briefly in Section 1.1.2.4, is a class of algorithms whose popularity has grown in recent years since its introduction in the population genetics literature in Tavare et al. (1997), Fu and Li (1997) and Pritchard et al. (1999). The characteristic feature of this methodology is that one cannot evaluate the likelihood function $f(y|\theta)$ but one can simulate according to it. If $y, z \in \mathcal{X}$ and $\mathcal{X}$ is discrete one can define $f(y, z|\theta) = \mathbf{1}_z(y)f(z|\theta)$ which has the identity

$$\sum_{z\in\mathcal{X}} f(y, z|\theta) = \sum_{z\in\mathcal{X}} \mathbf{1}_z(y)f(z|\theta) = f(y|\theta).$$

One is interested in inference on $\theta \in \Theta$, whose posterior density is given by

$$\pi(\theta) = p(\theta|y) = \frac{f(y|\theta)p(\theta)}{\int_\Theta f(y|\theta)p(\theta)d\theta}$$

but the inability to compute $f(y|\theta)$ makes direct computation of this density impossible. However, the auxiliary variable z provides a workaround since we have

$$\pi(\theta, z) = p(\theta, z|y) = \frac{f(y, z|\theta)p(\theta)}{\int_\Theta f(y|\theta)p(\theta)d\theta}.$$

This allows for the use of Monte Carlo schemes that propose values of z in importance sampling and rejection sampling schemes. For example, a valid rejection sampling scheme for $\pi(\theta, z)$ consists of sampling $\theta \sim g(\cdot), z \sim f(\cdot|\theta)$ and accepting (θ, z) with probability $\frac{p(\theta)\mathbf{1}_z(y)}{Cg(\theta)}$ where $C \geq \sup_\vartheta \frac{p(\vartheta)}{g(\vartheta)}$. Alternatively, an importance sampling estimate can be obtained by weighting each sample (θ, z) by $\frac{p(\theta)\mathbf{1}_z(y)}{g(\theta)}$.

For continuous densities f, the above approach is not useful. In this case, a standard approach is to define an approximation to $f(y|\theta)$. One first defines

$$f_\epsilon(y, z|\theta) = \frac{\mathbf{1}_{B_\epsilon(z)}(y)}{Z_\epsilon} f(z|\theta)$$

where $B_\epsilon(z)$ denotes a ball of radius ϵ around z under some metric and $Z_\epsilon = \int_\mathcal{X} \mathbf{1}_{B_\epsilon(z)}(y)dy$ is invariant to the value of z. Note that because B is a metric ball,

we have $\mathbf{1}_{B_\epsilon(z)}(y) = \mathbf{1}_{B_\epsilon(y)}(z)$. The marginal of this joint density

$$f_\epsilon(y|\theta) = \int_{\mathcal{X}} \frac{\mathbf{1}_{B_\epsilon(z)}(y)}{Z_\epsilon} f(z|\theta)dz$$

is the approximation to $f(y|\theta)$ used in ABC. In many cases one additionally defines a summary statistic $S(\cdot)$ and does inference conditional only on $S(y)$. In this setting, the approximation to the likelihood is given by

$$f_\epsilon(y|\theta) = \int_{\mathcal{X}} \frac{\mathbf{1}_{B_\epsilon(S(z))}(S(y))}{Z_\epsilon} f(z|\theta)dz$$

where $Z_\epsilon = \int_{\mathcal{X}} \mathbf{1}_{B_\epsilon(S(z))}(S(y))dS(y)$. This amounts to summarizing both y and z using summary statistics defined by S and allowing these summary statistics to differ by at most ϵ under some metric. This approach can also be useful when f denotes a probability mass function but y is high-dimensional so that $f(y|\theta)$ is prohibitively small. For the remainder of this section, explicit reference to S is avoided, but the methodology proposed can be used when such summary statistics are used.

We will define the approximate posterior used when using the approximate likelihood $f_\epsilon(y|\theta)$ as $\hat{\pi}_\epsilon(\theta|y)$, defined by

$$\hat{\pi}_\epsilon(\theta|y) = \frac{f_\epsilon(y|\theta)p(\theta)}{\int_\Theta f_\epsilon(y|\theta)p(\theta)d\theta}$$

which arises as the marginal of $\hat{\pi}_\epsilon(\theta, z|y) = f_\epsilon(y, z|\theta)p(\theta)/\int_\Theta f_\epsilon(y|\theta)p(\theta)d\theta$. It is useful for clarity to make the following definition

$$h_\epsilon(\theta) = \int \mathbf{1}_{B_\epsilon(z)}(y)f(z|\theta)dz = Z_\epsilon f_\epsilon(y|\theta)$$

which denotes the probability of a "hit" for a given ϵ when $z \sim f(\cdot|\theta)$. Note that the probability of a hit is proportional to Z_ϵ.

In Marjoram et al. (2003), an MCMC algorithm for sampling from $\hat{\pi}_\epsilon$ is presented, which again takes advantage of the property that if the proposal for z' is $f(\cdot|\theta')$, then it is not necessary to compute $f(z'|\theta')$ in the acceptance ratio. This scheme is given in Algorithm 4.12 and can be useful in ensuring that most of the computational

resources are spent in areas of high mass for θ. An alternative approach is to use SMC, as in Sisson et al. (2007); Beaumont et al. (2009); Del Moral et al. (2011) where there is some additional flexibility in the choice of kernels that are used to move the particles.

Algorithm 4.12 Standard ABC-MCMC Algorithm

Given (θ_0, z_0), at times $t = 0, 1, \ldots$

1. Set $(\theta, z) = (\theta_t, z_t)$.
2. Sample $\theta' \sim g(\cdot|\theta)$.
3. Sample $z' \sim f(\cdot|\theta')$
4. With probability

$$\min\left\{1, \mathbf{1}_{B_\epsilon(z')}(y)\frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)}\right\},$$

set $(\theta_{t+1}, z_{t+1}) = (\theta', z')$.
Otherwise, set $(\theta_{t+1}, z_{t+1}) = (\theta, z)$.

4.6.1.1 Fixed N Approach

ABC is a standard pseudo-marginal application and a simple MCMC kernel for exploring $\hat{\pi}_\epsilon$, using the framework developed in this chapter, is to use Algorithm 4.5 with $q(\cdot|\theta) = f(\cdot|\theta)$. The acceptance probability is given by

$$\min\left\{1, \frac{p(\theta')g(\theta|\theta')\sum_{j=1}^N \mathbf{1}_{B_\epsilon(z^{(j)})}(y)}{p(\theta)g(\theta'|\theta)\sum_{j=1}^N \mathbf{1}_{B_\epsilon(x^{(j)})}(y)}\right\}$$

where $\sum_{j=1}^N \mathbf{1}_{B_\epsilon(z^{(j)})}(y)$ is the number of hits observed from N attempts on θ' and $\sum_{j=1}^N \mathbf{1}_{B_\epsilon(x^{(j)})}(y)$ is one plus the number of hits observed from $N-1$ attempts on θ. The ratio of these terms is a consistent estimator of $f_\epsilon(y|\theta')/f_\epsilon(y|\theta)$ so that this multiple-try Metropolis type algorithm is indistinguishable from the marginal algorithm that computes these terms explicitly as $N \to \infty$.

4.6.1.2 Adaptive N Approach

In the ABC context, it can be difficult to specify an appropriate value of N in advance, especially when $h_\epsilon(\theta)$ is correspondingly small. In particular, if $\mathcal{X} = \mathbb{R}^d$ and B uses a p-norm for its metric then Z_ϵ is given by

$$Z_\epsilon = \frac{[2\Gamma(1+1/p)\epsilon]^d}{\Gamma(1+d/p)}$$

so that $h_\epsilon(\theta)$ can be very small when d is large and/or when ϵ is small. This formula is a simple case of a result by Dirichlet (Edwards 1922, pp. 157-159), which can also supply expressions when B uses different norms for different components of y (see also Wang 2005). It is clear that Z_ϵ scales as ϵ^d so that for low values of ϵ, which are typically required for $\hat{\pi}_\epsilon$ to be close to π, $h_\epsilon(\theta)$ is low and can impact the rate of convergence of an MCMC kernel in this context. The expression for Z_ϵ also suggests that for large d, the infinity norm may be preferable from a computational perspective.

A consequence for the MCMC approach in Marjoram et al. (2003) is that small values of ϵ cause the kernel to get stuck at values of θ for many iterations. In Algorithm 4.12, the expected acceptance probability before sampling $z' \sim f(\cdot|\theta')$ is proportional to $h_\epsilon(\theta')$, and does not depend on $h_\epsilon(\theta)$. As such, even if $\theta' = \arg\max_\vartheta h_\epsilon(\vartheta)$, the probability of acceptance can be arbitrarily low depending on how diffuse $f(\cdot|\theta')$ is and how small ϵ is. It should be clear that for large enough N, this problem can be alleviated by using Algorithm 4.5 as in the previous section since the acceptance probability involves an approximation of $f_\epsilon(y|\theta')/f_\epsilon(y|\theta)$. However, the value of N necessary for this approximation to be satisfactory is generally unknown in advance so an adaptive scheme could be helpful.

In the SMC context, there are existing methods that have a similar motivation. The approach of Beaumont et al. (2009) is to use a non-MCMC kernel to perturb the particles at each time that includes a rejection sampling step. In particular, they define a sequential importance sampling algorithm where the target distribution at each time is given by $\hat{\pi}_{\epsilon_t}$ for some $\{\epsilon_t\}_{t=0}^T$ and the algorithm works only on the marginal space Θ. At time $t > 0$, one has a set of N weighted samples approximating

$\hat{\pi}_{\epsilon_{t-1}}$, denoted $(\theta_{t-1}^{(1:N)}, W_{t-1}^{(1:N)})$ and this approach uses a proposal density for each $\theta_t^{(i)}$ of the form

$$q(\theta_t^{(i)}|\theta_{t-1}^{(1:N)}, W_{t-1}^{(1:N)}) \propto f_{\epsilon_t}(y|\theta_t^{(i)}) \sum_{j=1}^{N} W_{t-1}^{(j)} g(\theta_t^{(i)}|\theta_{t-1}^{(j)})$$

where one samples from this density by rejection. The importance weight associated with $\theta_t^{(i)}$ is then given by

$$\frac{p(\theta_t^{(i)})}{\sum_{j=1}^{N} W_{t-1}^{(j)} g(\theta_t^{(i)}|\theta_{t-1}^{(j)})}.$$

While this algorithm has the beneficial quality of always moving, the computational cost of weighting all N samples is quadratic in N, the number of particles. When the $\hat{\pi}_{\epsilon_t}$'s can be well approximated for small N and the number of simulations required to perform the rejection sampling step is high, this is unlikely to have much of an impact on the running time, but it can become a bottleneck as N grows. Another approach, that is also a form of partial rejection control, is given in Peters et al. (2008).

An alternative to both of these partial rejection control approaches is to use an MCMC kernel to move each particle that adapts naturally to the diffusivity of $\hat{\pi}_{\epsilon_t}$, which is what we propose to do here. There are many possibilities for this adaptation, even in this highly constrained application area. In particular, the choice of proposal is constrained to ensure that evaluations of the likelihood function f are not necessary but the choice of possible stopping probabilities β_j and $\bar{\beta}_j$ is vast. The general form of an adaptive MCMC-ABC scheme with distinct N, N' is given in Algorithm 4.13. The form of the proposal density for q in Algorithm 4.13 is

$$q(N', z^{(1:N')}|\theta') = \beta_{N'}(z^{(1:N')}) f(z^{(N')}|\theta') \prod_{j=1}^{N'-1} [1 - \beta_j(z^{(1:j)})] f(z^{(j)}|\theta')$$

while the form of $\bar{q}(N, u^{(2:N)}|\theta, u^{(1)})$ is

$$\bar{q}(N, u^{(2:N)}|\theta, u^{(1)}) = \bar{\beta}_N(u^{(1:N)}) f(u^{(N)}|\theta)[1 - \bar{\beta}_1(u^{(1)})] \prod_{j=2}^{N-1} [1 - \bar{\beta}_j(u^{(1:j)})] f(u^{(j)}|\theta).$$

The acceptance probability of a swap move can be written as

$$\min\left\{1, \frac{\hat{\pi}_\epsilon(\theta', v^{(1)})g(\theta|\theta')}{\hat{\pi}_\epsilon(\theta, u^{(1)})g(\theta'|\theta)} \times \rho_1\rho_2\right\}$$

where $\rho_1 = f(u^{(1)}|\theta)/f(v^{(1)}|\theta')$ and

$$\rho_2 = \frac{\beta_N(x^{(1:N)})\prod_{j=1}^{N-1}[1-\beta_j(x^{(1:j)})]}{\bar{\beta}_N(u^{(1:N)})\prod_{j=1}^{N-1}[1-\bar{\beta}_j(u^{(1:j)})]}\frac{\bar{\beta}_{N'}(v^{(1:N')})\prod_{j=1}^{N'-1}[1-\bar{\beta}_j(v^{(1:j)})]}{\beta_{N'}(z^{(1:N')})\prod_{j=1}^{N'-1}[1-\beta_j(z^{(1:j)})]},$$

with $v^{(1:N')} = \psi_{(N',l)}(z^{(1:N')})$ and $x^{(1:N)} = \psi^{-1}_{(N,k)}(u^{(1:N)})$.

Algorithm 4.13 Adaptive ABC-MCMC Algorithm with Distinct N, N'

Given $(\theta_0, u^{(1)})$, at times $t = 0, 1, \ldots$

1. Set $(\theta, u^{(1)}) = (\theta_t, u_t^{(1)})$.
2. Sample $\theta' \sim g(\cdot|\theta)$.
3. Sample $N', z^{(1:N')} \sim q(\cdot|\theta')$.
4. Sample $L = l$ with probability $\frac{w_L^{(l)}}{\sum_{j=1}^{N'} w_L^{(j)}}$.
5. Sample $N, u^{(2:N)} \sim \bar{q}(\cdot|\theta, u^{(1)})$.
6. Sample $K = k$ with probability $\frac{w_K^{(k)}}{\sum_{j=1}^{N} w_K^{(j)}}$.
7. Set $v^{(1:N')} = \psi_{(N',l)}(z^{(1:N')})$ and $x^{(1:N)} = \psi^{-1}_{(N,k)}(u^{(1:N)})$.
8. With probability

$$\min\left\{1, \frac{\tilde{\pi}(N', N, l, k, \theta', \theta, v^{(1:N')}, u^{(1:N)}, z^{(1:N')}, x^{(1:N)})}{\tilde{\pi}(N, N', k, l, \theta, \theta', u^{(1:N)}, v^{(1:N')}, x^{(1:N)}, z^{(1:N')})}\right\},$$

set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta', v^{(1)})$.
Otherwise, set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta, u^{(1)})$.

One possible approach is to let $\bar{\beta}_j = \beta_j$ be 1 if its arguments correspond to 2 hits and 0 otherwise, and let $\psi_{N,k}$ be defined by (4.9). To complete the scheme, it suffices to choose the form of $w_L^{(j)}$ and $w_K^{(j)}$, which we can define as $w_L^{(j)} = \mathbf{1}_{B_\epsilon(z^{(j)})}(y)/2$ for $j \in \{1, \ldots, N'\}$ and 0 otherwise, and $w_K^{(j)} = 1/N$ for $j \in \{1, \ldots, N\}$ and 0 otherwise. Because of the simple nature of this 0-1 stopping rule and the target density, one can enumerate the possible values of ρ_2 that can appear in the acceptance probability. In particular, $\rho_2 = 1$ if $l < N'$ or if $l = N'$ and $\mathbf{1}_{B_\epsilon(z^{(1)})}(y) = 1$ and 0 otherwise. This is due to the discrepancy between the probability of N' in the numerator and

denominator of the acceptance ratio. Note that the probability that $l = N'$ is $1/2$ since the number of hits is 2 and $z^{(N')}$ is by definition a hit. It is possible to remove this factor $1/2$ by defining instead $w_L^{(j)} = \mathbf{1}_{B_\epsilon(z^{(j)})}(y)$ for $j \in \{1, \ldots, N'-1\}$ and 0 otherwise, and $w_K^{(j)} = 1/(N-1)$ for $j \in \{1, \ldots, N-1\}$ and 0 otherwise. In this case, we have $\rho_2 = 1$ for any k, l with non-zero probability. The acceptance probability of the swap move simplifies to

$$\min\left\{1, \frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)} \times \frac{N-1}{N'-1}\right\}.$$

Pseudo-code for this particular scheme is given in Algorithm 4.14, for which the stopping rule and selection probabilities are an application of the observation in Section 4.5.2.2. The description of the algorithm is considerably less complex than the general form.

Algorithm 4.14 Adaptive 2-Hit ABC-MCMC Algorithm

Given $(\theta_0, u^{(1)})$, at times $t = 0, 1, \ldots$

1. Set $(\theta, u^{(1)}) = (\theta_t, u_t^{(1)})$.
2. Sample $\theta' \sim g(\cdot|\theta)$.
3. Sample $z^{(i)} \sim f(\cdot|\theta')$ for $i = 1, 2, \ldots, N'$ until 2 hits have been observed, i.e. $\sum_{j=1}^{N'} \mathbf{1}_{B_\epsilon(z^{(j)})}(y) = 2$ and $\mathbf{1}_{B_\epsilon(z^{(N')})}(y) = 1$.
4. Set $l = \min\{j : \mathbf{1}_{B_\epsilon(z^{(j)})}(y) = 1\}$.
5. Sample $u^{(i)} \sim f(\cdot|\theta)$ for $i = 2, 3, \ldots, N$ until 1 hit has been observed, i.e. $\sum_{j=2}^{N} \mathbf{1}_{B_\epsilon(u^{(j)})}(y) = 1$ and $\mathbf{1}_{B_\epsilon(u^{(N)})}(y) = 1$.
6. Sample $K = k$ with probability $\frac{1}{N-1}\mathbf{1}_{\{1,\ldots,N-1\}}(k)$.
7. With probability

$$\min\left\{1, \frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)} \times \frac{N-1}{N'-1}\right\},$$

set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta', z^{(l)})$.
Otherwise, set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta, u^{(1)})$.

There is a great degree of flexibility in constructing such schemes, and it is not immediately obvious what selection of constituent elements is optimal. Indeed, consider the choice of distribution for k, which we have defined via the weights $w_K^{(j)}$. One could define instead $w_K^{(j)} = 1/N$ for $j \in \{1, \ldots, N\}$ and 0 otherwise. In this

case, we have $\rho_2 = 1$ if $k \neq N$ and 0 if $k = N$. The acceptance probability is then

$$\min\left\{1, \frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)} \times \frac{N}{N'} \times \mathbf{1}_{\{1,\ldots,N-1\}}(k)\right\}$$

which leads to an MCMC kernel with a larger asymptotic variance. Indeed, by marginalizing out k the expected acceptance probability is

$$\min\left\{1, \frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)} \times \frac{N}{N'}\right\} \times \frac{N-1}{N} = \min\left\{\frac{N-1}{N}, \frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)} \times \frac{N-1}{N'}\right\}.$$

Note further that for any $C, N, N' > 0$ we have

$$\min\left\{1, C\frac{N-1}{N'-1}\right\} \geq \min\left\{1, C\frac{N-1}{N'}\right\} \geq \min\left\{\frac{N-1}{N}, C\frac{N-1}{N'}\right\}$$

so this scheme leads to greater asymptotic variance compared to Algorithm 4.14 by the arguments in (Tierney 1998, Section 3). In this case, the exclusion of $k = N$ in the conditional distribution for k is motivated by reducing the discrepancy between the distributions for k in the forward move and l in the reverse move.

We now consider another adaptive algorithm in the ABC context that uses a shared value of N. The general structure of such a scheme is given in Algorithm 4.15. The form of the proposal density for q in Algorithm 4.15 is

$$\begin{aligned} q(N, z^{(1:N)}, u^{(2:N)}|\theta, \theta', u^{(1)}) = \beta_N(z^{(1:N)}, u^{(1:N)}) f(z^{(N)}|\theta') f(u^{(N)}|\theta) \\ \times \prod_{j=2}^{N-1} [1 - \beta_j(z^{(1:j)}, u^{(1:j)})] f(z^{(j)}|\theta') f(u^{(j)}|\theta) \\ \times [1 - \beta_1(z^{(1)}, u^{(1)})] f(z^{(1)}|\theta'). \end{aligned}$$

The acceptance probability of a swap move can be written as

$$\min\left\{1, \frac{\hat{\pi}_\epsilon(\theta', v^{(1)})g(\theta|\theta')}{\hat{\pi}_\epsilon(\theta, u^{(1)})g(\theta'|\theta)} \times \rho_1\rho_2\right\}$$

where $\rho_1 = f(u^{(1)}|\theta)/f(v^{(1)}|\theta')$ and

$$\rho_2 = \frac{\beta_N(x^{(1:N)}, v^{(1:N)}) \prod_{j=1}^{N-1}[1 - \beta_j(x^{(1:j)}, v^{(1:j)})]}{\beta_N(z^{(1:N)}, u^{(1:N)}) \prod_{j=1}^{N-1}[1 - \beta_j(z^{(1:N)}, u^{(1:j)})]},$$

with $v^{(1:N)} = \psi_{(N,l)}(z^{(1:N)})$ and $x^{(1:N)} = \psi^{-1}_{(N,k)}(u^{(1:N)})$.

Algorithm 4.15 Adaptive ABC-MCMC Algorithm with Shared N

Given $(\theta_0, u^{(1)})$, at times $t = 0, 1, \ldots$

1. Set $(\theta, u^{(1)}) = (\theta_t, u_t^{(1)})$.
2. Sample $\theta' \sim g(\cdot|\theta)$.
3. Sample $N, u^{(2:N)}, z^{(1:N)} \sim q(\cdot|\theta, \theta', u^{(1)})$.
4. Sample $L = l$ with probability $\frac{w_L^{(l)}}{\sum_{j=1}^N w_L^{(j)}}$.
5. Sample $K = k$ with probability $\frac{w_K^{(k)}}{\sum_{j=1}^N w_K^{(j)}}$.
6. Set $v^{(1:N)} = \psi_{(N,l)}(z^{(1:N)})$ and $x^{(1:N)} = \psi^{-1}_{(N,k)}(u^{(1:N)})$.
7. With probability

$$\min\left\{1, \frac{\tilde{\pi}(N, l, k, \theta', \theta, v^{(1:N)}, u^{(1:N)}, z^{(1:N)}, x^{(1:N)})}{\tilde{\pi}(N, k, l, \theta, \theta', u^{(1:N)}, v^{(1:N)}, x^{(1:N)}, z^{(1:N)})}\right\},$$

set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta', v^{(1)})$.
Otherwise, set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta, u^{(1)})$.

