On Population-Based Simulation for Static Inference

By AJAY JASRA[1], DAVID A. STEPHENS[2]

Department of Mathematics, Imperial College London, SW7 2AZ, London, UK

ajay.jasra@imperial.ac.uk d.stephens@imperial.ac.uk

AND CHRISTOPHER C. HOLMES[3]

Department of Statistics, University of Oxford, OX1 3TG, Oxford, UK

cholmes@stats.ox.ac.uk

[1]A. Jasra is Ph.D student at Imperial College London

[2]D. Stephens is senior lecturer in statistics at Imperial College London

[3]C. Holmes is lecturer in statistics at the University of Oxford

Abstract

In this paper we present a review of *population-based simulation* for static inference problems. Such methods can be described as generating a collection of random variables $\{X_N\}$ in parallel in order to simulate from some target (or potentially sequence of targets) π. Population-based simulation is important as many challenging problems in applied statistics cannot be dealt with successfully by conventional Markov chain Monte Carlo (MCMC) methods (Robert & Casella, 2004). We summarize population-based MCMC (Geyer, 1991; Liang & Wong, 2001) and sequential Monte Carlo samplers (SMC) (Del Moral, Doucet & Peters, 2005), providing a comparison of the approaches. We give numerical examples from Bayesian mixture modelling (Richardson & Green, 1997).

Some key words: Markov chain Monte Carlo, Sequential Monte Carlo, Bayesian mixture models, Adaptive methods.

1. Introduction

A common problem in Bayesian statistics is that of evaluating an expectation of a function h with respect to a probability density π:

$$\mathbb{E}_\pi\big[h(X)\big] \quad = \quad \int_E h(x)\pi(x)dx \tag{1.1}$$

where h is integrable and π is a density with respect to measure dx on measurable space $(E, \mathcal{E})$. Since E is often of high dimension (1.1) can seldom be computed analytically, via (deterministic) numerical methods or by using Monte Carlo integration with independent sampling from π.

A solution to this problem is provided by Markov chain Monte Carlo (Metropolis-Hastings (MH) kernels (Metropolis, Rosenbluth, Rosenbluth, Teller & Teller, 1953; Hastings, 1970)) to generate samples from an ergodic Markov kernel $K : E \times \mathcal{E} \to [0, 1]$ with invariant distribution π and use the estimate:

$$S_h(x_{1:N}) \quad = \quad \frac{1}{N}\sum_{i=1}^{N} h(x_i) \tag{1.2}$$

where $x_{1:N} = (x_1, \ldots, x_N)$ (resp. $X_{1:N} = (X_1, \ldots, X_N)$) and $x_{1:N}$ have been drawn from K. Note that $dx_{1:N} \triangleq dx_1 \times \cdots \times dx_N$ and throughout this paper all probability measures are absolutely continuous with respect to some $\sigma-$finite measure dx.

However, in many modern areas of applied statistics, for example gene clustering (Heard, Holmes & Stephens, 2005) and admixture modelling in population genetics (Pritchard, Stephens & Donnelly, 2001), conventional MCMC methods are not able to correctly traverse the state space. This is because the statistical models used to analyze such data are often highly complex, inducing multimodal target densities, and leading to poorly mixing MCMC algorithms. We illustrate this problem with the following example from Bayesian mixture modelling.

1·1 *Illustrative Example: Mixture Modelling*

Throughout this article we use Bayesian mixture modelling to demonstrate the algorithms we review. Mixture models are typically used to model heterogeneous data, or as a simple means of density estimation; see McLachlan & Peel (2000) for an overview.

Let $y_1, \ldots, y_m$ denote observed data $y_l \in \mathbb{R}$, $l = 1, \ldots, m$. We assume that the y_l are i.i.d with density:

$$p(y_l|\boldsymbol{\phi}, \mathbf{w}, k) = \sum_{j=1}^{k} w_j f(y_l; \boldsymbol{\phi}_j)$$