A possible scheme in this setting is to let $\beta_j(z^{(1:j)}, u^{(1:j)}) = 1$ if

$$\sum_{j=1}^{N} \mathbf{1}_{B_\epsilon(z^{(j)})}(y) + \mathbf{1}_{B_\epsilon(u^{(j)})}(y) \geq 2,$$

and 0 otherwise. This means at least one extra hit has been observed, since $u^{(1)}$ is always a hit. In this setting, we can use $\psi_{N,k}$ defined by (4.9), and $w_K^{(j)} = w_L^{(j)} = 1$ if $j = N$ and 0 otherwise. In this case, $\rho_2 = 1$ if $\mathbf{1}_{B_\epsilon(u^{(N)})}(y) = 0$ and $N > 1$ and is 0 if $\mathbf{1}_{B_\epsilon(u^{(N)})}(y) = 1$ or $N = 1$. The acceptance probability can therefore be written

$$\min\left\{1, \frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)} \times \left[1 - \mathbf{1}_{B_\epsilon(u^{(N)})}(y) + \mathbf{1}_1(N)\right]\right\}.$$

Pseudo-code for this particular scheme is given in Algorithm 4.16.

This algorithm is the recommended algorithm for many reasons. In this simple ABC context, the probability that the move for a given proposal θ' is accepted can be

Algorithm 4.16 Adaptive 1-Hit ABC-MCMC Algorithm

Given $(\theta_0, u^{(1)})$, at times $t = 0, 1, \ldots$

1. Set $(\theta, u^{(1)}) = (\theta_t, u_t^{(1)})$.
2. Sample $\theta' \sim g(\cdot|\theta)$.
3. With probability $1 - \min\left\{1, \frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)}\right\}$, set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta, u^{(1)})$ and go to time $t+1$.
3. Sample $z^{(1)} \sim f(\cdot|\theta')$. If $\mathbf{1}_{B_\epsilon(z^{(1)})}(y) = 1$, set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta', z^{(1)})$ and go to time $t+1$.
4. For $i = 2, 3, \ldots$
   - Sample $u^{(i)} \sim f(\cdot|\theta)$.
   - If $\mathbf{1}_{B_\epsilon(u^{(i)})}(y) = 1$, set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta, u^{(1)})$ and go to time $t+1$.
   - Sample $z^{(i)} \sim f(\cdot|\theta')$.
   - If $\mathbf{1}_{B_\epsilon(z^{(i)})}(y) = 1$, set $(\theta_{t+1}, u_{t+1}^{(1)}) = (\theta', z^{(i)})$ and go to time $t+1$.

derived analytically, and is given by

$$\min\left\{1, \frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)}\right\} \times \frac{f_\epsilon(y|\theta')}{f_\epsilon(y|\theta) + f_\epsilon(y|\theta') - f_\epsilon(y|\theta)f_\epsilon(y|\theta')}.$$

This result shows that the transition probabilities have a limiting behaviour that is independent of the diffusivity of f, in the sense that for any $C > 0$,

$$\frac{Cf_\epsilon(y|\theta')}{Cf_\epsilon(y|\theta) + Cf_\epsilon(y|\theta') - C^2 f_\epsilon(y|\theta)f_\epsilon(y|\theta')} \geq \frac{f_\epsilon(y|\theta')}{f_\epsilon(y|\theta) + f_\epsilon(y|\theta')}.$$

A related, but distinct point, is that for a given f and y the transition probabilities associated with decreases in ϵ also have a limiting behaviour since we have $f_\epsilon(y|\theta) \to f(y|\theta)$ as $\epsilon \to 0$.

In fact, it can also be seen that if $\frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)} = 1$, which is possible if $p(\cdot)$ is a uniform prior and g is a symmetric proposal, Algorithm 4.14 has acceptance probability

$$\min\left\{1, \frac{f_\epsilon(\theta')}{f_\epsilon(\theta) + f_\epsilon(\theta') - f_\epsilon(\theta)f_\epsilon(\theta')}\right\}$$

although its expected cost in number of simulations is higher. However, in the event of large variation in the term $\frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)}$, Algorithm 4.14 may be preferable.

4.6.1.3 Remarks

The expected number of simulations for these adaptive algorithms cannot be computed without specification of the proposal g and prior p. In Algorithm 4.16, if a rejection does not occur in step 3, the expected number of simulations is

$$\frac{1}{h_\epsilon(\theta) + h_\epsilon(\theta') - h_\epsilon(\theta)h_\epsilon(\theta')}.$$

In Algorithm 4.14, a practical implementation would take advantage of the fact that both $(N', z^{(1:N')})$ and $(N, u^{(2:N)})$ can be sampled simultaneously, and once $(N', z^{(1:N')})$ or $(N, u^{(2:N)})$ is realized, one need only continue sampling the other until one knows the move will be accepted or rejected based on the value of the uniform random number used to perform this step. Nevertheless, the expected number of simulations is greater than that of Algorithm 4.16, and the above analytic transition probability suggests that this extra expense is unwarranted when one does not expect significant impact on the acceptance ratio from the prior and proposal terms.

A further comment is that the value of N in Algorithm 4.16 grows proportionally with $1/Z_\epsilon$ since $f_\epsilon(y|\theta) = h_\epsilon(\theta)/Z_\epsilon$. The expected value of N satisfies

$$\mathbb{E}[N] \geq \frac{1}{Z_\epsilon} \int_\Theta \hat{\pi}_\epsilon(\theta) \int_\Theta g(\theta'|\theta) \frac{1}{f_\epsilon(y|\theta) + f_\epsilon(y|\theta')} \min\left\{1, \frac{p(\theta')g(\theta|\theta')}{p(\theta)g(\theta'|\theta)}\right\} d\theta' d\theta$$

with this bound becoming tighter as $Z_\epsilon \to 0$.

The above result has the immediate implication that the expected number of simulations required to move to a new value of θ using Algorithm 4.16 is roughly twice that of the simplest approach, Algorithm 4.12, since $2N - 1$ is the approximate number of simulations performed per iteration in the former algorithm. Indeed, explicit computation of transition matrices is possible in small examples with discrete θ. In these cases inverse convergence rates can be calculated explicitly as the absolute value of the second largest eigenvalue in absolute value of these matrices. The expected value of N, denoted M, for Algorithm 4.16 can also be computed. One can denote the inverse rate by ρ_N in the case of fixed N for Algorithm 4.5 applied to ABC, and ρ_a for the adaptive 1-hit algorithm. For various discrete Θ examples in which inverse rates of convergence can be computed exactly, we have found that

$\rho_a < \rho_1^M$ but $\rho_a > \rho_1^{2M-1}$, indicating that the convergence rate in terms of simulations from $f(\cdot|\theta)$ is faster for Algorithm 4.12. However, $\rho_a < \rho_M$, in line with the idea that the adaptive algorithm spends more computational resources for values of (θ, θ') where it is helpful. A final note is that the marginal transition matrix, corresponding to the situation where one computes $f(y|\theta)$ explicitly, can have a large negative eigenvalue. As $N \to \infty$, the transition matrix associated with Algorithm 4.5 approaches that of the marginal algorithm. However, for intermediate values of N, the value of the second largest eigenvalue can be smaller in absolute value than this negative eigenvalue, indicating that intermediate values of N can have a faster rate of convergence, but perhaps at the cost of a larger asymptotic variance.

It is possible to define many different forms for the stopping probabilities β_j and $\bar{\beta}_j$. For example, one may want to wait until some number of hits r have been observed on each value of θ or both combined, as straightforward generalizations of Algorithms 4.14 and 4.16 respectively. This could be an application-specific choice, since the computational overhead for setting up simulations with a new value of θ could be very high in relation to the expense of running a simulation.

The algorithms presented in this context do not require explicit computation of the adaptive correction factor ρ_2 because one can determine exactly the value of this function by considering all possible values it can take. In general, one will have to compute this function explicitly at some computational expense but without the need to consider what values it can take.

The adaptive algorithms described here correspond to MCMC kernels for fixed ϵ. However, one can use these kernels for a fixed number of steps using different values of ϵ in an SMC sampler approach to ensure that there is a high probability of each particle moving when the proposed θ' has high mass under $\hat{\pi}_\epsilon$.

4.6.2 Model Selection

Pseudo-marginal algorithms are well suited to the problem of variable selection when the marginal likelihood of the data given the model cannot be computed

analytically. However, to showcase the differences in the algorithms, we focus on a variable selection example where the marginal likelihood can be computed explicitly. Consider the case where we have a set of n univariate observations $\mathbf{y} = y_{1:n} \in \mathbb{R}^n$ and a design matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ and we want to select from the 2^p models obtained by including only a subset of the p covariates in $\mathbf{X}$. As in Section 1.1.2.3, we can represent each model using a binary string of length p, $\gamma \in \{0,1\}^p$, where $\gamma_j = 1$ implies that variable j is in the model and $\gamma_j = 0$ implies it is not. Given a model $\mathcal{M}_\gamma$, we denote the associated coefficient by $\boldsymbol{\beta}_\gamma \in \mathbb{R}^{\|\gamma\|_1}$ and the associated design matrix by $\mathbf{X}_\gamma = \mathbf{X}_{1:n,\gamma}$, i.e. only columns of $\mathbf{X}$ corresponding to covariates in the model are included in $\mathbf{X}_\gamma$. We define the likelihood f_γ of $\mathbf{y}$ given $\mathcal{M}_\gamma$ as Gaussian with mean $\mathbf{X}_\gamma \boldsymbol{\beta}_\gamma$ and covariance $\sigma^2 I_\gamma$. We additionally define the density $g(\cdot|\mathcal{M}_\gamma)$ as Gaussian with mean $\mathbf{0}$ and covariance $\delta^2 I_\gamma$. In this case one can calculate the marginal likelihood of the model

$$f(\mathbf{y}|\mathbf{X}, \mathcal{M}_\gamma) = \int_{\mathbb{R}^{\|\gamma\|_1}} f_\gamma(\mathbf{y}|\mathbf{X}_\gamma, \boldsymbol{\beta}_\gamma) g(\boldsymbol{\beta}_\gamma|\mathcal{M}_\gamma) d\boldsymbol{\beta}_\gamma \tag{4.19}$$

analytically to check the output of an MCMC scheme that will instead use a pseudo-marginal algorithm to target

$$\pi(\mathcal{M}_\gamma) = p(\mathcal{M}_\gamma|\mathbf{y}) = \frac{f(\mathbf{y}|\mathbf{X}, \mathcal{M}_\gamma) p(\mathcal{M}_\gamma)}{\sum_{\gamma' \in \{0,1\}^p} f(\mathbf{y}|\mathbf{X}, \mathcal{M}'_\gamma) p(\mathcal{M}'_\gamma)}.$$

A suitable auxiliary variable for each model is $\boldsymbol{\beta}_\gamma$ in that

$$\pi(\mathcal{M}_\gamma, \boldsymbol{\beta}_\gamma) = p(\mathcal{M}_\gamma, \boldsymbol{\beta}_\gamma|\mathbf{y}) = \frac{f_\gamma(\mathbf{y}|\mathbf{X}_\gamma, \boldsymbol{\beta}_\gamma) g(\boldsymbol{\beta}_\gamma|\mathcal{M}_\gamma) p(\mathcal{M}_\gamma)}{\sum_{\gamma' \in \{0,1\}^p} f(\mathbf{y}|\mathbf{X}, \mathcal{M}'_\gamma) p(\mathcal{M}'_\gamma)}$$

does not require the computation of (4.19).

We present two simple low-dimensional problems here and present SMC methodology as a possible solution to issues arising when p is large in the next chapter. In the following examples we take $\sigma^2 = \delta^2 = 1$ as fixed and known.

4.6.2.1 Simple Case

We first show how simple pseudo-marginal methods work well when p is small and one uses an importance density for $\boldsymbol{\beta}_\gamma$ that is close to

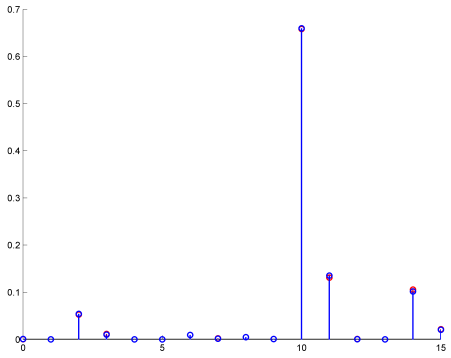
$$p(\boldsymbol{\beta}_\gamma|\mathbf{y}, \mathbf{X}_\gamma, \mathcal{M}_\gamma) = \frac{f_\gamma(\mathbf{y}|\mathbf{X}_\gamma, \boldsymbol{\beta}_\gamma)g(\boldsymbol{\beta}_\gamma|\mathcal{M}_\gamma)}{f(\mathbf{y}|\mathbf{X}, \mathcal{M}_\gamma)}.$$

For $p = 4$ and $n = 50$, we sample $\mathbf{X}^{n\times p}$ from a multivariate normal distribution with mean $\mathbf{0}$ and covariance I_4. We let $\beta_2 = \beta_4 = 0$ and sample β_1 and β_3 independently from a standard normal distribution. Then we sample $\mathbf{y}$ from the likelihood $f_\gamma(\cdot|\mathbf{X}_\gamma, \boldsymbol{\beta}_\gamma)$ where $\gamma = \{1, 0, 1, 0\}$.

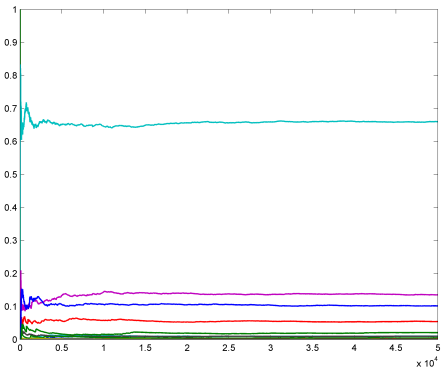
We now perform inference on γ given $\mathbf{y}$ using a simple multiple-try type pseudo-marginal algorithm with independent proposals, a conditioned or dependent pseudo-marginal algorithm that conditions the proposals for $\mathbf{z}^{(1:N)}$ on $\mathbf{x}^{(k)}$ and the proposals for $\mathbf{x}^{(-k)}$ on $\mathbf{z}^{(l)}$ and an adaptive conditioned pseudo-marginal algorithm. In this application the variables $\mathbf{x}$ and $\mathbf{z}$ correspond to values of $\boldsymbol{\beta}_\gamma$ and we have $\theta = \gamma$ and $\theta' = \gamma'$. For the non-adaptive algorithms we use $N = 100$ samples and for the adaptive algorithm we use an effective sample size threshold of 10 with a 0-1 stopping rule for the β_j's, but attempt to reduce the probability of $\rho_2 = 0$ by not permitting the selection of $l = N$.

For the multiple-try pseudo-marginal algorithm, a t-distribution with 4 degrees of freedom and mean 0 is used to independently propose each value of $z_j^{(i)}$ for $i \in \{1, \ldots, N\}$ and all j such that $\gamma'_j = 1$ and $x_j^{(i)}$ for $i \in \{1, \ldots, N\} \setminus \{k\}$ and all j such that $\gamma_j = 1$. For the conditioned and adaptive algorithms, for all $i \in \{1, \ldots, N\}$, the proposal for $z_j^{(i)}$ is a t-distribution with 4 degrees of freedom and mean 0 if $\gamma_j = 0$ and a t-distribution with 4 degrees of freedom and mean $x_j^{(k)}$ if $\gamma_j = 1$ for all j such that $\gamma'_j = 1$. An analogous proposal is used for $x^{(-k)}$ conditional on $z^{(l)}$ and both γ and γ'.

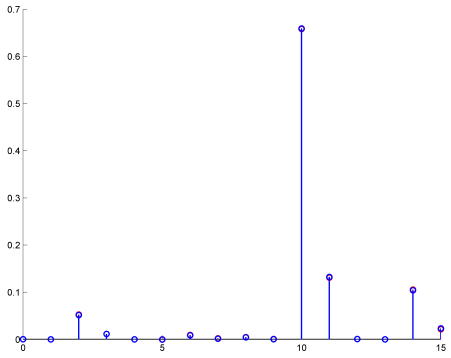
Each MCMC algorithm is run for 50000 iterations with the resulting estimates of the model probabilities given in Figure 4.1. It is clear that all the kernels mix quickly and provide good estimates of the model probabilities.



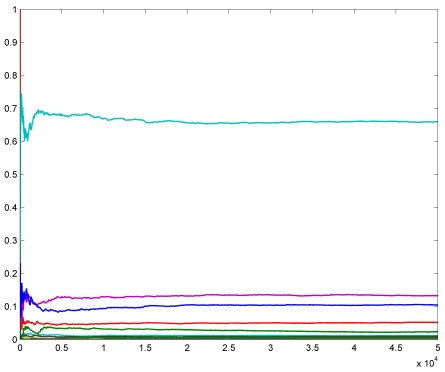
(a) MTM



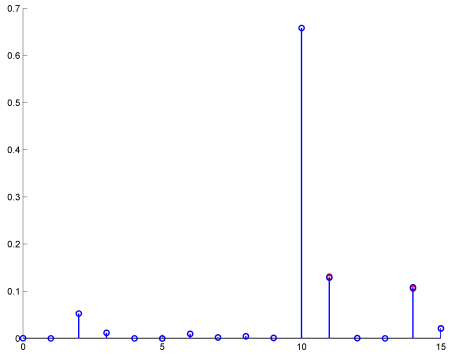
(b) MTM



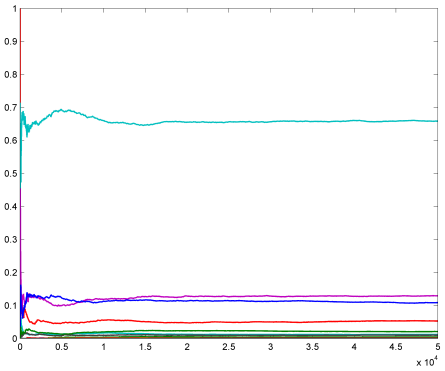
(c) Conditioned PM



(d) Conditioned PM



(e) Adaptive PM



(f) Adaptive PM

Figure 4.1: Estimated model probabilities for the simple case (left) along with the Monte Carlo estimate of the model probabilities at each iteration (right). The red stem plots on the left indicate the true model probabilities.

4.6.2.2 Benefits of Conditioning

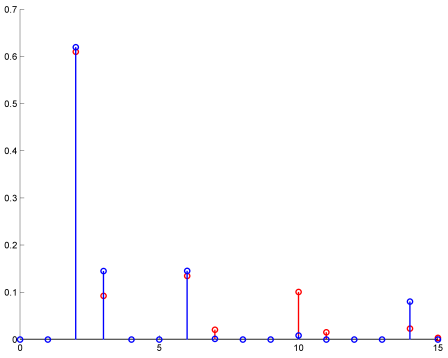
To highlight the potential benefits of having the proposals for the auxiliary variables depend on the current value of $x^{(k)}$, we now simulate data with $p = 4$ and $n = 50$ as before, with the same distribution for $\mathbf{X}$ but we set $\beta_3 = -10.3429$ with all other β_j's equal to zero. The same algorithms are used and each algorithm is run for 50000 iterations as before. The resulting estimates of the model probabilities given in Figure 4.2. In this case, it is apparent that the use of independent proposals leads to slow mixing, while both the conditioned pseudo-marginal and adaptive pseudo-marginal algorithms converge quickly. The adaptive algorithm in addition converges more quickly than the non-adaptive algorithm.

4.6.2.3 Effect of Adaptation

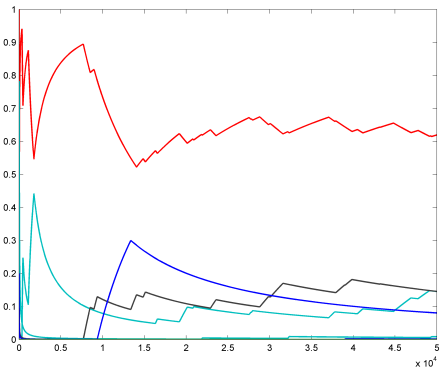
In this setting, adapting N using an ESS threshold is reasonably straightforward to interpret. The value of N is essentially adapted to the difficulty in obtaining 10 effective samples of $\boldsymbol{\beta}_\gamma$, which depends largely on the dimension of the model associated with γ. Figure 4.3 shows the relationship between $\log N$ and the dimension of the model, and we can see that N grows exponentially in the dimension. This indicates both that the adaptation of N is successful in ensuring that the marginal likelihood is estimated somewhat robustly for each model but also that the adaptation of N with the use of independent proposals cannot overcome the curse of dimensionality. As such, in Section 5.4.2 we will use SMC as a mechanism for proposing values of $\boldsymbol{\beta}_\gamma$ to keep the amount of computation required by the kernel reasonable.

4.7 Discussion

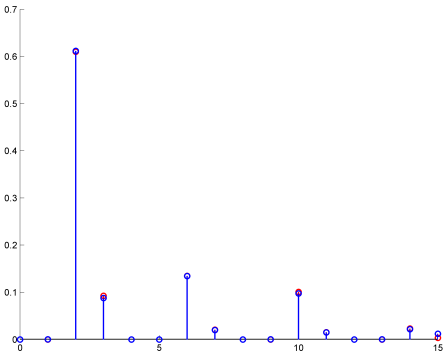
This chapter has presented a perspective on the construction of reversible MCMC kernels that target a given density π that is known up to a normalizing constant. The developments are generally motivated by the desire to include many auxiliary variables in a proposal, since these auxiliary variables can often be generated in



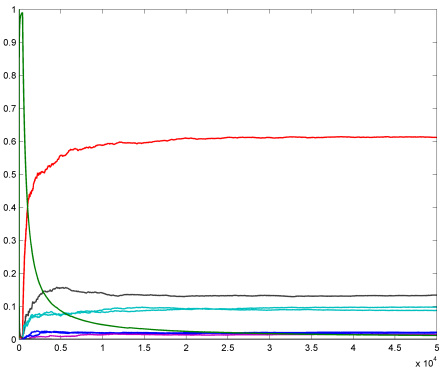
(a) MTM



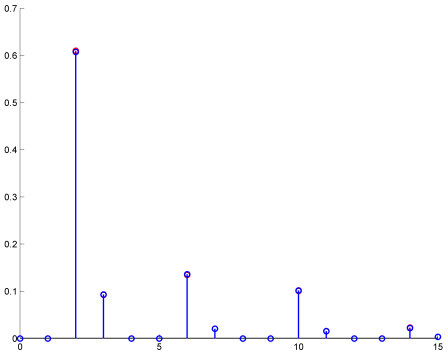
(b) MTM



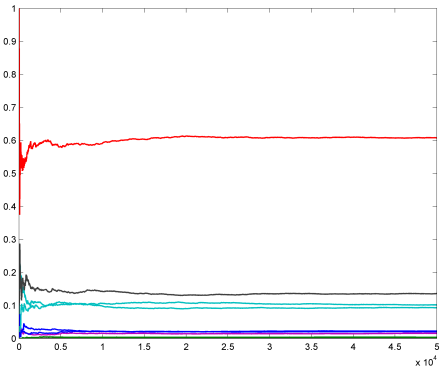
(c) Conditioned PM



(d) Conditioned PM



(e) Adaptive PM



(f) Adaptive PM

Figure 4.2: Estimated model probabilities with large β_3 (left) along with the Monte Carlo estimate of the model probabilities at each iteration (right). The red stem plots on the left indicate the true model probabilities.

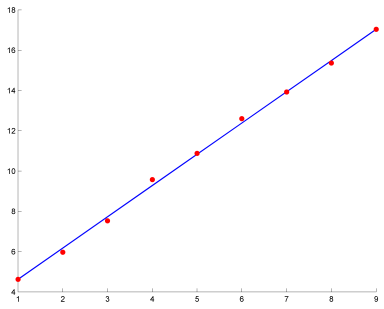


Figure 4.3: Plot of $\log N$ against the dimension of the model. The red dots are the average values of N over 10 trials for each model of the given dimension and the blue line is a linear approximation of the trend. There was practically no difference in the average value of N between models of the same dimension using the ESS criterion.

parallel and hence involve little to no extra computational cost on modern computing machines. However, the perspective can also be applied to understand more general kernels that do not involve the generation of a large number of auxiliary random variables.

The kernels, when perceived as a superficial generalization of Metropolis-Hastings kernels to the case where some fixed number of proposals N are generated instead of 1, can be seen as population-based MCMC kernels. However, in contrast to most population-based MCMC methods, the auxiliary random variables in this case have distributions that can be sampled from easily and in most situations enable the Markov kernel to be viewed as operating on the space $\mathcal{X}$ on which π is defined. This collapsing of the state of the chain is not generally seen in other population-based methods. In addition, the framework provides a natural interpretation and generalization of pseudo-marginal methods that have appeared in Beaumont (2003) and Andrieu and Roberts (2009), allowing proposals to depend on the current state of the chain as well as avoiding some scenarios in which the MCMC sampler gets 'stuck'.

In some studies with π defined on a small discrete set, and when the proposals are independent, we have observed that the computable, exact rate of convergence is not improved for any fixed value of N to the extent that these kernels offer advantages when analyzed from the perspective of serial computation. This is in accordance with comments to this effect in (Murray 2007, Section 3.2) and Neal (2011), but is

important to note that when implemented on a parallel architecture, the optimal choice of N can be superior to 1. It is also potentially useful to be able to guarantee that a kernel has a reasonable rate of convergence when it is used within another scheme such as an SMC sampler. This can be accomplished to a certain extent for ABC-MCMC kernels in the sense that we can derive kernels whose acceptance probability for a move $\theta \to \theta'$ has a limiting behaviour that depends only on the relative likelihoods $f_\epsilon(y|\theta)$ and $f_\epsilon(y|\theta')$. The extra computational power required to traverse the posterior for small values of ϵ, or if f_ϵ is very diffuse, is automatically accounted for in the kernel by sampling more auxiliary variables in expectation.

While not pursued here, it would be helpful to be able to characterize the relationship between N and the rate of convergence or the asymptotic variance of the resulting chain. Under some mild conditions one can derive the asymptotic kernel that is obtained as $N \to \infty$, and this makes clear that there is a point at which the extra computational effort required to move according to a kernel with extremely large values of N is unwarranted in terms of obtaining timely Monte Carlo estimates of an expectation of a given function under π. In some examples, the rate of convergence for intermediate values of N is faster than that of both the asymptotic kernel and the kernel with $N = 1$, although this could be at the cost of larger asymptotic variance.

In the next chapter, we will see how dependent proposals can be built in a systematic and effective way using SMC methods. This builds upon the work in Andrieu et al. (2010) and provides examples of situations in which intermediate values of N can provide superior performance to $N = 1$ even if the computation is not performed in parallel. Additionally, two algorithms that do not necessarily use multiple proposals but which rely on the use of auxiliary variables for other means are discussed from this perspective.

Chapter 5

Active Particle Views of Advanced MCMC Algorithms

5.1 Introduction

In this chapter, the perspective on reversible MCMC kernels developed in Chapter 4 is applied to proposals obtained through the use of particle methods. These powerful methods provide a systematic way to construct effective, dependent proposals whose density is typically close to a given target density π. This can be motivated by the observation that multiple independent proposals, while providing some benefit on a parallel architecture, do not seem to provide the same kind of benefit that dependent proposals can. A clear example is sampling latent variables in a state-space model.

We will first introduce SMC methodology in more detail than was done in Section 1.2.4 and, following this, explicitly define the joint density of the particles produced by such algorithms as well as a suitable extended target density that admits π as a marginal. This construction can then be used within an MCMC algorithm using the perspective developed in Chapter 4. The MCMC algorithms presented here extend those in in Andrieu et al. (2010) so that the sequence of active particles is also a Markov chain with equilibrium density π. Following derivation of particle MCMC algorithms in this perspective, we will then see how the number of particles

can be adapted at each time within the SMC algorithm without compromising the equilibrium distribution of the MCMC kernel.

In order to demonstrate the wider value of the perspective beyond just the introduction of multiple proposals, the active particle view is then applied to the interpretation of the stochastic potential switching algorithm of Mak (2005) and the auxiliary variable approach of Møller et al. (2006) for dealing with doubly intractable distributions. In the first case, the perspective allows for easy interpretation of the resulting kernel, whilst in the second this ease of interpretation additionally suggests potential modifications to the approach that could be beneficial in practice.

5.2 Particle Methods

The first step in incorporating sequential proposals with interactions through resampling is to determine the joint density of the particles that are produced by such a proposal, and a corresponding extended target density that admits the target density π as a marginal. In order to do this, we will have to reexamine the algorithms presented in Section 1.2.4 and present SMC in more detail.

5.2.1 Extended Target Density for SMC

In an SMC algorithm, one targets a sequence of densities $\pi_0, \ldots, \pi_T$, where $\pi_t : \mathcal{X}^{t+1} \to \mathbb{R}^+$, which we can compute up to a normalizing constant, i.e. we can compute $\gamma_t(x_{0:t}) = Z_t \pi_t(x_{0:t})$ where $Z_t = \int_{\mathcal{X}^{t+1}} \gamma_t(z_{0:t}) dz_{0:t}$. The density that we wish to obtain samples from is $\pi = \pi_T$. The algorithm progresses by generating a set of N samples in $\mathcal{X}$ at each time t, $x_t^{(1:N)}$, where $t \in \{0, \ldots, T\}$. Associated with each sample $x_t^{(i)}$ is a set of t samples, one from each time $j = t-1, \ldots, 0$, that together with $x_t^{(i)}$ constitute a sample, or particle, in $\mathcal{X}^{t+1}$ that is to be used in a particle approximation of π_t. The next section describes the notation that we will adopt to describe the set of particles produced more formally.

Let us denote all the samples produced in an SMC algorithm by $x_{0:T}^{(1:N)} \in \mathcal{X}^{N\times(T+1)}$ and the ancestral lineage of the particles by $A_{0:T-1}^{(1:N)} \in \{1,\ldots,N\}^{N\times T}$. The particles $x_t^{(1:N)}$ are the proposed particles at time t and $A_{t-1}^{(i)}$ denotes the index of the "parent" of particle i at time t. For clarity, we introduce the notation $C_{t-1}^i \in \{1,\ldots,N\}^t$ which represents the history of particle i at time t and is defined recursively by $C_{t-1,t-1}^i = A_{t-1}^{(i)}$ and then for $j = t-2,\ldots,0$ we let $C_{t-1,j}^i = A_j^{(C_{t-1,j+1}^i)}$. Using this notation, each of the N particles at time t can be denoted by $(x_{0:t-1}^{C_{t-1}^i}, x_t^{(i)})$ for each $i \in \{1,\ldots,N\}$ and where we have $x_{0:t-1}^{C_{t-1}^i} = (x_0^{(C_{t-1,0}^i)},\ldots,x_{t-1}^{(C_{t-1,t-1}^i)})$.

The conditional densities from which each particle $x_t^{(i)}$ is sampled are denoted g_t for $t \in \{0,\ldots,T\}$, which are to a certain extent arbitrary. Associated with each particle at time t is an importance weight to correct for the discrepancy between the proposal mechanism for the particles, which includes resampling, and the intermediate target densities π_t. To ease the presentation, we will further define $w_t^{(i)} = w_t(x_{0:t-1}^{C_{t-1}^i}, x_t^{(i)})$ and $W_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^N w_t^{(j)}}$.

One can now refer to Algorithm 5.1 to view the SMC methodology rewritten in the most simple case, where resampling occurs at every time. We will further assume that multinomial resampling is used, i.e. we sample $A_t^{(1:N)} \sim \mathcal{M}(\cdot|N, W_t^{(1:N)})$ where $\mathcal{M}(a^{(1:j)}|j, W^{(1:N)})$ denotes the probability mass function of j independent draws $a^{(1)},\ldots,a^{(j)}$ from a categorical distribution with probabilities given by $W^{(1:N)}$. This implies that if $n_k = \sum_{i=1}^j \mathbf{1}_k(a^{(i)})$ then the vector $n_{1:N}$ is a draw from the multinomial distribution with parameters j and $W^{(1:N)}$.

The proposal density for all the samples $x_{0:T}^{(1:N)}$ and the ancestral lineage $A_{0:T-1}^{(1:N)}$ is given by

$$q(x_{0:T}^{(1:N)}, A_{0:T-1}^{(1:N)}) = \left[\prod_{i=1}^N g_0(x_0^{(i)})\right]\left\{\prod_{t=1}^T \mathcal{M}\left(A_{t-1}^{(1:N)}\middle|N, W_{t-1}^{(1:N)}\right)\prod_{i=1}^N g_t(x_t^{(i)}|x_{0:t-1}^{C_{t-1}^i})\right\}.$$

This density can be extended to include the index k of a particle selected proportional to its weight at time T via:

$$q(k, x_{0:T}^{(1:N)}, A_{0:T-1}^{(1:N)}) = q(x_{0:T}^{(1:N)}, A_{0:T-1}^{(1:N)})\frac{w_T^{(k)}}{\sum_{i=1}^N w_T^{(i)}}.$$

Algorithm 5.1 SMC Reformalized

1. At time $t = 0$.
   - For $i = 1, \ldots, N$, sample $x_0^{(i)} \sim g_0(\cdot)$
   - For $i = 1, \ldots, N$, evaluate the importance weights:

$$w_0(x_0^{(i)}) = \frac{\gamma_0(x_0^{(i)})}{g_0(x_0^{(i)})}.$$

2. For times $t = 1, \ldots, T$.
   - Sample $A_{t-1}^{(1:N)} \sim \mathcal{M}(\cdot|N, W_{t-1}^{(1:N)})$.
   - For $i = 1, \ldots, N$, sample $x_t^{(i)} \sim g_t(\cdot|x_{0:t-1}^{C_{t-1}^i})$.
   - For $i = 1, \ldots, N$, evaluate the importance weights:

$$w_t(x_{0:t-1}^{C_{t-1}^i}, x_t^{(i)}) = \frac{\gamma_t(x_{0:t-1}^{C_{t-1}^i}, x_t^{(i)})}{\gamma_{t-1}(x_{0:t-1}^{C_{t-1}^i}) g_t(x_t^{(i)}|x_{0:t-1}^{C_{t-1}^i})}.$$

So that we can next define an extended target density that both admits π_T as a marginal and is defined on the same space as our joint density for the proposal, it is necessary to extend q to include the history of the selected particle, C_{T-1}^k, which is deterministic upon the choice of k. We define $\mathbf{k} = k_{0:T} \in \{1, \ldots, N\}^{T+1}$ and write

$$q(\mathbf{k}, x_{0:T}^{(1:N)}, A_{0:T-1}^{(1:N)}) = q(k_T, x_{0:T}^{(1:N)}, A_{0:T-1}^{(1:N)}) \delta_{(C_{T-1}^{k_T})}(\mathbf{k}_{0:T-1}).$$

We can introduce an extended target density $\tilde{\pi}^N$ of the form

$$\tilde{\pi}^N(\mathbf{k}, x_{0:T}^{(1:N)}, A_{0:T-1}^{(1:N)}) = N^{-(T+1)} \pi(x_{0:T}^{\mathbf{k}}) \bar{q}(x_{0:T}^{-\mathbf{k}}, A_{0:T-1}^{-\mathbf{k}} | \mathbf{k}, x_{0:T}^{\mathbf{k}})$$

where

$$\bar{q}(x_{0:T}^{-\mathbf{k}}, A_{0:T-1}^{-\mathbf{k}} | \mathbf{k}, x_{0:T}^{\mathbf{k}}) = \left[\prod_{i=1, i \neq k_0}^{N} g_0(x_0^{(i)}) \right] \left\{ \prod_{t=1}^{T} \mathcal{M}\left(A_{t-1}^{(-k_t)} \middle| N, W_{t-1}^{(1:N)} \right) \prod_{i=1, i \neq k_t}^{N} g_t(x_t^{(i)} | x_{0:t-1}^{C_{t-1}^i}) \right\}$$

and the term $N^{-(T+1)}$ refers to the probability of $\mathbf{k}$ which is, in this case, uniform on $\{1, \ldots, N\}^{T+1}$.

The ratio of the extended target density and the proposal density is given by:

$$\begin{aligned}\frac{\tilde{\pi}^N(\mathbf{k}, x_{0:T}^{(1:N)}, A_{0:T-1}^{(1:N)})}{q(\mathbf{k}, x_{0:T}^{(1:N)}, A_{0:T-1}^{(1:N)})} &= \frac{N^{-(T+1)}\pi(x_{0:T}^{\mathbf{k}})\bar{q}(x_{0:T}^{-\mathbf{k}}, A_{0:T-1}^{-\mathbf{k}}|\mathbf{k}, x_{0:T}^{\mathbf{k}})}{q(\mathbf{k}, x_{0:T}^{(1:N)}, A_{0:T-1}^{(1:N)})} \\ &= \frac{\pi(x_{0:T}^{\mathbf{k}})\prod_{t=0}^{T}\left[\frac{1}{N}\sum_{j=1}^{N} w_t^{(j)}\right]}{g(x_0^{k_0})\prod_{t=1}^{T} g(x_t^{k_t}|x_{0:t-1}^{k_{0:t-1}})\prod_{t=0}^{T} w_t^{(k_t)}} \\ &= \frac{\pi(x_{0:T}^{\mathbf{k}})}{\gamma_T(x_{0:T}^{\mathbf{k}})}\prod_{t=0}^{T}\left[\frac{1}{N}\sum_{j=1}^{N} w_t^{(j)}\right] = \frac{1}{Z_T}\prod_{t=0}^{T}\left[\frac{1}{N}\sum_{j=1}^{N} w_t^{(j)}\right] \end{aligned} \tag{5.1}$$

where $Z_T = \int \gamma_T(x_{0:T})dx_{0:T}$. This derivation also shows that the estimate of the normalizing constant $\hat{Z}_T = N^{-(T+1)}\prod_{t=0}^{T}\left[\sum_{j=1}^{N} w_t^{(j)}\right]$ is unbiased since it can be seen as the importance weight obtained with unnormalized density $\gamma_T \propto \pi$ as the target. A more general proof of the unbiasedness result can be found in Del Moral (2004) but it is instructive to additionally understand the densities on which the importance weight (5.1) is defined, as this provides a number of methodological opportunities.

An important difference between particle methods and those seen so far is that the selected particle after running SMC can be indexed by k. However, for the backwards proposal one must generally select an index $k_t \in \{1, \ldots, N\}$ at every time $t \in \{0, \ldots, T\}$, which is the position of the fixed particle at each time. The marginal probability of $\mathbf{k}$ in the target is essentially arbitrary, but is chosen here to be uniform because the particles one chooses from are exchangeable. We will see when we introduce criterion-based resampling, for example, that the distribution of k_t should generally be dependent on $(x_{0:t-1}^{(1:N)}, A_{0:t-1}^{(1:N)})$, and as a special case of this, dependent only on $W_{t-1}^{(1:N)}$.

A point of clarity is that the extended target $\tilde{\pi}^N$ is not an extended target density that corresponds to an MCMC algorithm. Instead, it is a target density for one set of proposals generated by an SMC algorithm. This target will, in Section 5.2.2, be extended further in order to derive MCMC algorithms that target π in essentially the same manner as was done in previous sections.

5.2.1.1 Conditional Resampling

Consider the addition of conditional resampling according to a rule based on all particles and their ancestors at time t. Assume further that the rule is invariant to permutations of particle indices that keep ancestry consistent. We will, in fact, assume that resampling occurs based only on the weights of the particles but not the indices of the weights, which is common in practice, for clarity. An SMC algorithm with conditional resampling is presented in Algorithm 5.2

Algorithm 5.2 SMC with Conditional Resampling

1. At time $t = 0$.
   - For $i = 1, \ldots, N$, sample $x_0^{(i)} \sim g_0(\cdot)$
   - For $i = 1, \ldots, N$, evaluate the importance weights:

$$w_0(x_0^{(i)}) = \frac{\gamma_0(x_0^{(i)})}{g_0(x_0^{(i)})}.$$

2. For times $t = 1, \ldots, T$.
   - With probability $\xi(W_{t-1}^{(1:N)})$, sample $A_{t-1}^{(1:N)} \sim \mathcal{M}(\cdot|N, W_{t-1}^{(1:N)})$ and set $r_{t-1} = 1$, $\omega_t^{(1:N)} = (1, \ldots, 1)$.
     Otherwise, set $A_{t-1}^{(1:N)} = (1, \ldots, N)$, $r_{t-1} = 0$ and $\omega_t^{(1:N)} = w_{t-1}^{(1:N)}$.
   - For $i = 1, \ldots, N$, sample $x_t^{(i)} \sim g_t(\cdot|x_{0:t-1}^{C_{t-1}^i})$.
   - For $i = 1, \ldots, N$, evaluate the importance weights:

$$w_t(x_{0:t-1}^{C_{t-1}^i}, x_t^{(i)}) = \omega_t^{(i)} \frac{\gamma_t(x_{0:t-1}^{C_{t-1}^i}, x_t^{(i)})}{\gamma_{t-1}(x_{0:t-1}^{C_{t-1}^i}) g_t(x_t^{(i)}|x_{0:t-1}^{C_{t-1}^i})}.$$

There are at least two possible ways to determine the correct acceptance weight in this scenario, and the two we consider depend on the description of the conditional resampling step. The first corresponds to what is traditionally implemented: if resampling is performed at time t, one samples $A_{t-1}^{(1:N)} \sim \mathcal{M}(\cdot|N, W_{t-1}^{(1:N)})$ and if not, one sets $A_{t-1}^{(1:N)} = (1, \ldots, N)$, which is what is done in Algorithm 5.2. The second corresponds to a difference in what happens when resampling does not occur and instead sets $A_{t-1}^{(1:N)}$ to be a uniform random permutation of $(1, \ldots, N)$. We can define $\mathbf{r} = r_{0:t-1} \in \{0, 1\}^T$ where $r_t = 1$ implies resampling takes place at time $t + 1$ and $r_t = 0$ implies it does not.

In the first case, it is easiest to write the proposal density as given by

$$q(\mathbf{k}, x_{0:T}^*, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)}) = \left[\prod_{i=1}^{N} g_0(x_0^{(i)})\right] \times \left\{\prod_{t=1}^{T} q(r_{t-1}, A_{t-1}^{(1:N)}|W_{t-1}^{(1:N)}) \prod_{i=1}^{N} g_t(x_t^{(i)}|x_{0:t-1}^{C_{t-1}^i})\right\} \times \frac{w_T^{(k_T)}}{\sum_{i=1}^{N} w_T^{(i)}} \delta_{(C_{T-1}^{k_T})}(k_{0:T-1}) \delta_{x_{0:T}^{\mathbf{k}}}(x_{0:T}^*) \tag{5.2}$$

where

$$q(r_{t-1}, A_{t-1}^{(1:N)}|W_{t-1}^{(1:N)}) = \xi(W_{t-1}^{(1:N)})\delta_1(r_{t-1})\mathcal{M}\left(A_{t-1}^{(1:N)}\middle|N, W_{t-1}^{(1:N)}\right) + (1 - \xi(W_{t-1}^{(1:N)}))\delta_0(r_{t-1})\delta_{(1:N)}(A_{t-1}^{(1:N)})$$

and we consider the simple, but common case where ξ encodes the probability of resampling based on the values of the normalized weights at time $t-1$. One can then write the extended target as

$$\tilde{\pi}^N(\mathbf{k}, x_{0:T}^*, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)}) = \pi(x_{0:T}^*)\bar{q}(\mathbf{k}, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)}|x_{0:T}^*)$$

where

$$\bar{q}(\mathbf{k}, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)}|x_{0:T}^*) = \frac{1}{N}\delta_{x_0^*}(x_0^{(k_0)}) \left[\prod_{i=1, i\neq k_0}^{N} g_0(x_0^{(i)})\right] \times \left\{\prod_{t=1}^{T} \bar{q}(k_t, r_{t-1}, A_{t-1}^{(1:N)}|W_{t-1}^{(1:N)}, k_{t-1})\delta_{x_t^*}(x_t^{(k_t)}) \prod_{i=1, i\neq k_t}^{N} g_t(x_t^{(i)}|x_{0:t-1}^{C_{t-1}^i})\right\} \tag{5.3}$$

and

$$\bar{q}(k_t, r_{t-1}, A_{t-1}^{(1:N)}|W_{t-1}^{(1:N)}, k_{t-1}) = \xi(W_{t-1}^{(1:N)})\delta_1(r_{t-1})\frac{1}{N}\delta_{k_{t-1}}(A_{t-1}^{(k_t)}) \times \mathcal{M}\left(A_{t-1}^{(-k_t)}\middle|N-1, W_{t-1}^{(1:N)}\right) + (1 - \xi(W_{t-1}^{(1:N)}))\delta_0(r_{t-1})\delta_{k_{t-1}}(k_t)\delta_{(1:N)}(A_{t-1}^{(1:N)}).$$

Note that the factor N^{-1} in $\bar{q}(k_t, r_{t-1}, A_{t-1}^{(1:N)}|W_{t-1}^{(1:N)}, k_{t-1})$ corresponds to a uniform distribution over $\{1, \ldots, N\}$ for k_t in the case where resampling occurs. The use of a special variable $x_{0:T}^*$ highlights the fact that the marginal density of $x_{0:T}^{\mathbf{k}} = x_{0:T}^*$ given $\mathbf{k}$ is π. The conditional SMC algorithm corresponding to sampling from the

density $\bar{q}(\mathbf{k}, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)} | x_{0:T}^*)$ is given in Algorithm 5.3.