where $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k)$ are component specific parameters, the weights $\mathbf{w} = (w_1, \ldots, w_k)$ are such that $\sum_{j=1}^{k} w_j = 1$, $w_j \geq 0\ \forall j$, $p(\cdot)$ denotes an arbitrary probability mass/density function and $f(\cdot)$ is the component density. We take $f(\cdot)$ to be normal, $\mathcal{N}(\mu, \lambda^{-1})$ where (μ, λ) are the location and scale parameters respectively. The priors (which are the same for each component $j = 1, \ldots, k$) have the following hierarchical structure (as in Richardson & Green (1997)): $\mu_j \sim \mathcal{N}(\xi, \kappa^{-1})$, $\lambda_j|\beta \sim \mathcal{G}a(\alpha, \beta)$, $\beta \sim \mathcal{G}a(g, h)$ and $\mathbf{w} \sim \mathcal{D}(\delta)$. Our notation is such that: $\mathcal{D}(\delta)$ is the symmetric Dirichlet distribution with parameter δ and $\mathcal{G}a(\alpha, \beta)$ is the gamma distribution, shape α, scale β. If k is unknown we assume $k \sim \mathcal{U}_{\{1,\ldots,k_{\max}\}}$ where $\mathcal{U}_{\{1,\ldots,k_{\max}\}}$ is the uniform distribution on the integers $1, \ldots, k_{\max}$ with $k_{\max}$ known. The prior parameter setting is as Richardson & Green (1997) and we refer the reader to that paper. Note that one feature of this mixture model is that due to the invariance of the posterior distribution to permutation of the labels of the parameters, it features $k!$ symmetric modes (given that there are k components in the mixture model); see Jasra, Homes & Stephens. (2005a) for a review.

To illustrate the aspects discussed above we consider the Hidalgo stamp data in Figure 1 (a); see McLachlan & Peel (2000) for details. We analyze these data with the above model (k unknown). We ran a reversible jump (Green, 1995) MCMC algorithm (detailed in Section 2·7) for half a million iterations and plotted every 100^{th} value of k in Figure 1 (b).

In Figure 1 (b) we see that the target density appears to have two distinct modes: at 7-15 components and 3-5 components. The sampler has behaved reasonably, in the sense that it is able to jump between these modes, but cannot regularly do so.

This situation is typical of many standard MCMC kernels when attempting to sample multimodal densities. Although the difficulties presented here may be alleviated using adaptive methods (Andrieu & Robert, 2001) (that is, to update the transition kernel on the basis of realized values of the chain, for example, we may record the modes and build a proposal based upon these), there can be examples in which the kernels mix so badly that such adaptive strategies cannot always be expected to work. Due to our experience with more advanced single chain MCMC methods (such as tempered transitions (Neal, 1996) and delayed rejection (Green & Mira, 2001)), it seems clear that alternative methods are needed.

1·2 *Population-based Simulation*

Borrowing the term 'population-based' from Iba (2000), we define a population-based simulation method as one which, instead of sampling a single (independent/dependent) sample, generates a collection of samples in parallel. We distinguish two types population algorithm; one which relies solely upon MCMC methodology and another which uses importance sampling/resampling ideas (Doucet, De Freitas & Gordon 2001).

The first method, which we refer to as population-based MCMC, works by building a new target $\pi^*(x_{1:N})dx_{1:N}$ on the product space $(E^N, \bigvee_{i=1}^N \mathcal{E})$ and embeds the target density (in some way to be discussed later) of interest π into it. To our knowledge, this method was originally developed by Geyer (1991) who defined a new target density $\pi^*(x_{1:2}) = \pi(x_1)\pi_1(x_2)$, with π_1 different (but related) to π and swapped x_1 and x_2 between invariant distributions. This approach was independently developed by Hukushima & Nemoto (1996) who described this as 'Exchange Monte Carlo'. Another population method, adaptive direction sampling, was devised by Gilks, Roberts & George (1994). Further advances came in Liang & Wong (2001) who attempted to produce genetic algorithm (GA) type moves to improve the mixing of the Markov chains (the method was termed 'evolutionary Monte Carlo'). Liu (2001) provides some further background.

The second approach is sequential Monte Carlo, exemplified by the SMC sampler method of Del Moral et al. (2005). Sequential Monte Carlo methods have a rich history, originating from the initial work of Hammersely & Morton (1954); see Doucet, Godsill & Andrieu (2000) for a historical review. SMC methods were constructed to sample from a sequence of related target distributions, using importance sampling to reweight the population of samples (or *particles*) from the previous target density and resampling to allow the samples to interact. Such simulation cannot be achieved using MCMC. However, as noted by Chopin (2002) and Del Moral & Doucet (2003), SMC methods may also be used to simulate from a single, static target. As a result, SMC is an alternative population-based simulation method. We note that there are many SMC methods appropriate for static inference such as annealed importance sampling (Neal, 2001), resample-move (Gilks & Berzuini, 2001), the sequential particle filter of Chopin (2002) and population Monte Carlo (Cappé, Gullin, Marin & Robert, 2004) but since SMC sampler approach contains all of these methods as a special case we concentrate upon this.