Algorithm 5.3 Conditional SMC Algorithm

1. At time $t = 0$.
   - Sample k_0 uniformly on $\{1, .., N\}$.
   - For $i \in \{1, \ldots, N\} \setminus \{k_0\}$, sample $x_0^{(i)} \sim g_0(\cdot)$ and set $x_0^{(k_0)} = x_0^*$.
   - For $i = 1, \ldots, N$, evaluate the importance weights:

$$w_0(x_0^{(i)}) = \frac{\gamma_0(x_0^{(i)})}{g_0(x_0^{(i)})}.$$

2. For times $t = 1, \ldots, T$.
   - With probability $\xi(W_{t-1}^{(1:N)})$, sample k_t uniformly on $\{1, \ldots, N\}$, set $A_{t-1}^{(k_t)} = k_{t-1}$, sample $A_{t-1}^{(-k_t)} \sim \mathcal{M}(\cdot | N-1, W_{t-1}^{(1:N)})$ and set $r_{t-1} = 1$, $\omega_t^{(1:N)} = (1, \ldots, 1)$.
     Otherwise, set $k_t = k_{t-1}$, $A_{t-1}^{(1:N)} = (1, \ldots, N)$, $r_{t-1} = 0$ and $\omega_t^{(1:N)} = w_{t-1}^{(1:N)}$.
   - For $i \in \{1, \ldots, N\} \setminus \{k_t\}$, sample $x_t^{(i)} \sim g_t(\cdot | x_{0:t-1}^{C_{t-1}^i})$ and set $x_t^{(k_t)} = x_t^*$.
   - For $i = 1, \ldots, N$, evaluate the importance weights:

$$w_t(x_{0:t-1}^{C_{t-1}^i}, x_t^{(i)}) = \omega_t^{(i)} \frac{\gamma_t(x_{0:t-1}^{C_{t-1}^i}, x_t^{(i)})}{\gamma_{t-1}(x_{0:t-1}^{C_{t-1}^i}) g_t(x_t^{(i)} | x_{0:t-1}^{C_{t-1}^i})}.$$

With the target and proposal densities defined as above, and defining the set of resampling times by $\mathcal{R} = \{t \in \{0, \ldots, T-1\} : r_t = 1\}$ their ratio is given by

$$\frac{\tilde{\pi}^N(\mathbf{k}, x_{0:T}^*, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)})}{q(\mathbf{k}, x_{0:T}^*, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)})} = \frac{1}{Z_T} \prod_{t \in \mathcal{R} \cup \{T\}} \left[\frac{1}{N} \sum_{j=1}^{N} w_t^{(j)} \right]. \tag{5.4}$$

In fact, the second approach to implementing conditional resampling can be explained more easily, but, at least theoretically, requires implementation of a non-standard algorithm. In this case, recall that we set $A_{t-1}^{(1:N)}$ to be a uniform random permutation of $(1, \ldots, N)$ if resampling does not occur. As such, there is no conflict with having k_t be uniformly distributed on $\{1, \ldots, N\}$ at every time $t \in \{0, \ldots, T\}$. The resampling weights, which are what appear in the acceptance ratio and are in this case distinct from the importance weights of the particles, satisfy $\tilde{w}_t^{(j)} = 1$ for

all $j \in \{1, \ldots, N\}$ at times t when resampling is not performed. As such,

$$\frac{1}{N} \sum_{j=1}^{N} \tilde{w}_t^{(j)} = 1$$

at these times and one recovers the same expression as (5.4).

An important observation for the material in Section 5.2.2 is that the representation of the target as $\tilde{\pi}^N(\mathbf{k}, x_{0:T}^*, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)})$ makes it clear that one can define a Gibbs move that samples $(\mathbf{k}, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)})$ conditional on $x_{0:T}^*$ under $\tilde{\pi}^N$, where we emphasize that $(\mathbf{k}, \mathbf{r})$ in particular is not fixed. This is important because in the material in that section, one will want to, in some cases, produce samples according to this conditional density. The ability to sample $(\mathbf{k}, \mathbf{r})$ along with the remaining variables allows for the number and value of resampling times to be different when running a conditional SMC algorithm that corresponds to sampling from the distribution with density $\bar{q}(\mathbf{k}, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)} | x_{0:T}^*)$.

5.2.1.2 Importance Sampling

The above derivation of a joint density and corresponding extended target density is, as yet, distinct from the earlier work in this chapter, in that it is not part of an MCMC algorithm. It is worth noting that the above derivation of an extended target density for samples produced by SMC, and which summarizes some of the material in Andrieu et al. (2010), is motivated primarily by our inability to evaluate the marginal density $q(x_{0:T}^*)$, i.e. the density of the selected particle at time T. A further note is that one can view particle methods in general as a specification of a general mechanism by which one can produce dependent proposals through a combination of importance sampling and resampling, the particle interaction step that is characteristic of the methodology.

A simple use of the above extended density is in importance sampling, where the unnormalized importance weight associated with each time T particle i from an

SMC algorithm is given by

$$\prod_{t\in\mathcal{R}\cup\{T\}}\left[\frac{1}{N}\sum_{j=1}^{N}w_t^{(j)}\right]\frac{w_T^{(i)}}{\sum_{i=1}^{N}w_T^{(j)}}.$$

This weighting is common practice for samples from a single algorithm, but the above shows that it is possible to include particles from different SMC algorithms targeting the same density π. This is of limited use since one will typically obtain lower variance estimates of expectations with respect to π by running a single algorithm with larger N, due to the interaction between a larger set of particles in this case.

5.2.2 Particle Markov Chain Monte Carlo

A major advantage of the ability to weight samples with respect to π is in allowing for the use of SMC within more general frameworks, such as MCMC (Andrieu et al. 2010) and even SMC (Chopin et al. 2011). We turn now to the use of such samples within our framework for designing reversible MCMC kernels, in which the particle independent Metropolis-Hastings (PIMH) algorithm, particle marginal Metropolis-Hastings (PMMH) and particle Gibbs algorithms of (Andrieu et al. 2010) can be seen as utilizing advanced sequential proposals, the latter two within a pseudo-marginal framework.

In order to distinguish between particle histories for variables denoted generally by x and those denoted by z, we introduce the notation (x) and (z) respectively to make this clear. For clarity, we replace the particular form of the importance weight with the symbol $\hat{Z}_T$, since the weight is an unbiased estimate $Z_T = \int \gamma(x_{0:T})dx_{0:T}$, the normalizing constant of the density $\gamma(\cdot) \propto \pi(\cdot)$. In cases where θ is a parameter, we use the symbol $\hat{Z}_T(\theta', z)$ to denote the importance weight associated with an SMC algorithm with static parameter θ' for the variables denoted generally by z.

5.2.2.1 Particle "Independent" Metropolis-Hastings

Consider the target

$$\begin{aligned}&\tilde{\pi}^N(\mathbf{k}, \mathbf{l}, x^*_{0:T}, z^*_{0:T}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x), z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z)) \\ &\quad = \pi(x^*_{0:T})q(\mathbf{l}, z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z))\delta_{z^{\mathbf{l}}_{0:T}}(z^*_{0:T})\bar{q}(\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x)|x^*_{0:T})\end{aligned}$$

Due to the results in the previous section, the swap move

$$\begin{aligned}&(\mathbf{k}, \mathbf{l}, x^*_{0:T}, z^*_{0:T}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x), z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z)) \\ &\qquad \rightarrow (\mathbf{l}, \mathbf{k}, z^*_{0:T}, x^*_{0:T}, z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z), x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x))\end{aligned}$$

has acceptance probability $\min\left\{1, \hat{Z}_T(z)/\hat{Z}_T(x)\right\}$.

The particle independent Metropolis-Hastings (PIMH) method of Andrieu et al. (2010), samples $(\mathbf{l}, z^*_{0:T}, z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z)) \sim \tilde{\pi}^N(\cdot|\mathbf{k}, x^*_{0:T}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x))$ before attempting the swap move. However, in the same manner as that described in section 4.4, it is possible to additionally sample $(\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x)) \sim \bar{q}(\cdot|x^*_{0:T})$ before attempting the swap. In this case, the state $x^*_{0:T}$ at each iteration of the MCMC algorithm defines a Markov chain.

Another potential improvement in some applications is that q can condition on $x^*_{0:T}$ and $\bar{q}$ can condition on $z^*_{0:T}$. This can be seen by rewriting the extended target density as

$$\begin{aligned}&\tilde{\pi}^N(\mathbf{k}, \mathbf{l}, x^*_{0:T}, z^*_{0:T}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x), z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z)) \\ &\qquad = \pi(x^*_{0:T})q(\mathbf{l}, z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z)|x^*_{0:T})\delta_{z^{\mathbf{l}}_{0:T}}(z^*_{0:T}) \\ &\qquad \times \bar{q}(\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x)|x^*_{0:T}, z^{\mathbf{l}}_{0:T})\end{aligned}$$

The implementation of such a scheme will usually involve the proposal densities within the SMC algorithm $\{g_t\}_{t=0}^T$ conditioning on $x^*_{0:T}$ and $z^*_{0:T}$ respectively, and in this case the form of the acceptance ratio is unchanged if $(\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x))$

is sampled. More specifically, one can extend (5.2) via

$$q(\mathbf{l}, z^*_{0:T}, z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z)|x^*_{0:T}) = \left[\prod_{i=1}^{N} g^*_0(z^{(i)}_0|x^*_{0:T})\right]\left\{\prod_{t=1}^{T} q(r_{t-1}, A^{(1:N)}_{t-1}|W^{(1:N)}_{t-1})\prod_{i=1}^{N} g^*_t(z^{(i)}_t|z^{C^i_{t-1}}_{0:t-1}, x^*_{0:T})\right\} \times \frac{w_T^{(l_T)}}{\sum_{i=1}^{N} w_T^{(i)}}\delta_{(C^{l_T}_{T-1})}(l_{0:T-1})\delta_{z^{\mathbf{l}}_{0:T}}(z^*_{0:T}) \quad (5.5)$$

and (5.3) via

$$\bar{q}(\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x)|x^*_{0:T}, z^*_{0:T}) = \frac{1}{N}\delta_{x^*_0}(x^{(k_0)}_0)\left[\prod_{i=1, i\neq k_0}^{N} g^*_0(x^{(i)}_0|z^*_{0:T})\right] \times \left\{\prod_{t=1}^{T} \bar{q}(k_t, r_{t-1}, A^{(1:N)}_{t-1}|W^{(1:N)}_{t-1}, k_{t-1})\delta_{x^*_t}(x^{(k_t)}_t)\prod_{i=1, i\neq k_t}^{N} g^*_t(x^{(i)}_t|x^{C^i_{t-1}}_{0:t-1}, z^*_{0:T})\right\}. \quad (5.6)$$

This conditioning could be especially helpful when the proposals $\{g_t\}_{t=0}^T$ on their own have difficulty targeting π. More specifically, it can be the case that

$$\bar{g}_t(z_{0:t}) = \pi_{t-1}(z_{0:t-1})g_t(z_t|z_{0:t-1})$$

is a poor importance density for π_t, leading to significant degeneracy of the weights at time t, but that if $x^*_{0:T} \sim \pi_T(\cdot)$ then one can build a better importance density for π_t as

$$\bar{g}^*_t(z_{0:t}|x^*_{0:T}) = \pi_{t-1}(z_{0:t-1})g^*_t(z_t|z_{0:t-1}, x^*_{0:T}).$$

A description of the PIMH algorithm is given in Algorithm 5.4, where the densities q and $\bar{q}$ are given by (5.5) and (5.6) respectively. The extra terms $x^*_{0:T}$ in q and $z^*_{0:T}$ in $\bar{q}$ are included to indicate that it is possible to condition the proposals $g_{0:T}$ on these terms. Indeed, step 3 of the algorithm is not mandatory if the proposals do not condition on these values. A possible remark is that the proposals used in the resulting PIMH algorithm are no longer independent of the current state of the chain. However, it seems useful to consider the extra conditioning as only a minor extension of the existing algorithm.

Algorithm 5.4 Particle "Independent" Metropolis-Hastings

- At iteration t:
  1. Set $x^*_{0:T} = x^*_{0:T,t-1}$.
  2. Sample $\mathbf{l}, z^*_{0:T}, z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z) \sim q(\cdot|x^*_{0:T})$
  3. Sample $\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x) \sim \bar{q}(\cdot|x^*_{0:T}, z^*_{0:T})$
  4. With probability $\min\left\{1, \hat{Z}_T(z)/\hat{Z}_T(x)\right\}$, set $x^*_{0:T,t} = z^*_{0:T}$. Otherwise, set $x^*_{0:T,t} = x^*_{0:T}$.

5.2.2.2 Particle Marginal Metropolis-Hastings

We now consider inference on a parameter θ where $\pi(\theta)$ is not computationally available. This is often because $\pi(\theta) = f(\theta|\mathbf{y}) \propto f(\mathbf{y}|\theta)f(\theta)$ but $f(\mathbf{y}|\theta)$ cannot be computed analytically. When we have access to some density $f(\mathbf{y}, \mathbf{x}|\theta)$ such that $\int f(\mathbf{y}, \mathbf{x}|\theta)d\mathbf{x} = f(\mathbf{y}|\theta)$, then this is a pseudo-marginal problem of the type described in Section 4.4 and we can define

$$\pi(\theta, \mathbf{x}) = f(\theta, \mathbf{x}|\mathbf{y}) = \frac{f(\mathbf{y}, \mathbf{x}|\theta)f(\theta)}{f(\mathbf{y})}.$$

SMC in this setting can be seen as a method by which one constructs dependent proposals with density close to $\pi(\mathbf{x}|\theta)$ in a principled fashion, as a general approach to constructing schemes with the form of Algorithm 4.8. This involves the definition of T auxiliary densities $\{\pi_t\}_{t=0}^{T-1}$ and $T+1$ proposal densities $\{g_t\}_{t=0}^{T}$, where the strength of the methodology is that judicious but simple choices for these densities and large enough N lead to the weighted time T samples providing a good approximation to the distribution with density $\pi(\mathbf{x}|\theta)$.

Given that we want to use SMC to target $\pi(\mathbf{x}|\theta)$, there are a number of possible ways to select the sequence of auxiliary densities $\{\pi_t\}_{t=0}^{T-1}$. In state-space models with $\mathbf{y} = y_{1:T}$, it is common to use $\pi_t(x_{0:t}|\theta) = f(x_{0:t}|y_{1:t}, \theta)$. In static models, we typically define $\pi_T(x_{0:T}|\theta) = \pi(x_T|\theta)\prod_{j=0}^{T-1} L_j(x_{j+1}, x_j)$, where the L_j's are a set of

auxiliary backwards kernels, and one might choose instead

$$\pi_t(x_{0:t}|\theta) = \frac{\pi(x_t|\theta)^{\beta_t}}{\int \pi(z|\theta)^{\beta_t} dz} \prod_{j=0}^{t-1} L_j(x_{j+1}, x_j)$$

as is done in Section 2.4.1 or

$$\pi_t(x_{0:t}|\theta_{0:t}) = \pi(x_t|\theta_t) \prod_{j=0}^{t-1} L_j(x_{j+1}, x_j)$$

as is done in Section 3.5. Many other schemes are possible.

To incorporate SMC proposals in a pseudo-marginal MCMC kernel, we can consider the target

$$\begin{aligned}\tilde{\pi}^N(\mathbf{k}, \mathbf{l}, \theta, \theta', x^*_{0:T}, z^*_{0:T}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x), z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z)) \\ = \pi(\theta, x^*_{0:T}) g(\theta'|\theta) q(\mathbf{l}, z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z)|\theta', x^*_{0:T}) \\ \times \delta_{z^{\mathbf{l}}_{0:T}}(z^*_{0:T}) \bar{q}(\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x)|\theta, x^*_{0:T}, z^*_{0:T})\end{aligned}$$

The swap move

$$\begin{aligned}(\mathbf{k}, \mathbf{l}, \theta, \theta', x^*_{0:T}, z^*_{0:T}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x), z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z)) \\ \rightarrow (\mathbf{l}, \mathbf{k}, \theta', \theta, z^*_{0:T}, x^*_{0:T}, z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z), x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x))\end{aligned}$$

has acceptance probability

$$\min\left\{1, \frac{f(\theta')g(\theta|\theta')\hat{Z}_T(\theta', z)}{f(\theta)g(\theta'|\theta)\hat{Z}_T(\theta, x)}\right\}.$$

The potential improvements noted in the previous section are also suggested here, namely that one can both sample $(\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x))$ conditional on $x^*_{0:T}$ and also let q and $\bar{q}$ condition on $x^*_{0:T}$ and $z^*_{0:T}$, respectively. A description of the PMMH scheme is given in Algorithm 5.5, where step 4 is not mandatory if q and $\bar{q}$ do not condition on $x^*_{0:T}$ and $z^*_{0:T}$ respectively.

Algorithm 5.5 Particle Marginal Metropolis-Hastings

- At iteration t:
  1. Set $\theta = \theta_{t-1}$ and $x^*_{0:T} = x^*_{0:T,t-1}$.
  2. Sample $\theta' \sim g(\cdot|\theta)$.
  3. Sample $\mathbf{l}, z^*_{0:T}, z^{(1:N)}_{0:T}, \mathbf{r}(z), A^{(1:N)}_{0:T-1}(z) \sim q(\cdot|\theta', x^*_{0:T})$
  4. Sample $\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}(x), A^{(1:N)}_{0:T-1}(x) \sim \bar{q}(\cdot|\theta, x^*_{0:T}, z^*_{0:T})$
  5. With probability

$$\min\left\{1, \frac{f(\theta')g(\theta|\theta')\hat{Z}_T(\theta', z)}{f(\theta)g(\theta'|\theta)\hat{Z}_T(\theta, x)}\right\},$$

  set $(\theta_t, x^*_{0:T,t}) = (\theta', z^*_{0:T})$.
  Otherwise, set $(\theta_t, x^*_{0:T,t}) = (\theta, x^*_{0:T})$.

5.2.2.3 Particle Gibbs

For the particle Gibbs algorithm, one can consider the following extended target

$$\tilde{\pi}^N(\mathbf{k}, \theta, x^*_{0:T}, x^{(1:N)}_{0:T}, \mathbf{r}, A^{(1:N)}_{0:T-1}) = \pi(\theta, x^*_{0:T})\bar{q}(\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}, A^{(1:N)}_{0:T-1}|x^*_{0:T}, \theta)$$

The algorithm is defined by first sampling $\theta \sim \pi(\cdot|x^*_{0:T})$ and $\mathbf{k}, x^{(1:N)}_{0:T}, \mathbf{r}, A^{(1:N)}_{0:T-1} \sim \bar{q}(\cdot|x^*_{0:T}, \theta)$. Then one samples $\mathbf{k}, x^*_{0:T} \sim \tilde{\pi}^N(\cdot|\theta, x^{(1:N)}_{0:T}, \mathbf{r}, A^{(1:N)}_{0:T-1})$ via

$$\mathbb{P}[K = \mathbf{k}|\theta, x^{(1:N)}_{0:T}, \mathbf{r}, A^{(1:N)}_{0:T-1}] = \frac{\tilde{\pi}^N((C^{k_T}_{T-1}, k_T), \theta, (x^{C^{k_T}_{T-1}}_{0:T-1}, x^{(k_T)}_T), x^{(1:N)}_{0:T}, \mathbf{r}, A^{(1:N)}_{0:T-1})}{\sum_{j=1}^N \tilde{\pi}^N((C^j_{T-1}, j), \theta, (x^{C^j_{T-1}}_{0:T-1}, x^{(j)}_T)x^{(1:N)}_{0:T}, \mathbf{r}, A^{(1:N)}_{0:T-1})}$$

where division of both numerator and denominator by

$$q(x^{(1:N)}_{0:T}, \mathbf{r}, A^{(1:N)}_{0:T-1}|\theta) = \sum_{k_T=1}^N \delta_{C^{k_T}_{T-1}}(k_{0:T-1})\delta_{x^{\mathbf{k}}_{0:T}}(x^*_{0:T})q(\mathbf{k}, x^*_{0:T}, x^{(1:N)}_{0:T}, \mathbf{r}, A^{(1:N)}_{0:T-1}|\theta)$$

gives

$$\mathbb{P}[K = \mathbf{k}|\theta, x^{(1:N)}_{0:T}, \mathbf{r}, A^{(1:N)}_{0:T-1}] = \frac{w^{(k_T)}_T}{\sum_{j=1}^N w^{(j)}_T}\delta_{C^{k_T}_{T-1}}(k_{0:T-1})$$

and one can set $x^*_{0:T} = x^{\mathbf{k}}_{0:T}$ upon sampling $\mathbf{k}$.

A potential improvement is that instead of Gibbs sampling $\mathbf{k}$, one could propose a swap where we choose $L_T = l_T$ with probability proportional to $w_T^{(l)}$ for $l_T \in \{1, \ldots, N\} \setminus \{k_T\}$ and zero otherwise, and set $l_{0:T-1} = C_{T-1}^{l_T}$. The acceptance probability for the swap move $(\mathbf{k}, \theta, x_{0:T}^*, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)}) \to (\mathbf{l}, \theta, x_{0:T}^{\mathbf{l}}, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)})$ is then

$$\min\left\{1, \frac{\sum_{j=1, j\neq k_T}^N w_T^{(j)}}{\sum_{j=1, j\neq l_T}^N w_T^{(j)}}\right\}.$$

Algorithm 5.6 Particle Gibbs

- At iteration t:
  1. Set $x_{0:T}^* = x_{0:T,t-1}^*$.
  2. Sample $\theta_t \sim \pi(\cdot|x_{0:T}^*)$.
  3. Sample $\mathbf{k}, x_{0:T}^{(1:N)}, \mathbf{r}, A_{0:T-1}^{(1:N)} \sim \bar{q}(\cdot|\theta_t, x_{0:T}^*)$.
  4. Sample $\mathbf{l} = (C_{T-1}^{l_T}, l_T)$ with probability $\frac{w_T^{(l_T)}}{\sum_{j=1, j\neq k_T}^N w_T^{(j)}}$
  5. With probability
  $$\min\left\{1, \frac{\sum_{j=1, j\neq k_T}^N w_T^{(j)}}{\sum_{j=1, j\neq l_T}^N w_T^{(j)}}\right\},$$
  set $x_{0:T,t}^* = (x_{0:T-1}^{C_{T-1}^{l_T}}, x_T^{(l_T)})$.
  Otherwise, set $x_{0:T,t}^* = x_{0:T}^*$.

5.2.2.4 Remark

One of the characteristic features of the algorithms presented is that the state space of the Markov chain, when we sample according to $\bar{q}$, can be collapsed to $\Theta \times \mathcal{X}^{T+1}$, i.e. it need not include the auxiliary variables introduced in both q and $\bar{q}$ since $(\theta, x_{0:T}^*)$ defines a Markov chain. A further reduction of the state space is possible in some cases, particularly when the target distribution can be factorized as $\pi_T(\theta, x_{0:T}) = \pi(\theta)\pi(x_T|\theta)\prod_{t=1}^T L_{t-1}(x_t, x_{t-1}; \theta)$. If one can sample according to the backwards kernel L_{t-1} for each $t \in \{1, \ldots, T\}$, this can be incorporated into any of the above algorithms as a preliminary step. As an example, this is possible in the somewhat generic SMC sampler situation where K_t is a reversible MCMC kernel with invariant distribution $\pi_t(x_t|\theta)$, in which case $L_{t-1} = K_t$. By sampling $x_{0:T-1}^* \sim \prod_{t=1}^T L_{t-1}(x_t, x_{t-1}; \theta)$ with $x_T = x_T^*$, one can collapse the state space of the

Markov chain to $\Theta \times \mathcal{X}$.

5.2.3 Adapting N

We now consider the adaptation of $\mathbf{N} = N_{0:T}$ in SMC, where we now adapt N_t, the number of particles at each time t of the SMC algorithm. In this case, it makes sense for resampling to occur at each stage, although this operation is now slightly different to what has occurred before. In particular, we do not want to require $N_t \geq N_{t-1}$ so some form of resampling occurs at each time. The primary motivation for such a scheme is that the SMC methodology can allocate different amounts of computational resources when approximating different auxiliary target densities. If the adaptation is successful, one could obtain an approximation of π_T such that an equally good approximation using fixed N would be considerably more costly.

We will introduce the random variables $\mathbf{u}_{0:T}$ and $\mathbf{v}_{0:T}$, as well as a mapping function $\psi_{N,k}$ as in Section 4.5.2. Given $\mathbf{N} = N_{0:T}$ and $\mathbf{N}' = N'_{0:T}$, we have $\mathbf{u}_{0:T} = (u_0^{(1:N_0)}, \ldots, u_T^{(1:N_T)})$ and $\mathbf{v}_{0:T} = (v_0^{(1:N'_0)}, \ldots, v_T^{(1:N'_T)})$. We denote $\mathbf{x}_{0:T}$ and $\mathbf{z}_{0:T}$ analogously, and given $\mathbf{k} \in \{1, \ldots, N_0\} \times \cdots \times \{1, \ldots, N_T\}$, $\mathbf{l} \in \{1, \ldots, N'_0\} \times \cdots \times \{1, \ldots, N'_T\}$, $\mathbf{x}_{0:T}$ and $\mathbf{z}_{0:T}$ are defined by

$$x_t^{(1:N_t)} = \psi^{-1}_{(N_t,k_t)}(u_t^{(1:N_t)})$$

and

$$v_t^{(1:N'_t)} = \psi_{(N'_t,l_t)}(z_t^{(1:N'_t)})$$

for every $t \in \{0, \ldots, T\}$. We also need to introduce a random variable $\bar{\mathbf{A}}_{0:T-1}$ to denote the reordered ancestries. This random variable satisfies

$$\bar{A}_{t-1}^{(1:N'_t)} = \psi_{(N'_t,k_t)}(A_{t-1}^{(1:N'_t)}).$$

In the following, we focus on the case where the stopping probabilities β_j depend only on the weights at time j, but this restriction is not necessary. In general, β_j could depend on any random variables sampled up to and including time j.

We can write an SMC proposal density as

$$q(\mathbf{N}, \mathbf{k}, \mathbf{x}_{0:T}, \mathbf{A}_{0:T-1}, \mathbf{u}_{0:T}, \bar{\mathbf{A}}_{0:T-1}) =$$
$$q_0(N_0, x_0^{(1:N_0)}) \prod_{t=1}^{T} q_t(N_t, x_t^{(1:N_t)}, A_{t-1}^{(1:N_t)} | \mathbf{x}_{0:t-1}, \mathbf{A}_{0:t-2})$$
$$\times \frac{w_T^{(k_T)}}{\sum_{i=1}^{N} w_T^{(i)}} \delta_{(C_{T-1}^{k_T})}(\mathbf{k}_{0:T-1}) \prod_{t=0}^{T} \delta_{\psi_{N_t,k_t}(x_t^{(1:N_t)})}(u_t^{(1:N_t)}) \prod_{t=1}^{T} \delta_{\psi_{N_t,k_t}(A_{t-1}^{(1:N_t)})}(\bar{A}_{t-1}^{(1:N_t)})$$

where

$$q_0(N_0, x_0^{(1:N_0)}) = \beta_{N_0}(w_0^{(1:N_0)}) g_0(x_0^{(N_0)}) \prod_{j=1}^{N_0-1} [1 - \beta_j(w_0^{(1:j)})] g_0(x_0^{(j)})$$

and

$$q_t(N_t, x_t^{(1:N_t)}, A_{t-1}^{(1:N_t)} | \mathbf{x}_{0:t-1}, \mathbf{A}_{0:t-2}) =$$
$$\beta_{N_t}(w_t^{(1:N_t)}) \mathcal{M}(A_{t-1}^{(N_t)} | 1, W_{t-1}^{(1:N_{t-1})}) g_t(x_t^{(N_t)} | x_{0:t-1}^{C_{t-1}^{N_t}})$$
$$\times \prod_{j=1}^{N_t-1} [1 - \beta_t(w_t^{(1:j)})] \mathcal{M}(A_{t-1}^{(j)} | 1, W_{t-1}^{(1:N_{t-1})}) g_t(x_t^{(j)} | x_{0:t-1}^{C_{t-1}^{j}}).$$

The extended target density can be written as

$$\tilde{\pi}(\mathbf{N}, \mathbf{k}, \mathbf{x}_{0:T}, \mathbf{A}_{0:T-1}, \mathbf{u}_{0:T}, \bar{\mathbf{A}}_{0:T-1}) = \pi(u_{0:T}^{\mathbf{1}}) \delta_{\mathbf{1}}(\bar{A}_{0:T-1}^{(1)}) \bar{q}(\mathbf{N}, \mathbf{k}, u_{0:T}^{-\mathbf{1}}, \bar{A}_{0:T-1}^{-\mathbf{1}} | u_{0:T}^{\mathbf{1}})$$
$$\times \prod_{t=0}^{T} \delta_{\psi_{N_t,k_t}^{-1}(u_t^{(1:N_t)})}(x_t^{(1:N_t)}) \times \prod_{t=1}^{T} \delta_{\psi_{N_t,k_t}^{-1}(\bar{A}_{t-1}^{(1:N_t)})}(A_{t-1}^{(1:N_t)})$$

where

$$\bar{q}(\mathbf{N}, \mathbf{k}, u_{0:T}^{-\mathbf{1}}, \bar{A}_{0:T-1}^{-\mathbf{1}} | u_{0:T}^{\mathbf{1}}) = \bar{q}_0(N_0, u_0^{(2:N_0)} | u_0^{(1)}) \bar{q}(k_0 | N_0, u_0^{(1:N_0)})$$
$$\times \prod_{t=1}^{T} \bar{q}_t(N_t, u_t^{(2:N_t)}, \bar{A}_{t-1}^{(2:N_t)} | \mathbf{u}_{0:t-1}, \mathbf{A}_{0:t-2}, u_t^{(1)}) \bar{q}(k_t | N_t, u_t^{(1:N_t)}),$$

$$\bar{q}_0(N_0, u_0^{(2:N_0)}|u_0^{(1)}) =$$
$$\bar{\beta}_{N_0}(\bar{w}_0^{(1:N_0)})g_0(u_0^{(N_0)})\left[\prod_{j=2}^{N_0-1}[1-\bar{\beta}_j(\bar{w}_0^{(1:j)})]g_0(u_0^{(j)})\right][1-\bar{\beta}_1(\bar{w}_0^{(1)})]$$

and

$$\bar{q}_t(N_t, u_t^{(2:N_t)}, \bar{A}_{t-1}^{(2:N_t)}|\mathbf{u}_{0:t-1}, \mathbf{A}_{0:t-2}, u_t^{(1)}) =$$
$$\bar{\beta}_{N_t}(\bar{w}_t^{(1:N_t)})\mathcal{M}(\bar{A}_{t-1}^{(N_t)}|1, \bar{W}_{t-1}^{(1:N_{t-1})})g_t(u_t^{(N_t)}|u_{0:t-1}^{C_{t-1}^{N_t}})$$
$$\times\left[\prod_{j=2}^{N_t-1}[1-\bar{\beta}_t(\bar{w}_t^{(1:j)})]\mathcal{M}(A_{t-1}^{(j)}|1, \bar{W}_{t-1}^{(1:N_{t-1})})g_t(u_t^{(j)}|u_{0:t-1}^{C_{t-1}^{j}})\right][1-\bar{\beta}_1(\bar{w}_t^{(1)})].$$

with the weights in $\bar{q}$ denoted by $\bar{w}$ and $\bar{W}$ respectively to make clear that they are the weights associated with the variables $\mathbf{u}_{0:T}$ and not $\mathbf{x}_{0:T}$.

Despite the rather tedious description of the proposal and extended target densities, the importance weight

$$\frac{\tilde{\pi}(\mathbf{N}, \mathbf{k}, \mathbf{x}_{0:T}, \mathbf{A}_{0:T-1}, \mathbf{u}_{0:T}, \bar{\mathbf{A}}_{0:T-1})}{q(\mathbf{N}, \mathbf{k}, \mathbf{x}_{0:T}, \mathbf{A}_{0:T-1}, \mathbf{u}_{0:T}, \bar{\mathbf{A}}_{0:T-1})}$$

can take a simple form, depending on the choice of the β_j's, $\bar{\beta}_j$'s and $\bar{q}(k_t|N_t, u_t^{(1:N_t)})$. In a simple scenario, one might have $\bar{q}(k_t|N_t, u_t^{(1:N_t)}) = 1/N_t$. In this case, the weight is given by

$$\frac{\tilde{\pi}(\mathbf{N}, \mathbf{k}, \mathbf{x}_{0:T}, \mathbf{A}_{0:T-1}, \mathbf{u}_{0:T}, \bar{\mathbf{A}}_{0:T-1})}{q(\mathbf{N}, \mathbf{k}, \mathbf{x}_{0:T}, \mathbf{A}_{0:T-1}, \mathbf{u}_{0:T}, \bar{\mathbf{A}}_{0:T-1})} = \frac{\rho_2}{Z_T}\prod_{t=0}^{T}\left[\frac{1}{N_t}\sum_{j=1}^{N} w_t^{(j)}\right]$$

where

$$\rho_2 = \prod_{t=0}^{T}\frac{\bar{\beta}_{N_t}(\bar{w}_t^{(1:N_t)})\prod_{j=1}^{N_t-1}[1-\bar{\beta}_j(\bar{w}_t^{(1:j)})]}{\beta_{N_t}(w_t^{(1:N_t)})\prod_{j=1}^{N_t-1}[1-\beta_j(w_t^{(1:j)})]}$$

Here, ρ_2 denotes the correction needed to make the estimate of $\hat{Z}_T$ unbiased, which is slightly different from the meaning of ρ_2 in Section 4.5.2. It is included in this fashion here to make explicit the fact that this estimate remains unbiased when $\mathbf{N}$ is adapted.

Algorithm 5.7 Adaptive SMC Algorithm

1. At time $t = 0$.
   - For $j = 1, \ldots$
     - Sample $x_0^{(j)} \sim g_0(\cdot)$.
     - Evaluate the importance weight:
     $$w_0^{(j)} = w_0(x_0^{(j)}) = \frac{\gamma_0(x_0^{(j)})}{g_0(x_0^{(j)})}.$$
     - With probability $\beta_j(w_0^{(1:j)})$, set $N_0 = j$ and go to time $t + 1$.
2. For times $t = 1, \ldots, T$.
   - For $j = 1, \ldots$
     - Sample $A_{t-1}^{(j)} \sim \mathcal{M}(\cdot|1, W_{t-1}^{(1:N_{t-1})})$.
     - Sample $x_t^{(j)} \sim g_t(\cdot|x_{0:t-1}^{C_{t-1}^j})$.
     - Evaluate the importance weight:
     $$w_t^{(j)} = w_t(x_{0:t-1}^{C_{t-1}^i}, x_t^{(j)}) = \frac{\gamma_t(x_{0:t-1}^{C_{t-1}^j}, x_t^{(j)})}{\gamma_{t-1}(x_{0:t-1}^{C_{t-1}^j}) g_t(x_t^{(j)}|x_{0:t-1}^{C_{t-1}^j})}.$$
     - With probability $\beta_j(w_t^{(1:j)})$, set $N_t = j$ and go to time $t + 1$.
3. - Sample k_T with probability $\frac{w_T^{(k_T)}}{\sum_{i=1}^{N_T} w_T^{(i)}}$ and set $k_{0:T-1} = C_{T-1}^{k_T}$.
   - Evaluate
   $$\rho_2 = \prod_{t=0}^{T} \frac{\bar{\beta}_{N_t}(\bar{w}_t^{(1:N_t)}) \prod_{j=1}^{N_t-1}[1 - \bar{\beta}_j(\bar{w}_t^{(1:j)})]}{\beta_{N_t}(w_t^{(1:N_t)}) \prod_{j=1}^{N_t-1}[1 - \beta_j(w_t^{(1:j)})]}$$
   where $\bar{w}_t^{(1:N_t)} = \psi_{N_t,k_t}(w_t^{(1:N_t)})$.
   - Evaluate
   $$\hat{Z}_T = \rho_2 \prod_{t=0}^{T} \left[\frac{1}{N_t} \sum_{j=1}^{N_t} w_t^{(j)} \right].$$

5.2.3.1 Adaptive Particle Markov Chain Monte Carlo

Adaptive versions of the particle Markov chain Monte Carlo algorithms are relatively straightforward extensions to the existing methods, generally requiring only the specification of stopping rule for N and N' and the additional computation of the correction factors ρ_2 for both the forward filter and the backward filter. For simplicity, we ignore the ability for q and $\bar{q}$ to condition on $u^{(1)}$ and $v^{(1)}$ respectively, but this remains a possibility. Adaptive schemes are given in Algorithms 5.8-5.10. For the adaptive particle Gibbs sampler, it would be theoretically advantageous to incorporate ρ_2, which encodes the discrepancy between the density of $\mathbf{N}$ when $u^{\mathbf{1}}_{0:T} = x^{\mathbf{k}}_{0:T}$ and when $u^{\mathbf{1}}_{0:T} = x^{\mathbf{l}}_{0:T}$, in the selection of $\mathbf{l}$. However, this would involve the computation of ρ_2 for all N_T possible values of l_T, which may be excessive in some cases.

Algorithm 5.8 Adaptive Particle Independent Metropolis-Hastings

- At iteration t:
  1. Set $u^{(1)}_{0:T} = u^{(1)}_{0:T,t-1}$.
  2. Sample $\mathbf{N}', \mathbf{l}, \mathbf{z}_{0:T}, \mathbf{A}_{0:T-1}(z), \mathbf{v}_{0:T}, \bar{\mathbf{A}}_{0:T-1}(z) \sim q(\cdot)$.
  3. Sample $\mathbf{N}, \mathbf{k}, \mathbf{x}_{0:T}, \mathbf{A}_{0:T-1}(x), \mathbf{u}^{-\mathbf{1}}_{0:T}, \bar{\mathbf{A}}^{-\mathbf{1}}_{0:T-1}(x) \sim \bar{q}(\cdot|u^{(1)}_{0:T})$.
  4. With probability $\min\left\{1, \frac{\hat{Z}_T(z)}{\hat{Z}_T(x)}\right\}$, set $u^{(1)}_{0:T,t} = v^{(1)}_{0:T}$.
     Otherwise, set $u^{(1)}_{0:T,t} = u^{(1)}_{0:T}$.

5.2.3.2 Remarks

The effect of the adaptation of $\mathbf{N}$ in an SMC framework can be complex to analyze. The choice of stopping rules β_j and $\bar{\beta}_j$ is not obvious and can have a large impact on the effectiveness of resulting algorithms. A possible choice is to use an effective sample size (ESS) threshold to ensure that there is diversity at each time. However, while promising and effective in some circumstances, there are two major issues with this criterion. The first is that it is not clear what choice of threshold is good. In fact, a low choice of threshold does not necessarily decrease the total number of particles sampled for all time steps since low diversity can drastically affect the

Algorithm 5.9 Adaptive Particle Marginal Metropolis-Hastings

- At iteration t:
  1. Set $\theta = \theta_{t-1}$ and $u_{0:T}^{(1)} = u_{0:T,t-1}^{(1)}$.
  2. Sample $\theta' \sim g(\cdot|\theta)$.
  3. Sample $\mathbf{N}', \mathbf{l}, \mathbf{z}_{0:T}, \mathbf{A}_{0:T-1}(z), \mathbf{v}_{0:T}, \bar{\mathbf{A}}_{0:T-1}(z) \sim q(\cdot|\theta')$.
  4. Sample $\mathbf{N}, \mathbf{k}, \mathbf{x}_{0:T}, \mathbf{A}_{0:T-1}(x), \mathbf{u}_{0:T}^{-\mathbf{1}}, \bar{\mathbf{A}}_{0:T-1}^{-\mathbf{1}}(x) \sim \bar{q}(\cdot|\theta, u_{0:T}^{(1)})$.
  5. With probability
  $$\min\left\{1, \frac{f(\theta')g(\theta|\theta')\hat{Z}_T(\theta', z)}{f(\theta)g(\theta'|\theta)\hat{Z}_T(\theta, x)}\right\},$$
  set $(\theta_t, u_{0:T,t}^{(1)}) = (\theta', v_{0:T}^{(1)})$.
  Otherwise, set $(\theta_t, u_{0:T,t}^{(1)}) = (\theta, u_{0:T}^{(1)})$.

Algorithm 5.10 Adaptive Particle Gibbs

- At iteration t:
  1. Set $u_{0:T}^{(1)} = u_{0:T,t-1}^{(1)}$.
  2. Sample $\theta_t \sim \pi(\cdot|x_{0:T}^*)$.
  3. Sample $\mathbf{N}, \mathbf{k}, \mathbf{x}_{0:T}, \mathbf{A}_{0:T-1}(x), \mathbf{u}_{0:T}^{-\mathbf{1}}, \bar{\mathbf{A}}_{0:T-1}^{-\mathbf{1}}(x) \sim \bar{q}(\cdot|\theta_t, u_{0:T}^{(1)})$.
  4. Sample $\mathbf{l} = (C_{T-1}^{l_T}, l_T)$ with probability $\frac{w_T^{(l_T)}}{\sum_{j=1, j\neq k_T}^{N_T} w_T^{(j)}}$
  5. With probability
  $$\min\left\{1, \rho_2(\mathbf{l})\frac{\sum_{j=1, j\neq k_T}^{N_T} w_T^{(j)}}{\sum_{j=1, j\neq l_T}^{N_T} w_T^{(j)}}\right\},$$
  set $u_{0:T,t}^{(1)} = (x_{0:T-1}^{C_{T-1}^{l_T}}, x_T^{(l_T)})$.
  Otherwise, set $u_{0:T,t}^{(1)} = u_{0:T}^{(1)}$.

number of samples required to meet the threshold at a later time. The second is that it is difficult to ensure that for large T, the correction factor ρ_2 does not tend to 0.

To address the second question, one can use stopping criteria and conditional distributions for the active particles to ensure that $\rho_2 = 1$, as described in Section 4.5.2.2. With particle methods, this entails not allowing the selection of particle N_t in the categorical sampling step at time $t+1$, in addition to the restrictions on the stopping rule. An adaptive SMC algorithm where $\rho_2 = 1$ always is given in Algorithm 5.11. The notation β_j^* is used to remind the reader that the choice of stopping probabilities must satisfy additional requirements for $\hat{Z}_T$ to be an unbiased estimate of Z_T.

5.3 Other Methods

It seems that any reversible MCMC kernel can be viewed using this framework, and it is particularly useful for interpreting kernels for whom detailed balance is often proved by inspection.

5.3.1 Stochastic Potential Switching

The basic stochastic potential switching algorithm of Mak (2005) is the following, where we are interested in sampling from π and we have defined an arbitrary 'switching density' $\tilde{\pi}$. We know only the densities pointwise up to a normalizing constant, so we can use the unnormalized forms γ and $\tilde{\gamma}$. We define some $C > \max_x\{\frac{\tilde{\gamma}(x)}{\gamma(x)}\}$.

We define a switching probability at x by the function $S(x) = \frac{\tilde{\gamma}(x)}{C\gamma(x)}$ and another density proportional to $\bar{\gamma}(x) = (1 - S(x))\gamma(x)$. The algorithm is to at each point x, 'switch' with probability $S(x)$ and if successful, move with a kernel that satisfies detailed balance for $\tilde{\gamma}$ and if unsuccessful, move with a kernel that satisfies detailed balance for $\bar{\gamma}$. This mixture of kernels satisfies detailed balance for $\gamma(x)$.