1·3 *Structure and Objectives of the Paper*

In this paper we provide a review of population-based simulation methods and a tutorial on how to use them. Our intent is to convince applied statisticians that population-based simulation provides an implementable and important method, which can often be used in situations for which no other methods work.

This paper is structured as follows. In Section 2 we review population-based MCMC, returning to the Hidalgo stamp example. In Section 3 we consider SMC samplers, detailing some important implementation issues. In Section 4 we provide a comparison of these methodologies. Finally, in Section 5 we conclude.

2. Population-based MCMC

2·1 *The Method*

Population-based MCMC may be described as follows. Suppose that we are interested in sampling from a target density π, which is likely to be multimodal, we define a new target density:

$$\pi^*(x_{1:N}) = \prod_{i=1}^{N} \pi_i(x_i) \tag{2.1}$$

where we assume that $\pi \equiv \pi_i$ for at least one $i = 1, \ldots, N$.

In order to construct a valid MCMC algorithm, we need a (time homogeneous) Markov kernel that is π^*-irreducible, aperiodic and admits π^* as its invariant distribution. This is easily achieved by considering the target as having vector components $(x_1, \ldots, x_N)$ as in hybrid MCMC; see Robert & Casella (2004) (e.g. Metropolis-within-Gibbs updates of x_1, x_2 etc.). Note that (1.2) is computed by using samples from the chain with target of interest.

The basic idea of extending the state space is that by using the information in the population more global moves may be constructed resulting in faster mixing MCMC algorithms.

We now discuss some population moves and the sequence of distributions.

2·2 *Population Moves*

We now describe the different types of population moves that have been used. We use the GA terminology of Liang & Wong (2001) and Del Moral & Doucet (2003) (for example).

Mutation. This move seeks to update a single member of the population via a Markov kernel. That is, $X_i' \sim K(x_i, \cdot)$ (where $'$ denotes a new value of the particle). For example, we may update all chains via:

$$K(x_{1:N}, dx'_{1:N}) = \prod_{i=1}^{N} K_i(x_i, dx'_i)$$

where K_i is a Markov kernel that is π_i-invariant. The purpose of this move is to allow local exploration of the state space, as well as ensuring the required irreducibility of the algorithm
Exchange. The standard way to swap information between chains is to use the exchange move. This is a Metropolis-Hastings move that proposes to swap the value of two chains and is accepted with probability $\min\{1, A\}$

$$A = \frac{\pi_i(x_j)\pi_j(x_i)}{\pi_i(x_i)\pi_j(x_j)} \tag{2.2}$$

where we have assumed that we have selected both chains with equal probability and the labelling of the proposed state of the chain is with respect to the current state. This move can easily be shown to satisfy detailed balance, even in the trans-dimensional case; see Jasra (2005) for example. The idea is to move information between the population. For example, we may try to swap information between chains that have similar target distributions. One interesting approach, suggested by Green & Mira (2001), is to use delayed rejection to propose a bold swap (e.g. between chains that are very different in some sense) and, if rejected, a more timid swap (e.g. between chains that are similar in some sense).
Crossover. This idea was introduced (in the MCMC literature) by Liang & Wong (2001). Suppose $x_i = (x_{1i}, \dots, x_{pi})\ \forall i$, then the crossover selects a position in the vector to crossover information. That is, if we propose to crossover the l^{th} position in the vector for chains i and j we have:

$$\begin{aligned} x'_i &= (x_{1i}, \dots, x_{(l-1)i}, x_{lj}, \dots, x_{pj}) \\ x'_j &= (x_{1j}, \dots, x_{(l-1)j}, x_{li}, \dots, x_{pi}). \end{aligned}$$

This move is accepted with probability $\min\{1, A\}$, with A as for (2.2) with appropriate change of notation and assuming all choices (chains, crossover position) are made with uniform probability. We have found that this move can be reasonably efficient (in terms of acceptance rate, often in the range of 2-3% for quite challenging problems) if we do not attempt to crossover too much information.
Snooker Moves. Gilks et al. (1994) use the idea of moving population members towards each other. That is, it is expected that some population members should have high (original) target

density and thus an intelligent move is to propose values close to other members of the population. We note that we do not want this move to occur so often so that all chains are close together, that is, reducing the *diversity* of the population. This is of importance, as the idea of population-based simulation is to use the extra information of the particles to improve the mixing of the algorithm. If all population members are stuck in a single mode say, then the advantage is lost.