Algorithm 5.11 Adaptive SMC Algorithm where $\rho_2 = 1$

1. At time $t = 0$.
   - For $j = 1, \ldots$
     - Sample $x_0^{(j)} \sim g_0(\cdot)$.
     - Evaluate the importance weight:
     $$w_0^{(j)} = w_0(x_0^{(j)}) = \frac{\gamma_0(x_0^{(j)})}{g_0(x_0^{(j)})}.$$
     - With probability $\beta_j^*(w_0^{(1:j)})$, set $N_0 = j$ and go to time $t + 1$.
2. For times $t = 1, \ldots, T$.
   - For $j = 1, \ldots$
     - Sample $A_{t-1}^{(j)} \sim \mathcal{M}(\cdot|1, W_{t-1}^{(1:N_{t-1}-1)})$.
     - Sample $x_t^{(j)} \sim g_t(\cdot|x_{0:t-1}^{C_{t-1}^j})$.
     - Evaluate the importance weight:
     $$w_t^{(j)} = w_t(x_{0:t-1}^{C_{t-1}^i}, x_t^{(j)}) = \frac{\gamma_t(x_{0:t-1}^{C_{t-1}^j}, x_t^{(j)})}{\gamma_{t-1}(x_{0:t-1}^{C_{t-1}^j}) g_t(x_t^{(j)}|x_{0:t-1}^{C_{t-1}^j})}.$$
     - With probability $\beta_j^*(w_t^{(1:j)})$, set $N_t = j$ and go to time $t + 1$ if $t < T$ and step 3 if $t = T$.
3. - Sample k_T with probability $\frac{w_T^{(k_T)}}{\sum_{i=1}^{N_T-1} w_T^{(i)}}$, for $k_T \in \{1, \ldots, N_T - 1\}$ and set $k_{0:T-1} = C_{T-1}^{k_T}$.
   - Evaluate
   $$\hat{Z}_T = \prod_{t=0}^{T} \left[\frac{1}{N_t - 1} \sum_{j=1}^{N_t - 1} w_t^{(j)} \right].$$

This scheme arises naturally from writing an extended target density as

$$\tilde{\pi}^2(k, s, x^{(1:2)}) = \frac{1}{2}\pi(x^{(k)})\mathbb{P}[S = s|x^{(k)}]q(x^{(\bar{k})}|s, x^{(k)})$$

where $S \in \{0, 1\}$ is 1 corresponding to a switch and 0 for a failed switch. One Gibbs samples $s, x^{(\bar{k})}|k, x^{(k)}$ and then attempts the swap move $(k, s, x^{(1:2)}) \to (\bar{k}, s, x^{(1:2)})$, which has acceptance probability

$$\min\left\{1, \frac{\pi(x^{(\bar{k})})\mathbb{P}[S = s|x^{(\bar{k})}]q(x^{(k)}|s, x^{(\bar{k})})}{\pi(x^{(k)})\mathbb{P}[S = s|x^{(k)}]q(x^{(\bar{k})}|s, x^{(k)})}\right\}.$$

Note that if $q(x^{(\bar{k})}|1, x^{(k)})$ satisfies detailed balance for $\tilde{\gamma}$ and $q(x^{(\bar{k})}|0, x^{(k)})$ satisfies detailed balance for $\bar{\gamma}$ then this acceptance probability is always 1. The extension in Mak (2005) to the case of multidimensional S and x can be derived in a similar fashion.

5.3.2 Intractable Normalizing Constants

Consider the problem of sampling from a posterior density π defined by

$$\pi(\theta) = p(\theta|y) = \frac{f(y|\theta)/Z(\theta)p(\theta)}{\int_\Theta f(y|\theta)/Z(\theta)p(\theta)d\theta}$$

where $Z(\theta) = \int_\mathcal{X} f(y|\theta)dy$ is the unavailable normalizing constant of the likelihood function f and, crucially, it depends on θ. This normalizing constant could be unavailable because it cannot be computed analytically or because it is too expensive to do so.

In this case, we cannot evaluate π up to a normalizing constant. However, in Møller et al. (2006), an auxiliary variable approach is proposed to overcome this difficulty. This approach can be obtained by defining first an extended target density of the form

$$\tilde{\pi}(\theta, \theta', x, z) = \frac{f(y|\theta)}{Z(\theta)}p(\theta)g(\theta'|\theta)\frac{f(z|\theta')}{Z(\theta')}q(x|\theta, y). \tag{5.7}$$

In this extended target, θ is distributed according to π, θ' is distributed according to the proposal $g(\cdot|\theta)$, z is a sample from $f(\cdot|\theta')/Z(\theta')$ and x is distributed according

to some auxiliary density $q(\cdot|\theta, y)$. The approach of Møller et al. (2006) is to sample $\theta', z \sim \tilde{\pi}(\cdot|\theta, x) = \tilde{\pi}(\cdot|\theta)$ and then attempt the swap move $(\theta, \theta', x, z) \to (\theta', \theta, z, x)$, which has acceptance probability

$$\min\left\{1, \frac{f(y|\theta')p(\theta')g(\theta|\theta')}{f(y|\theta)p(\theta)g(\theta'|\theta)} \times \frac{q(z|\theta', y)}{f(z|\theta')} \times \frac{f(x|\theta)}{q(x|\theta, y)}\right\}. \tag{5.8}$$

Note that in this scheme, the variable x is not sampled in the Gibbs step that occurs before the swap move, and indeed the density q here is not an importance density but rather an auxiliary density that is ideally not too different from the normalized form of f. In fact, since x is at stationarity distributed according to q and $z \sim f(\cdot|\theta)/Z(\theta)$, we have

$$\mathbb{E}\left[\frac{q(z|\theta', y)}{f(z|\theta')} \times \frac{f(x|\theta)}{q(x|\theta, y)}\right] = \frac{Z(\theta)}{Z(\theta')}.$$

Indeed, if one can sample according to q, and so one samples $\theta', z, x \sim \tilde{\pi}(\cdot|\theta)$ before attempting the swap move, the acceptance ratio is again given by (5.8) but in this case, a new value of x is used at every iteration. This can be helpful when q and f have large discrepancies, since when $\frac{f(x|\theta)}{q(x|\theta,y)}$ is close to 0, the previous approach can get stuck.

The MCMC algorithm in Marjoram et al. (2003) for approximate Bayesian computation (ABC) can be seen as a special case of this framework, where one chooses

$$q(x|\theta, y) = \frac{f(x|\theta)\mathbf{1}_{B_\epsilon(x)}(y)}{\int_{\mathcal{X}} f(x|\theta)\mathbf{1}_{B_\epsilon(x)}(y)dx}$$

with the ABC-specific notation defined in 1.1.2.4. This involves an approximation unless $f(y|\theta) = Z_\epsilon^{-1}\int_{\mathcal{X}} f(x|\theta)\mathbf{1}_{B_\epsilon(x)}(y)dx$, which generally holds only when the data is discrete and $\epsilon = 0$. The actual posterior distribution explored by such an algorithm is the one with likelihood defined by $f_\epsilon(y|\theta)$, with more details in Section 4.6.1. This, however, has the obvious advantage of being computationally available when one cannot even evaluate $f(\cdot|\theta)$. Nevertheless, the algorithm has been used to perform approximate Bayesian model selection for Gibbs random fields (Grelaud et al. 2009) in cases where one can evaluate $f(\cdot|\theta)$.

An observation in Murray et al. (2006) is that the introduction of q is not necessary

to define an MCMC kernel that admits π as a marginal. Consider the extended target density

$$\tilde{\pi}(\theta, \theta', z) = \frac{f(y|\theta)}{Z(\theta)} p(\theta) g(\theta'|\theta) \frac{f(z|\theta')}{Z(\theta')}. \tag{5.9}$$

The acceptance probability for the swap move $(\theta, \theta', z) \to (\theta', \theta, z)$ is

$$\min\left\{1, \frac{f(y|\theta')p(\theta')g(\theta|\theta')}{f(y|\theta)p(\theta)g(\theta'|\theta)} \times \frac{f(z|\theta)}{f(z|\theta')}\right\}$$

and the resulting algorithm is termed the exchange method.

Murray et al. (2006) also introduces two bridge sampling (Jarzynski 1997; Neal 2001) modifications, whose algorithms can be derived upon expression of their corresponding extended target densities. The first is the multiple auxiliary variable method, for which we can write

$$\begin{aligned}\tilde{\pi}(\theta, \theta', x_{0:T}, z_{0:T}) = &\frac{f(y|\theta)}{Z(\theta)} p(\theta) g(\theta'|\theta) \frac{f(z_0|\theta')}{Z(\theta')} \\ &\times \prod_{t=1}^{T} K_t(z_{t-1}, z_t) q(x_T|\theta, y) \prod_{t=0}^{T-1} L_t(x_{t+1}, x_t)\end{aligned}$$

and note that their algorithm does not sample $x_{0:T}$, whose value is only changed during a swap move. Each kernel K_t is an MCMC kernel with invariant distribution q_t, and the $q_1, q_2, \ldots, q_T$ are typically chosen such that q_1 is close to $f(\cdot|\theta')/Z(\theta')$, $q_T = q$, and q_j is close to q_{j+1} for all $j \in \{1, T-1\}$. The backwards kernels satisfy

$$L_{t-1}(x_t, x_{t-1}) = \frac{q_t(x_{t-1}) K_t(x_{t-1}, x_t)}{q_t(x_t)}$$

where one defines $q_0(\cdot) = f(\cdot|\theta')/Z(\theta')$.

Their second bridging method, the exchange algorithm with bridging, can be derived from the extended target density

$$\tilde{\pi}(\theta, \theta', x_{0:T}, z_{0:T}) = \frac{f(y|\theta)}{Z(\theta)} p(\theta) g(\theta'|\theta) \frac{f(z_0|\theta')}{Z(\theta')} \prod_{t=1}^{T} K_t(z_{t-1}, z_t) \delta_{z_{T:0}}(x_{0:T})$$

where in this case it is essential that K_t be a reversible MCMC kernel and we define $z_{T:0} = (z_T, \ldots, z_0)$. In this case, the bridging occurs between $f(\cdot|\theta')/Z(\theta')$ and $f(\cdot|\theta)/Z(\theta)$.

5.3.2.1 Flexible Auxiliary Distributions

The definition of (5.7) can be extended, by noting that x can be defined conditionally upon both θ and θ'.

$$\tilde{\pi}(\theta, \theta', x, z) = \frac{f(y|\theta)}{Z(\theta)} p(\theta) g(\theta'|\theta) \frac{f(z|\theta')}{Z(\theta')} q(x|\theta, \theta', y). \tag{5.10}$$

The swap move $(\theta, \theta', x, z) \to (\theta', \theta, z, x)$ in this case has acceptance probability, if one samples $x \sim q(\cdot|\theta, \theta', y)$ at each iteration, given by

$$\min \left\{ 1, \frac{f(y|\theta')p(\theta')g(\theta|\theta')}{f(y|\theta)p(\theta)g(\theta'|\theta)} \times \frac{q(z|\theta', \theta, y)}{f(z|\theta')} \times \frac{f(x|\theta)}{q(x|\theta, \theta', y)} \right\}. \tag{5.11}$$

A number of new potential choices for q are possible in this case. Interesting possibilities include the specification

$$q(x|\theta, \theta', y) = \frac{f(x|h(\theta, \theta'))}{Z(h(\theta, \theta'))}$$

where $h : \Theta \times \Theta \to \Theta$ satisfies $h(\theta, \theta') = h(\theta', \theta)$. In this case, the unknown normalizing constant $Z(h(\theta, \theta'))$ will cancel in the acceptance probability. Possible choices of h include $h(\theta, \theta') = \min\{\theta, \theta'\}$, $h(\theta, \theta') = \max\{\theta, \theta'\}$, $h(\theta, \theta') = \frac{\theta+\theta'}{2}$ or $h(\theta, \theta') = \sqrt{\theta\theta'}$. These last two specifications have the best performance in the toy example described in Section 5.4.3. Another possible choice, although it exhibited poor performance in the toy example is $q(x|\theta, \theta', y) \propto f(x|\theta)f(x|\theta')$, where the cancellation of the normalizing constant in the acceptance probability also occurs. These choices can be better than choosing $q(x|\theta, y) \propto f(x|\hat{\theta})$ for a single value $\hat{\theta}$, one of the suggestions in Møller et al. (2006).

The algorithm with the additional sampling of x is described in Algorithm 5.12, where it should be clear that the collapsed state space of the Markov chain is Θ.

Algorithm 5.12 MCMC Algorithm for Intractable Normalizing Constants

Given (θ_0), at times $t = 0, 1, \ldots$

1. Set $\theta = \theta_t$.
2. Sample $\theta' \sim g(\cdot|\theta)$.
3. Sample $z \sim f(\cdot|\theta')/Z(\theta')$.
4. Sample $x \sim q(\cdot|\theta, \theta', y)$.
5. With probability

$$\min\left\{1, \frac{f(y|\theta')p(\theta')g(\theta|\theta')}{f(y|\theta)p(\theta)g(\theta'|\theta)} \times \frac{q(z|\theta', \theta, y)}{f(z|\theta')} \times \frac{f(x|\theta)}{q(x|\theta, \theta', y)}\right\},$$

set $\theta_{t+1} = \theta'$.
Otherwise, set $\theta_{t+1} = \theta$.

5.3.2.2 More Auxiliary Variables

Returning to the general approach of Møller et al. (2006), when sampling according to $f(\cdot|\theta)/Z(\theta)$ is inexpensive, one can define a further extended target density

$$\tilde{\pi}^N(k, l, \theta, \theta', x^{(1:N)}, z^{(1:N)}) = \frac{1}{N}\frac{f(y|\theta)}{Z(\theta)}p(\theta)g(\theta'|\theta)\left[\prod_{i=1}^{N}\frac{f(z^{(i)}|\theta')}{Z(\theta')}\right]\frac{w_L^{(l)}}{\sum_{j=1}^{N} w_L^{(j)}}q(x^{(k)}|\theta, \theta', y)\prod_{i=1, i\neq k}^{N}\frac{f(x^{(i)}|\theta)}{Z(\theta)}$$

where in addition we have introduced the fact that q can condition also on θ' if we can sample according to q. If we let $w_L^{(i)} = q(z^{(i)}|\theta', \theta, y)/f(z^{(i)}|\theta')$, the acceptance probability of the swap $(k, l, \theta, \theta', x^{(1:N)}, z^{(1:N)}) \to (l, k, \theta', \theta, z^{(1:N)}, x^{(1:N)})$ is

$$\min\left\{1, \frac{f(y|\theta')p(\theta')g(\theta'|\theta)}{f(y|\theta)p(\theta)g(\theta|\theta')} \times \frac{\frac{1}{N}\sum_{j=1}^{N} q(z^{(j)}|\theta', \theta, y)/f(z^{(j)}|\theta')}{\frac{1}{N}\sum_{j=1}^{N} q(x^{(j)}|\theta, \theta', y)/f(x^{(j)}|\theta)}\right\}.$$

Note that one need not sample according to q if it is independent of θ', but it may be beneficial to do so anyway.

We have

$$\mathbb{E}\left[\frac{1}{N}\sum_{j=1}^{N}\frac{q(z^{(j)}|\theta', \theta, y)}{f(z^{(j)}|\theta')}\right] = \frac{1}{Z(\theta')}$$

and that $N^{-1}\sum_{j=1}^{N} q(x^{(j)}|\theta,\theta',y)/f(x^{(j)}|\theta)$ is a consistent estimate of $1/Z(\theta)$ as $N \to \infty$ as long as $q(x^{(j)}|\theta,\theta',y)/f(x^{(j)}|\theta)$ is well-behaved. This may be easy to guarantee for some of the choices in Section 5.3.2.1.

It is also possible to design algorithms using SMC samplers to obtain lower variance estimates of the quantities that appear in the acceptance ratio, and the extra expense of this type of approach may be justified when the cost of perfect samples from $f(\cdot|\theta)/Z(\theta)$ is very high. Note that an option in this case is using only one particle for the first time step, so that each SMC sampler run needs exactly one perfect sample according to f.

5.4 Applications

As in Chapter 4, the applications here are intended to highlight features of the methodology. An application to parameter estimation in state-space models is followed by a brief comment on Bayesian variable selection in high dimensions and some simulations showing the effectiveness of various choices of auxiliary densities when dealing with a toy doubly intractable inference problem.

5.4.1 State-Space Models

With independent proposals in an MCMC scheme, empirical performance suggests that $N > 1$ does not lead to superior performance of the kernel when run on a machine with no parallel capabilities. In contrast, for the dependent proposals produced using SMC in a state-space model this does not appear to be the case for some intermediate fixed values of N. This suggests further that the adaptation of $\mathbf{N}$ could lead to superior performance both in a pseudo-marginal setting and in settings with a fixed value θ.

We consider again the toy non-linear state-space model described in Section 2.6.1.5:

$$X_t = \frac{X_{t-1}}{2} + 25\frac{X_{t-1}}{1+X_{t-1}^2} + 8\cos(1.2t) + V_t$$
$$Y_t = \frac{X_t^2}{20} + W_t$$

where $V_t \sim N(0, \sigma_V^2)$, $W_t \sim N(0, \sigma_W^2)$ and $X_0 \sim N(0, \sigma_V^2)$ and we adopt a prior on $\theta = (\sigma_V^2, \sigma_W^2)$ where $\sigma_V^2 \sim IG(1,1)$ and $\sigma_W^2 \sim IG(1,1)$, where IG denotes the inverse-gamma density. We let $T = 200$ and simulated $y_{1:T}$ according to this model, using $\theta = (10, 0.1)$, where the low value of σ_W^2 can make the model more difficult to sample from.

5.4.1.1 Adaptive vs. Standard SMC

We first try to characterize the effect of adaptation of $\mathbf{N}$ on the quality of samples from $\pi(x_{0:T}|\theta)$ with θ fixed. Since the expectation of $\hat{Z} = \hat{Z}_T$ is equal to $p(y_{1:T}|\theta)$ regardless of the presence and type of adaptation, one possible method is to compute the effective sample size of $\{\hat{Z}_i\}_{i=1}^M$ for $M = 100000$ running an adaptive and non-adaptive SMC algorithm. This corresponds to, in some sense, the effective number of filters run. For simplicity, we consider the use of the bootstrap filter that proposes particles using the transition dynamics and weights each particle using the observation density. The adaptive SMC algorithm defines $\bar{\beta}_j(w^{(1:j)}) = \beta_j(w^{(1:j)}) = \mathbf{1}_{[1000,\infty)}(j)\mathbf{1}_{[c,\infty)}(\sum_{i=1}^j w^{(i)})$, where $c = 25N(0;0,\sigma_W^2)$, i.e. 25 times the maximum possible weight of a particle. The first term indicates that the minimum value of N_t is 1000. This adaptive filter uses Algorithm 5.11 to ensure the correction factor is always 1. After running the adaptive filter M times, a standard bootstrap filter is run with fixed N given by the average value of N_t over all times and filters. This value was 2042. The effective sample size using the adaptive filter was 3969 while using the standard filter it was 2561. Figure 5.1 displays the sorted values of $\log \hat{Z}$ from both filters, where it is clear that the estimate from the adaptive filter has less empirical variance, and Figure 5.2 shows histograms and density estimates of $\log \hat{Z}$ for the two filters.

These results suggest two important points. The relatively large variation in $\log \hat{Z}$

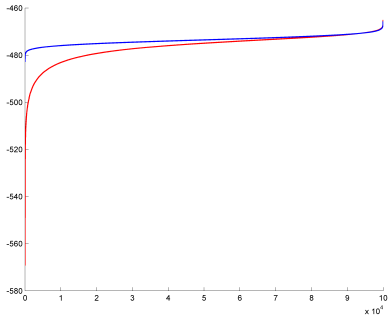
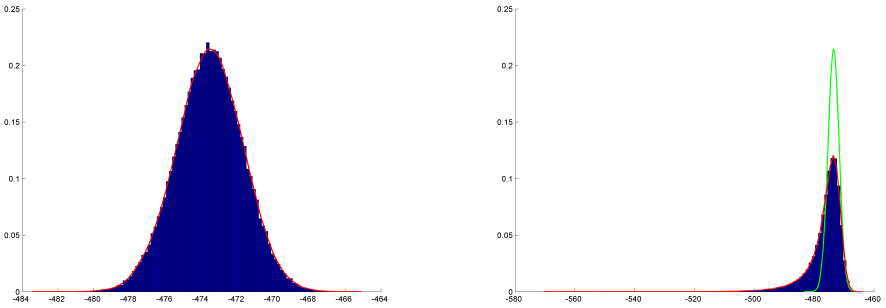


Figure 5.1: Plot of sorted $\log \hat{Z}$ values for the standard (red) and adaptive (blue) filters.



(a) Adaptive Particle Filter (b) Standard Particle Filter

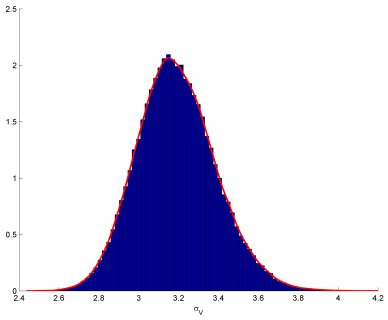
Figure 5.2: Histograms and density estimates of $\log \hat{Z}$ for fixed θ. The green density estimate in (b) is that of the adaptive filter, for comparison.

suggests that in a pseudo-marginal setting where one has proposed a move from θ to θ', the rejuvenation of the non-active particles associated with θ can have a significant effect on the acceptance probability of a move. In addition, the distribution of $\log \hat{Z}$ for the adaptive filter is close to that of a normal distribution, despite the fact that the central limit theorems in the literature do not apply to the adaptive algorithm. This is especially interesting since the distribution of $\log \hat{Z}$ for the standard filter is not approximately normal for the given fixed value of N. It is also important to note that while the average value of N is 2042, the adaptive filter uses at most times t, $N_t = 1000$ but that at a few times the value of N_t is much larger, with values typically between 10000 and 70000.

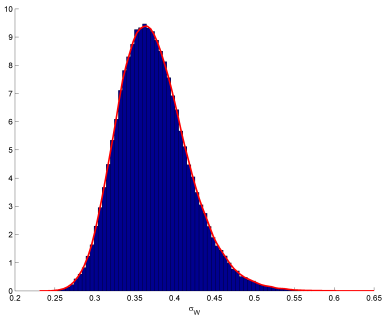
5.4.1.2 Adaptive Particle Markov Chain Monte Carlo

We next consider inference on θ using the same data set, and run an adaptive particle Gibbs sampler, a particle Gibbs sampler, an adaptive PMMH sampler and a standard PMMH sampler. All samplers sample from $\bar{q}$ in addition to q at each iteration. The filters are each run for 150000 iterations and the number of particles used for the particle Gibbs and PMMH filters are equal to the average number of particles used for the adaptive particle Gibbs and adaptive PMMH filters respectively. Note that in a real application, one would not be able to use such a value without running an adaptive algorithm and the value of N would typically be chosen by some other means. Figures 5.3-5.7 show the resulting histograms and autocorrelation plots. The particle Gibbs algorithms converge more quickly than the PMMH algorithms and the adaptive algorithms seem to converge more quickly than their non-adaptive counterparts, with the difference more noticeable for the PMMH algorithms. Indeed, the adaptive PMMH algorithm accepted 28805 moves in θ whilst the PMMH algorithm accepted only 17006. The average value of N used by the adaptive particle Gibbs sampler was 1911 and the average value of N used by the adaptive PMMH sampler was 1992.

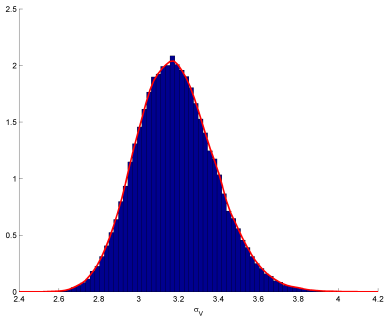
An important observation is that the separate adaptation of N and N' can be wasteful of computational resources. This is due to the fact that for values of θ that have low density under π, the value $\hat{Z}(\theta)$ is small because the values of N_t are high



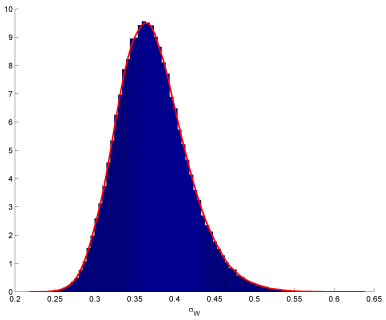
(a) Adaptive Particle Gibbs



(b) Adaptive Particle Gibbs

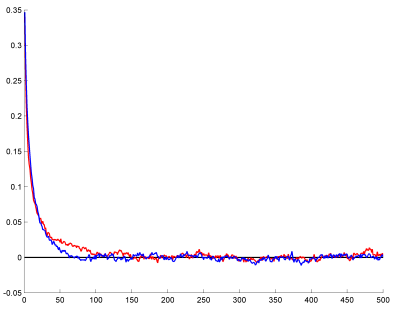


(c) Particle Gibbs

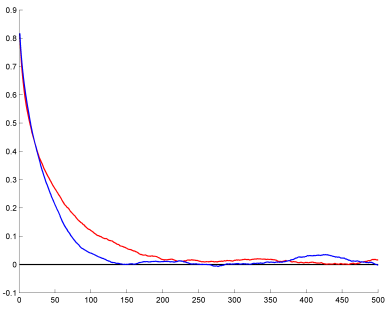


(d) Particle Gibbs

Figure 5.3: Adaptive Particle Gibbs and Particle Gibbs Histograms.

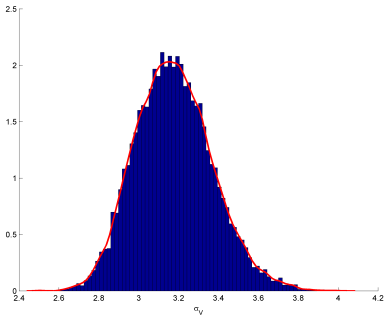


(a) σ_V

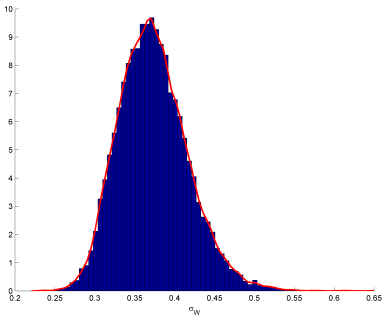


(b) σ_W

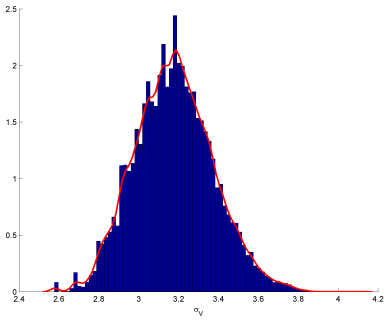
Figure 5.4: Autocorrelations for the Adaptive Particle Gibbs (blue) and Particle Gibbs (red) algorithms.



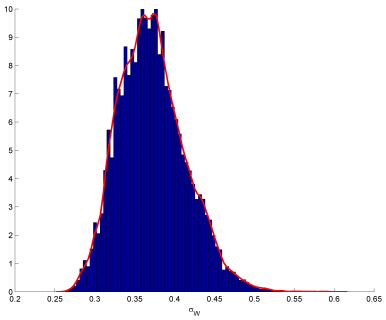
(a) Adaptive PMMH



(b) Adaptive PMMH

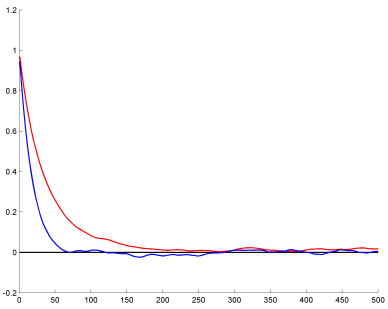


(c) PMMH

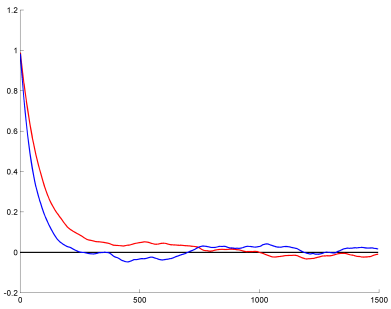


(d) PMMH

Figure 5.5: Adaptive PMMH and PMMH Histograms

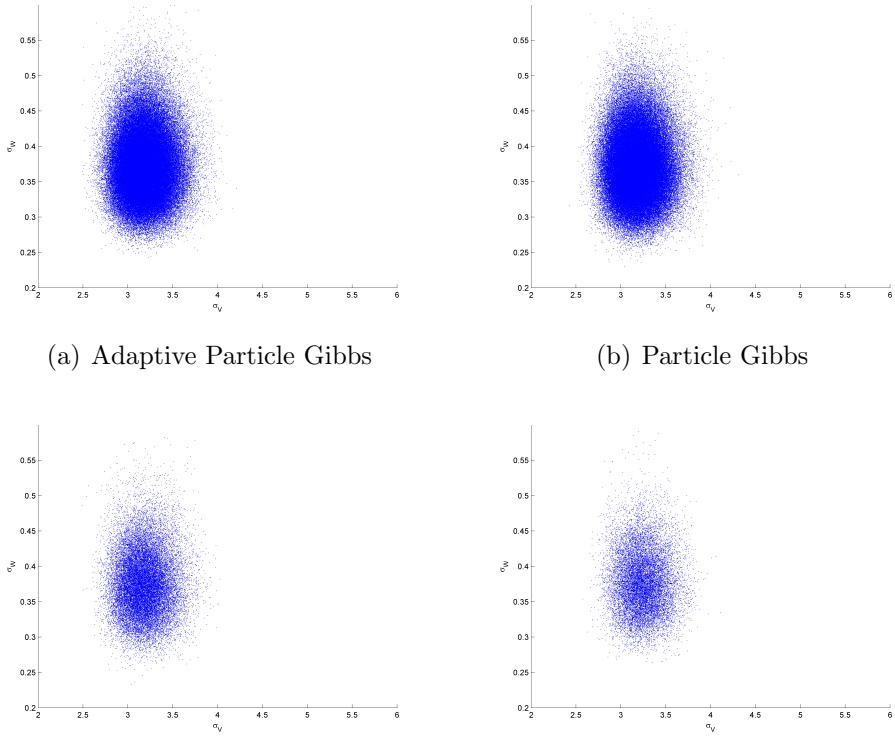


(a) σ_V



(b) σ_W

Figure 5.6: Autocorrelations for the Adaptive PMMH (blue) and PMMH (red) algorithms. Note that the x-axis for σ_W has been lengthened to 1500.



(a) Adaptive Particle Gibbs

(b) Particle Gibbs

(c) Adaptive PMMH

(d) PMMH

Figure 5.7: Scatter Plots.

for some $t \in \{0, \ldots, T\}$. This has the unfortunate consequence that large amounts of time are spent running filters for poor values of θ. A possible remedy would be to have $N = N'$ and to define the distribution of N jointly with the random variables associated with both filters.

In addition, many modifications could be made to the adaptive scheme to improve performance. For example, the use of purely categorical sampling regardless of the value of N_{t-1} negatively impacts the performance of the algorithm. Other adjustments could also be made to improve the filter.

5.4.2 Variable Selection in High Dimensions

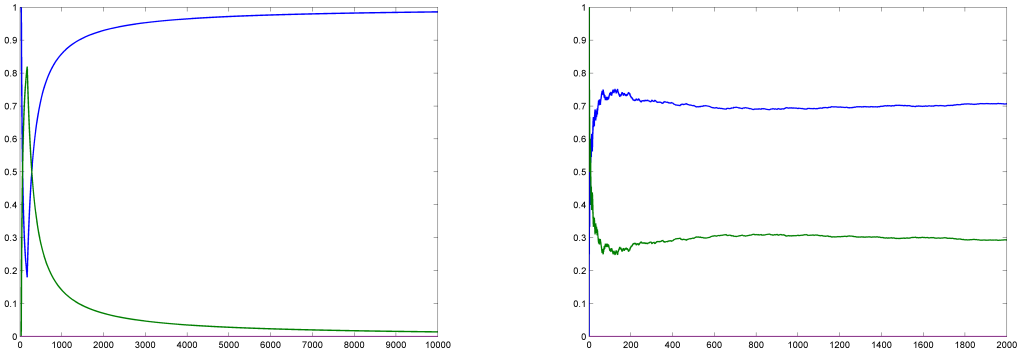
We return to the variable selection example described in Section 4.6.2. When p is large, the use of simple importance sampling weights in a pseudo-marginal algorithm is subject to the curse of dimensionality, in that the variance of the weights

will typically increase exponentially in the dimension of the model and can lead to poor performance. SMC methodology can be employed instead to produce samples, motivated by the fact that the variance of the normalizing constant estimate, which is the importance weight associated with an pseudo-marginal SMC algorithm, can have a variance that does not increase exponentially in the dimension of its target (Del Moral 2004, Chapter 9). Whether or not this holds depends on the choice of auxiliary target densities, the kernels used to move the particles, and the backwards kernels used to define the importance weights associated with those moves.

To highlight this issue, we consider model selection between just two models, one with 50 variables and another with 51 variables. The values of β_j used to simulate n data points in this case are each drawn independently from a standard normal distribution. The multiple-try pseudo-marginal method uses the same proposal density as before and an adaptive SMC pseudo-marginal algorithm is compared with it that uses $T = 100$, a quadratic tempering schedule from $g(\cdot|\mathcal{M}_\gamma)$ to $p(\boldsymbol{\beta}_\gamma|\mathbf{y}, \mathbf{X}_\gamma, \mathcal{M}_\gamma)$, a random-walk MCMC kernel and the generic backwards kernel described in Section 1.2.4.2. The adaptive stopping rule is 0-1 and uses a minimum value of $N = 100$ and stops only when the sum of the importance weights for the next time is greater than or equal to 10. In the event that $N_t > 100$, the N_tth particle cannot be resampled at time $t + 1$, in order to ensure that $\rho_2 = 1$.

Figure 5.8 shows the resulting plots of the Monte Carlo estimates of the model probabilities at each iteration, for 50000 iterations. It is clear that the standard multiple-try pseudo-marginal method is not mixing well whilst the SMC variant's estimates are converging to their true values quickly. In this case, the adaptation of $\mathbf{N}$ has little effect as most of the values of N_t are 100 and in all simulations are less than 120. While clearly somewhat expensive in requiring 10000 samples and MCMC moves, this number is dwarfed by the number of samples one would need using independent proposals, as can be seen in Section 4.6.2.3 where in 9 dimensions one needs roughly 25 million samples to obtain an ESS of 10.

This type of method does not solve the twin problem in high-dimensional variable selection: that it is difficult to construct a good proposal for moves in the model space. A natural way to deal with both issues is to employ a suitable tempering



(a) MTM (b) PM-SMC

Figure 5.8: Monte Carlo estimate of the model probabilities at each iteration. The relative true model probabilities are 0.72 (blue) and 0.28 (green).

scheme in a population-based MCMC approach with SMC proposals for the coefficients. At high temperatures, less effort can be required to obtain samples whilst at low temperatures more effort is required but the proposed models are more likely to be good, justifying the expense of obtaining accurate estimates of the marginal likelihood for these models. Related ideas for tempering as a method to generate good proposals for the models have been suggested in Bottolo and Richardson (2010).

5.4.3 Intractable Normalizing Constants

We consider here the toy example in Murray et al. (2006), in which we are interested in sampling from the posterior distribution of the precision parameter θ of a zero-mean normal distribution when we have n i.i.d. observations $y_{1:n}$ and a conjugate prior on θ. In particular, we have $p(y_{1:n}|\theta) = \prod_{i=1}^{n} p(y_i|\theta)$, $p(y_i|\theta) = N(y_i; 0, \theta^{-1})$ for $i \in \{1, \ldots, n\}$ and $p(\theta|\alpha, \beta) = \text{Gamma}(\theta; \alpha, \beta)$. We decompose $p(y_i|\theta) = f(y_i|\theta)/Z(\theta)$ where $f(x|\theta) = \exp(-\theta x^2/2)$ and $Z(\theta) = (2\pi/\theta)^{-1/2}$. While $Z(\theta)$ is easily computable, we will treat it as unavailable for the sake of experiment. Note that in this case, we can also write the posterior density analytically since $p(\theta|y_{1:n}) = \text{Gamma}(\theta; \alpha + n/2, \beta + 1/2 \sum_{i=1}^{n} y_i^2)$.

We generated data with $n = 10$, $y_i \overset{\text{iid}}{\sim} N(0, 1.5^2)$ for $i \in \{1, \ldots, n\}$ and proceeded to implement MCMC algorithms using Møller's method with $q(x|\theta, y) \propto f(x|\hat{\theta})$,

the exchange algorithm, Møller's method as above where we additionally sample $x \sim q(\cdot|\theta, y)$ at each iteration of the algorithm, the averaged method where $q(x|\theta, \theta', y) \propto f(x|\frac{\theta+\theta'}{2})$ and the standard MCMC algorithm that computes $p(y_{1:n}|\theta)$ exactly. Møller's method requires the specification of $\hat{\theta}$ for which we first try $\hat{\theta} = 1$. After 10^6 iterations, all of the samplers provide excellent density estimates of the posterior and have converged. Figure 5.9 plots the autocorrelation functions of the methods, where it is clear that of the auxiliary variable methods, the averaged method performs best, followed closely by the exchange method and Møller's method with additional sampling from q. Møller's method without additional sampling has a slowly decaying autocorrelation function in comparison.

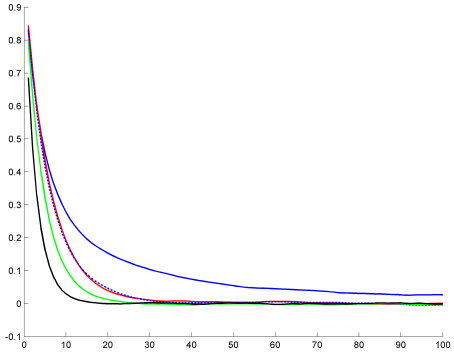
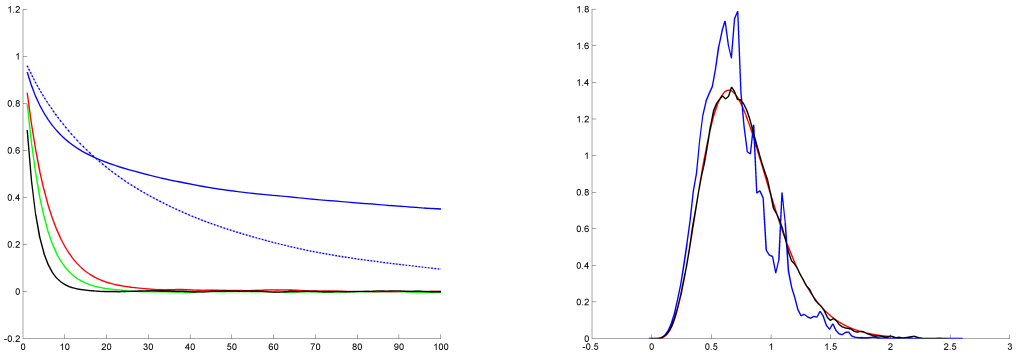


Figure 5.9: Autocorrelations for the first example. Overlaid are the plots for Møller's method (blue), the exchange method (red), Møller's method with sampling from q (blue dots), the averaged method (green) and the non-auxiliary variable method (black).

The performance of Møller's method is particularly sensitive to the choice of q, which is in this case given by $\hat{\theta}$. Using $\hat{\theta} = 0.25$, both Møller's method and its variant that additionally samples from q perform much worse than the exchange algorithm, as shown in Figure 5.10. It is clear that both variant's have considerably poorer performance than the methods that do not depend on a chosen parameter $\hat{\theta}$, and that the method that additionally samples x according to q is superior. In this latter case, it is interesting to note that the acceptance rate is roughly equal for both methods, and so this is not a good measure of the effectiveness of the sampler.

Other choices of q as $q(x|\theta, \theta', y) \propto f(x|\min\{\theta, \theta'\})$, $q(x|\theta, \theta', y) \propto f(x|\max\{\theta, \theta'\})$, $q(x|\theta, \theta', y) \propto f(x|\sqrt{\theta\theta'})$ and $q(x|\theta, \theta', y) \propto f(x|\theta)f(x|\theta')$ were also tried. For the first two, the resulting sampler was similar in effectiveness to the exchange algorithm. The third had the same performance as the averaged method. The last option



(a) Autocorrelations

(b) Density estimates

Figure 5.10: Autocorrelations and density estimates with poor choice of $\hat{\theta}$. The autocorrelation plots have the same colour scheme as Figure 5.9. In (b) we have the true density (red), the density estimate for Møller's method (blue) and the density estimate for Møller's method with sampling from q (black).

performed poorly, only providing a significant improvement over fixed $\hat{\theta}$.

The results on this toy problem will not necessarily be replicated in more sophisticated applications. However, it seems sensible to define q in such a way that its distribution can be sampled from, and to sample x from it at each iteration of an MCMC scheme. This avoids to a certain extent the artificially slow mixing of the chain when samples from f are particularly bad at approximating q. The ability to define q conditionally upon both θ and θ', in which case sampling x according to q is mandatory for the acceptance probability to be of the same form, additionally allows q to be adapted to the current state of the chain, and in particular to be proportional to $f(x|h(\theta,\theta'))$, where $h : \Theta \times \Theta \to \Theta$ satisfies $h(\theta,\theta') = h(\theta',\theta)$.

5.5 Discussion

The interaction of the proposed samples in particle methods, or SMC, lead to a situation in which the rate of convergence is empirically much improved for intermediate values of N, even when run on a serial machine. This suggests their use more widely when considering obtaining benefit from the use of multiple proposals.

The adaptation of $\mathbf{N}$ in particle methods occurs at each step of the SMC algorithm, providing an opportunity to automatically tune the number of particles sampled in an attempt to simultaneously approximate an auxiliary distribution more accurately and avoid future particle degeneracy. In this case, the ability to ensure that the correction factor associated with the adaptation of N is always equal to 1 provides some assurance that the method is not limited to cases where the number of auxiliary distributions in the SMC sampler is suitably small.

The material presented in this chapter builds upon the work in Chapter 4 and provides a number of methodological opportunities in the further development of robust particle methods. A number of minor results that can be derived have been omitted for space, such as the ability to perform alternative resampling schemes satisfying the 'unbiasedness' property in Andrieu et al. (2010) or the derivation of the extended target density associated with the marginal particle filter of Klaas et al. (2005).

In some cases, writing the joint density of all random variables involved in a non-trivial reversible MCMC kernel with auxiliary variables can suggest modifications that lead to a new kernel with better performance. In some cases, one can collapse the state of the resulting Markov chain to a lower dimensional space, which may provide computational benefits.

The adaptation of N, which can also be ancillary to the adaptation of some random variable used in the proposal, also provides many further opportunities for investigation. In the applications of both this and the preceding chapter, we have focused on only a subset of the possibilities that the approach allows, and a considerable amount of future work is anticipated in this area.

Chapter 6

Conclusions

This thesis has been motivated by recent trends in computer architecture from single-core to multi-core and further to many-core or massively multi-core processors. Such architectures present opportunities in computational statistics for the proliferation and improvement of methodology that is amenable to parallel computation in specific ways.

In Chapter 2, we have seen that population-based methods are a particular branch of Monte Carlo methodology that can benefit immensely from implementation on emerging many-core devices. In addition, the fact that random variables in these methods interact in specific ways allows the resulting Monte Carlo estimates to have lower variance than that of classical Monte Carlo methods run in parallel. The chapter ended with the derivation of a population-based MCMC algorithm for static parameter inference in state-space models that exhibited reasonable convergence rates on the non-trivial state-space model considered, and highlights the need for methodological attention to interaction kernels used in population-based methods in general.

In Chapter 3, a hierarchical framework for constructing sparsity-inducing priors was introduced, providing a Bayesian interpretation and generalization of many popular penalized optimization approaches in the literature. The MAP estimates that result from priors in this framework can be calculated quickly and are sparse under

some conditions, but their use from a Bayesian perspective is not particularly justified. The chapter concludes with a description of sparsity path analysis, a type of exploratory analysis characterized by Monte Carlo sampling from a sequence of posteriors defined by a sequence of priors with increasing sparsity. When implemented using SMC on a many-core architecture, this computationally expensive method has practical running time for medium-sized data and provides data owners with a variety of statistics that can be used to inform future experiments.

Chapter 4 describes a reinterpretation of Metropolis-Hastings kernels defined on an extended space, which provides a fertile perspective on reversible MCMC kernels and auxiliary variables. In particular, this view of MCMC allows for the formulaic addition of additional auxiliary variables whose purpose is ostensibly to improve the rate of convergence of an MCMC kernel. Importantly, despite the addition of auxiliary variables, the state space of the Markov chain can often be collapsed considerably. While many potential improvements for a variety of reversible kernels are suggested in this chapter, the most notable is perhaps the ability to define a joint distribution for the number of auxiliary variables sampled along with those same auxiliary variables in what could be called locally adaptive MCMC. This occurs in such a way that the equilibrium distribution of the kernel is unchanged by the adaptation, which is in contrast to classical adaptive MCMC methods.

Chapter 5 presents the application of the perspective developed in Chapter 4 to proposals generated using SMC, or particle methods, as well as to auxiliary variable MCMC schemes involving switching densities and doubly intractable distributions. The adaptation of the number of particles in SMC occurs during the approximation of each intermediate density and suggests that the resulting algorithm can spend extra computational resources at stages where they will be most useful.

The work presented here, particularly in Chapters 4 and 5, suggest a wealth of possible research questions in the near term. These include questions on how to specify the stopping rules associated with locally adaptive algorithms, how to specify general dependent proposals outside of an SMC framework, and what strategies for adaptation will be effective outside of a pseudo-marginal framework. With respect to locally adaptive particle filters, understanding the effects of adaptation can be

complicated by the interacting nature of the proposals despite the fact that adaptation occurs only within each step of the algorithm. Questions also remain on the characterization of pseudo-marginal approaches as a function of a fixed number of auxiliary variables, and in what circumstances lazy Gibbs sampling could be preferable due to its cheaper computational cost. The ability to use a joint, and potentially adaptive, proposal for both the forward and backward auxiliary variables also provides opportunities for developing a general framework specifying how this can be done in order to enhance algorithms presented in both chapters. In addition, generalizations of the methodology to incorporate multiple active particles, as well as determining when this is beneficial are also interesting directions for research.

More general questions remain on how computational statistics, and particularly Monte Carlo methodology, will adapt to emerging many-core architectures. It is clear that some degree of parallelism will be desired in state of the art algorithms. The shared memory component of new hardware allows for significant interaction between random variables generated in parallel within complex methods, which allows for considerably more flexibility than previously available parallel architectures. Future work more generally could involve looking for new types of particle interactions or general purpose non-population-based auxiliary variables that could be used to exploit parallelism.

Bibliography

Andrieu, C., Doucet, A., and Holenstein, R. "Particle Markov chain Monte Carlo methods." *Journal of the Royal Statistical Society B*, 72(3):269–342 (2010).

Andrieu, C. and Roberts, G. "The pseudo-marginal approach for efficient Monte Carlo computations." *Annals of Statistics*, 37(2):697–725 (2009).

Andrieu, C. and Thoms, J. "A tutorial on adaptive MCMC." *Statistics and Computing*, 18(4):343–373 (2008).

Armagan, A., Dunson, D., and Lee, J. "Generalized double Pareto shrinkage." (2011). ArXiv:1104.0861.

Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W., and Yelick, K. A. "The Landscape of Parallel Computing Research: A View from Berkeley." Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley (2006).

Athreya, K., Doss, H., and Sethuraman, J. "On the convergence of the Markov chain simulation method." *Annals of Statistics*, 24(1):69–100 (1996).

Ayers, K. L. and Cordell, H. J. "SNP Selection in genome-wide and candidate gene studies via penalized logistic regression." *Genetic Epidemiology*, 34:879–891 (2010).

Balding, D. J. "A tutorial on statistical methods for population association studies." *Nature Reviews Genetics*, 7:81–791 (2006).

Barker, A. "Monte Carlo Calculations of the Radial Distribution Functions for a Proton-Electron Plasma." *Australian Journal of Physics*, 18(2):119–134 (1965).

Barnard, G. A. "Discussion of paper by M. S. Bartlett." *Journal of the Royal Statistical Society B*, 25 (1963).

Beaumont, M. "Estimation of population growth or decline in genetically monitored populations." *Genetics*, 164(3):1139 (2003).

Beaumont, M., Cornuet, J., Marin, J., and Robert, C. "Adaptive approximate Bayesian computation." *Biometrika*, 96(4):983 (2009).

Besag, J. and Green, P. "Spatial statistics and Bayesian computation." *Journal of the Royal Statistical Society B*, 55(1):25–37 (1993).

Bornn, L., Doucet, A., and Gottardo, R. "An efficient computational approach for prior sensitivity analysis and cross-validation." *Canadian Journal of Statistics*, 38:47–64 (2010).

Bottolo, L. and Richardson, S. "Evolutionary Stochastic Search for Bayesian Model Exploration." *Bayesian Analysis*, 5(3):583–618 (2010).

Brockwell, A. E. "Parallel Processing in Markov chain Monte Carlo Simulation by Pre-Fetching." *Journal of Computational and Graphical Statistics*, 15(1):246–261 (2006).

Candès, E. J., Wakin, M. B., and Boyd, S. P. "Enhancing Sparsity by Reweighted ℓ_1 Minimization." *Journal of Fourier Analysis and Applications*, 14:877–905 (2008).

Caron, F. and Doucet, A. "Sparse Bayesian nonparametric regression." In *Proceedings of the 25th international conference on Machine learning*, 88–95. ACM (2008).

Carvalho, C., Polson, N., and Scott, J. "The horseshoe estimator for sparse signals." *Biometrika*, 97(2):465 (2010).

Celeux, G., Hurn, M., and Robert, C. P. "Computational and Inferential Difficulties with Mixture Posterior Distributions." *Journal of the American Statistical Association*, 95:957–970 (2000).

Cevher, V. "Learning with Compressible Priors." In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems 22*, 261–269 (2009).

Chartrand, R. and Yin, W. "Iteratively reweighted algorithms for compressive sensing." In *33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2008).

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

Chopin, N., Jacob, P., and Papaspiliopoulos, O. "SMC^2: A sequential Monte Carlo algorithm with particle Markov chain Monte Carlo updates." (2011). ArXiv:1101.1528.

Crooks, G. E. "Nonequilibrium measurements of free energy differences for microscopically reversible Markovian systems." *Journal of Statistical Physics*, 90:1481–1487 (1998).

Cui, T., Fox, C., Nicholls, G. K., and O'Sullivan, M. J. "Using Parallel MCMC Sampling to Calibrate a Computer Model of a Geothermal Reservoir." Technical Report 686, Univeristy of Auckland, Faculty of Engineering (2011).

Del Moral, P. *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. Springer Verlag (2004).

Del Moral, P., Doucet, A., and Jasra, A. "Sequential Monte Carlo Samplers." *Journal of the Royal Statistical Society B*, 68(3):411–436 (2006).

—. "An adaptive sequential Monte Carlo method for approximate Bayesian computation." *Statistics and Computing* (2011). To appear.

Dempster, A. P., Laird, N. M., and Rubin, D. B. "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society B*, 39(1):1–38 (1977).

Doucet, A., Godsill, S., and Andrieu, C. "On sequential Monte Carlo sampling methods for Bayesian filtering." *Statistics and Computing*, 10(3):197–208 (2000).

Doucet, A. and Johansen, A. M. "A Tutorial on Particle Filtering and Smoothing: Fifteen years later." In Crisan, D. and Rozovsky, B. (eds.), *Handbook of Nonlinear Filtering*. Oxford University Press (2008).

Druilhet, P. and Marin, J.-M. "Invariant HPD credible sets and MAP estimators." *Bayesian Analysis*, 2(4):681–692 (2007).

Dutta, S. and Bhattacharya, S. "Markov Chain Monte Carlo Based on Deterministic Transformations." (2011). ArXiv:1106.5850.

Edwards, J. *A treatise on the integral calculus: with applications, examples and problems*, volume 2. Macmillan and co., ltd. (1922).

Edwards, R. G. and Sokal, A. D. "Generalization of the Fortuin-Kasteleyn-Swendsen-Wang representation and Monte Carlo algorithm." *Physical Review D*, 38(6):2009–2012 (1988).

Fan, J. and Li, R. "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties." *Journal of the American Statistical Association*, 96:1348–1360 (2001).

Figueiredo, M. A. T. "Adaptive Sparseness for Supervised Learning." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1150–1159 (2003).

Flynn, M. "Very high-speed computing systems." *Proceedings of the IEEE*, 54(12):1901–1909 (1966).

Fridley, B. "Bayesian variable and model selection methods for genetic association studies." *Genetic Epidemiology*, 33(1):27–37 (2009).

Friedman, J., Hastie, T., and Tibshirani, R. "Sparse inverse covariance estimation with the graphical lasso." *Biostatistics*, 9:432–441 (2008).

Friedrichs, M. S., Eastman, P., Vaidyanathan, V., Houston, M., Legrand, S., Beberg, A. L., Ensign, D. L., Bruns, C. M., and Pande, V. S. "Accelerating molecular dynamic simulation on graphics processing units." *Journal of Computational Chemistry*, 30(6):864–872 (2009).

Fu, Y. and Li, W. "Estimating the age of the common ancestor of a sample of DNA sequences." *Molecular Biology and Evolution*, 14(2):195–199 (1997).

Garrigues, P. J. "Sparse Coding Models of Natural Images: Algorithms for Efficient Inference and Learning of Higher-Order Structure." Ph.D. thesis, University of California, Berkeley (2009).

Gelfand, A. "Sampling-based approaches to calculating marginal densities." *Journal of the American Statistical Association*, 85(410):398–409 (1990).

Gelman, A., Jakulin, A., Pittau, M., and Su, Y.-S. "A weakly informative default prior distribution for logistic and other regression models." *Annals of Applied Statistics*, 2:1360–1383 (2008).

Geman, S. and Geman, D. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741 (1984).

George, E. I. and McCulloch, R. E. "Variable selection via Gibbs sampling." *Journal of the American Statistical Association*, 88:881–889 (1993).

Geyer, C. J. "Markov chain Monte Carlo maximum likelihood." In Kerigamas, E. (ed.), *Computing Science and Statistics: Proceedings of 23rd Symposium on the Interface Interface Foundation*, 156–163. Fairfax Station (1991).

Gordon, N., Salmond, D., and Smith, A. "Novel approach to nonlinear/non-Gaussian Bayesian state estimation." *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113 (1993).

Gramacy, R., Samworth, R., and King, R. "Importance tempering." *Statistics and Computing*, 20(1):1–7 (2010).

Green, P. "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination." *Biometrika*, 82(4):711 (1995).

Green, P. and Mira, A. "Delayed rejection in reversible jump Metropolis-Hastings." *Biometrika*, 1035–1053 (2001).

Grelaud, A., Robert, C., and Marin, J. "ABC methods for model choice in Gibbs random fields." *Comptes Rendus Mathematique*, 347(3-4):205–210 (2009).

Griffin, J. and Brown, P. "Inference with normal-gamma prior distributions in regression problems." *Bayesian Analysis*, 5:171–188 (2010a).

Griffin, J. E. and Brown, P. J. "Bayesian adaptive lassos with non-convex penalization." Technical report, IMSAS, University of Kent (2007).

—. "Inference with normal-gamma prior distributions in regression problems." *Bayesian Analysis*, 5(1):171–188 (2010b).

Hastie, T., Tibshirani, R., and Friedman, J. *The elements of statistical learning: data mining, inference, and prediction*. Springer Verlag (2009).

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

Higham, N. J. "The accuracy of floating point summation." *SIAM Journal on Scientific Computing*, 4(4):783–799 (1993).

—. *Accuracy and stability of numerical algorithms (second edition)*. SIAM (2002).

Hoggart, C. J., Whittaker, J. C., De Iorio, M., and Balding, D. J. "Simultaneous analysis of all SNPs in genome-wide and re-sequencing association studies." *PLOS Genetics*, 4:e1000130 (2008).

Hukushima, K. and Nemoto, K. "Exchange Monte Carlo Method and Application to Spin Glass Simulations." *Journal of the Physical Society of Japan*, 65:1604–1608 (1996).

Jarzynski, C. "Nonequilibrium equality for free energy differences." *Physical Review Letters*, 78(14):2690 (1997).

Jasra, A., Holmes, C. C., and Stephens, D. A. "Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling." *Statistical Science*, 50–67 (2005).

Jasra, A., Stephens, D. A., and Holmes, C. C. "On population-based simulation for static inference." *Statistics and Computing*, 17(3):263–279 (2007).

Jermyn, I. H. "Invariant Bayesian estimation on manifolds." *Annals of Statistics*, 33(2):583–605 (2005).

Kendall, W. S. "Notes on Perfect Simulation." In Kendall, W. S., Liang, F., and Wang, J. S. (eds.), *Markov chain Monte Carlo: innovations and applications*, 93–146. World Scientific (2005).

Kindermann, R., Snell, J., and Society, A. M. *Markov random fields and their applications*. American Mathematical Society (1980).

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

Klaas, M., de Freitas, N., and Doucet, A. "Toward Practical N^2 Monte Carlo: the Marginal Particle Filter." In *Proceedings of the Twenty-First Annual Conference on Uncertainty in Artificial Intelligence*, 308–315 (2005).

Kontoghiorghes, E. J. (ed.). *Handbook of Parallel Computing and Statistics*. Chapman & Hall (2006).

L'Ecuyer, P. "Good parameter sets for combined multiple recursive random number generators." *Operations Research*, 47(1):159–164 (1999).

L'Ecuyer, P., Chen, E. J., and Kelton, W. D. "An object-oriented random-number package with many long streams and substreams." *Operations Research*, 50(6):1073–1075 (2002).

Liu, J., Liang, F., and Wong, W. "The multiple-try method and local optimization in Metropolis sampling." *Journal of the American Statistical Association*, 95(449):121–134 (2000).

Liu, J. and West, M. "Combined parameter and state estimation in simulation-based filtering." In Doucet, A., de Freitas, N., and Gordon, N. (eds.), *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York (2000).

Liu, J. S. *Monte Carlo Strategies in Scientific Computing*. Springer (2001).

Liu, J. S. and Chen, R. "Blind Deconvolution via Sequential Imputations." *Journal of the American Statistical Association*, 90:567–576 (1995).

Mak, C. "Stochastic potential switching algorithm for Monte Carlo simulations of complex systems." *Journal of Chemical Physics*, 122:214110 (2005).

Marjoram, P., Molitor, J., Plagnol, V., and Tavaré, S. "Markov chain Monte Carlo without likelihoods." *Proceedings of the National Academy of Sciences*, 100(26):15324 (2003).

Marsaglia, G. "Xorshift RNGs." *Journal of Statistical Software*, 8(14):1–6 (2003).

McDonald, J. and Newey, W. "Partially Adaptive Estimation of Regression Models via the Generalized T Distribution." *Econometric Theory*, 4(3):428–457 (1988).

McLachlan, G. and Peel, D. *Finite Mixture Models*. New York: Wiley (2000).

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. "Equation of state calculations by fast computing machines." *Journal of Chemical Physics*, 21(6):1087 (1953).

Metropolis, N. and Ulam, S. "The Monte Carlo method." *Journal of the American Statistical Association*, 44(247):335–341 (1949).

Mira, A. "On Metropolis-Hastings algorithms with delayed rejection." *Metron*, 59(3-4):231–241 (2001).

Møller, J., Pettitt, A., Reeves, R., and Berthelsen, K. "An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants." *Biometrika*, 93(2):451 (2006).

Murphy, K. P. *Machine Learning: a Probabilistic Perspective*. MIT Press (2012). To be published.

Murray, I. "Advances in Markov chain Monte Carlo methods." Ph.D. thesis, University College London (2007).

Murray, I., Ghahramani, Z., and MacKay, D. "MCMC for doubly-intractable distributions." In *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence*, 359–366. Citeseer (2006).

Neal, P. and Roberts, G. "Optimal scaling for partially updating MCMC algorithms." *Annals of Applied Probability*, 16(2):475–515 (2006).

Neal, R. M. "Annealed Importance Sampling." *Statistics and Computing*, 11(2):125–139 (2001).

—. "MCMC Using Ensembles of States for Problems with Fast and Slow Variables such as Gaussian Process Regression." (2011). ArXiv:1101.0387.

Nummelin, E. and Tweedie, R. L. "Geometric ergodicity and R-positivity for general Markov chains." *Annals of Probability*, 404–420 (1978).

Obozinski, G., Taskar, B., and Jordan, M. "Multi-task feature selection." Technical report, Department of Statistics, University of California, Berkeley (2006).

Park, T. and Casella, G. "The Bayesian Lasso." *Journal of the American Statistical Association*, 103:681–686 (2008).

Peskun, P. "Optimum Monte-Carlo sampling using Markov chains." *Biometrika*, 60(3):607 (1973).

Peters, G., Fan, Y., and Sisson, S. "On sequential Monte Carlo, partial rejection control and approximate Bayesian computation." (2008). ArXiv:0808.3466.

Pitt, M. K. and Shephard, N. "Time-Varying Covariances: A Factor Stochastic Volatility Approach." In Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M. (eds.), *Bayesian Statistics 6*, 547–570. Oxford University Press (1999).

Predescu, C., Predescu, M., and Ciobanu, C. V. "The incomplete beta function law for parallel tempering sampling of classical canonical systems." *Journal of Chemical Physics*, 120(9):4119–4128 (2004).

Pritchard, J., Seielstad, M., Perez-Lezaun, A., and Feldman, M. "Population growth of human Y chromosomes: a study of Y chromosome microsatellites." *Molecular Biology and Evolution*, 16(12):1791–1798 (1999).

Robert, C. and Casella, G. "A Short History of Markov Chain Monte Carlo: Subjective Recollections from Incomplete Data." *Statistical Science*, 26(1):102–115 (2011).

Robert, C. P. and Casella, G. *Monte Carlo Statistical Methods (Second Edition)*. New York: Springer-Verlag (2004).

Roberts, G. and Rosenthal, J. "Geometric ergodicity and hybrid Markov chains." *Electronic Communications in Probability*, 2(2):13–25 (1997).

—. "General state space Markov chains and MCMC algorithms." *Probability Surveys*, 1:20–71 (2004).

—. "Harris recurrence of Metropolis-within-Gibbs and trans-dimensional Markov chains." *Annals of Applied Probability*, 16(4):2123–2139 (2006).

—. "Examples of adaptive MCMC." *Journal of Computational and Graphical Statistics*, 18(2):349–367 (2009).

Rosenthal, J. S. "Parallel computing and Monte Carlo algorithms." *Far East Journal of Theoretical Statistics*, 4:207–236 (2000).

Sisson, S., Fan, Y., and Tanaka, M. "Sequential monte carlo without likelihoods." *Proceedings of the National Academy of Sciences*, 104(6):1760 (2007).

Sohn, A. "Parallel N-ary speculative computation of simulated annealing." *Parallel and Distributed Systems, IEEE Transactions on*, 6(10):997 –1005 (1995).

Stephens, M. and Balding, D. "Bayesian statistical methods for genetic association studies." *Nature Reviews Genetics*, 10(10):681–690 (2009).

Stone, J. E., Phillips, J. C., Freddolino, P. L., Hardy, D. J., Trabuco, L. G., and Schulten, K. "Accelerating Molecular Modeling Applications with Graphics Processors." *Journal of Computational Chemistry*, 28(16):2618–2640 (2007).

Storvik, G. "On the Flexibility of Metropolis-Hastings Acceptance Probabilities in Auxiliary Variable Proposal Generation." *Scandinavian Journal of Statistics*, 38:342–358 (2011).

Suchard, M. A. and Rambaut, A. "Many-Core Algorithms for Statistical Phylogenetics." *Bioinformatics*, 25(11):1370–1376 (2009).

Sutter, H. "The free lunch is over." *Dr. Dobbs Journal*, 30(3) (2005).

Swendsen, R. and Wang, J. "Nonuniversal critical dynamics in Monte Carlo simulations." *Physical Review Letters*, 58(2):86–88 (1987).

Tavare, S., Balding, D., Griffiths, R., and Donnelly, P. "Inferring coalescence times from DNA sequence data." *Genetics*, 145(2):505–518 (1997).

Tibshirani, R. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society B*, 58:267–288 (1996).

—. "Regression shrinkage and selection via the lasso: a retrospective." *Journal of the Royal Statistical Society B*, 73(3):273–282 (2011).

Tierney, L. "Markov Chains for Exploring Posterior Distributions." *Annals of Statistics*, 22(4):1701–1762 (1994).

—. "A note on Metropolis-Hastings kernels for general state spaces." *Annals of Applied Probability*, 8(1):1–9 (1998).

Tierney, L. and Mira, A. "Some adaptive Monte Carlo methods for Bayesian inference." *Statistics in Medicine*, 18(1718):2507–2515 (1999).

Ulam, S., Richtmyer, R. D., and Von Neumann, J. "Statistical methods in neutron diffusion." Technical Report LAMS-551, Los Alamos National Laboratory (1947).

Walker, S. G. and Gutiérrez-Peña, E. "Robustifying Bayesian procedures." In *Bayesian Statistics 6*, 685–710. New York: Oxford University Press (1999).

Wang, X. "Volumes of generalized unit balls." *Mathematics Magazine*, 78(5):390–395 (2005).

Warnes, G. "The Normal Kernel Coupler: An adaptive Markov Chain Monte Carlo method for efficiently sampling from multi-modal distributions." Technical Report 395, Department of Statistics, University of Washington (2001).

West, M. "On scale mixtures of normal distributions." *Biometrika*, 74:646–648 (1987).

Williams, D. *Probability with Martingales*. Cambridge University Press (1991).

Wilson, M., Iversen, E. S., Clyde, M., Schmidler, S. C., and Schildkraut, J. M. "Bayesian Model Search and Multilevel Inference for SNP Association Studies." *Annals of Applied Statistics*, 4:1342–1364 (2010).

Yuan, M. and Lin, Y. "Model selection and estimation in regression with grouped variables." *Journal of the Royal Statistical Society B*, 68(1):49–67 (2006).

—. "Model selection and estimation in the Gaussian graphical model." *Biometrika*, 94:19–35 (2007).

Zou, H. "The Adaptive Lasso and Its Oracle Properties." *Journal of the American Statistical Association*, 101:1418–1429 (2006).

Zou, H. and Li, R. "One-step sparse estimates in nonconcave penalized likelihood models." *Annals of Statistics*, 36:1509–1533 (2008).