We note that further move types may be found in Goswami & Liu (2005). They use a method termed target orientated evolutionary Monte Carlo, where the idea is to produce more updates of the original target π (it is assumed only one copy lies in π^*). Whilst, from a CPU time perspective, this seems a good idea, it is not clear that this will lead to faster theoretical convergence to π^*.

2·3 *Sequence of Distributions*

The possible types of distributions that have been used are as follows. Note that any of these methods may be used for SMC.

Identical. One simple idea is to take $\pi_i \equiv \pi \ \forall i$. This approach is used by Warnes (2001) and Mengersen & Robert (2002). We note that for this approach to be effective, the kernels used to sample the original target must mix reasonably quickly, or some additional approach must be used to improve exploration of the target. For example, Warnes (2001) updates a single population member x_i using a proposal that is a normal kernel estimate of π, fitted to the other particles. Robert & Casella (2004) note that this may not work well for high dimensional targets due to the poor performance of density estimation in large spaces. Mengersen & Robert (2003) present an interesting approach where particles are encouraged to avoid each other, leading to diversity in the population.

Tempered. Suppose that

$$\int_E \big[\pi(x)\big]^{\zeta} dx \quad < \quad \infty$$

for $\zeta \in \Xi$ with Ξ a set of values on $(0,1]$. Then we may take: $\pi_i(x) \propto \big[\pi(x)\big]^{\zeta_i}$ $\zeta_i \in \Xi$ and suppose that $\zeta_i = 1$ for at least one $i = 1, \dots, N$. This approach is used by Liang & Wong (2001) and Jasra, Stephens & Holmes (2005b). The idea is that the distributions at high *temperatures*, that is, ζ close to zero, are easily sampled and can improve the mixing entire algorithm; this was the idea of Geyer (1991). Selecting the ζ is not always easy. The intuition is that, given a number of distributions, we want the temperatures to be high enough to allow fast movement around the state space, but at the same time, that there are enough chains which hold information related to the original target density (i.e. as noted by Neal (2001) we want the evolution of densities from the flattest to the target to be 'smooth' in some sense). Liu (2001) states that temperatures should be set so that the exchange move is accepted about half of the time. Further discussion can be found in Goswami & Liu (2005). A related approach, suggested by Gelman & Meng (1998), is to take

$$\pi_i(x) \quad \propto \quad \big[\pi(x)\big]^{\zeta_i}\big[p(x)\big]^{1-\zeta_i}$$

where p is some probability density that may be sampled from and $\int_E \big[\pi(x)\big]^{\zeta_i}\big[p(x)\big]^{1-\zeta_i}dx < \infty\forall i$.

Data Point Tempered. In the situation that we work with a target density related to observed data, some of the densities in the population may only include some subset of the data. For example in Chopin (2002), we seek to draw inference from a posterior distribution $\pi(\theta|y)$ and suppose $N = m$ (the number of data-points). Chopin (2002) took the sequence of densities to be

$$\pi_i(\theta|y_{1:i}) \quad \propto \quad p(y_{1:i}|\theta)\pi(\theta)$$

we note that the densities may change by 'batches' of data. To our knowledge, this has not been used in the MCMC literature (however is often used in SMC; see the next Section). Our experience is that this approach does not often work well for population-based MCMC, unless a careful choice of data partitioning is made; we discuss this more in Section 3·4.

Stratified. Suppose $A_1, \ldots, A_{N-1}$ form a partition of E, $A_i \in \mathcal{E}\ \forall i = 1, \ldots, N-1$, $\int_{A_i} \pi(x)dx > 0$ and take

$$\pi^*(x_{1:N}) \quad \propto \quad \pi(x_N)\prod_{i=1}^{N-1}\pi(x_i)\mathbb{I}(x \in A_i)$$

where $\mathbb{I}(\cdot)$ is the indicator function. The objective is that in each E_i we may be able to construct a Markov kernel that mixes quickly across the space, so by constraining chains to lie in different regions we can correctly sample the target. The difficulty lies in determining the partition. One application for which this may not be so problematic is trans-dimensional simulation; see Atachade & Liu (2004). Additionally, it is not always easy to make the chains interact. For example, Jasra et al. (2005b) use a special exchange that only allows jumps between the same region (with some chains constrained to lie in overlapping sets), but this cannot always be expected to work (since we need be able to jump between regions).

An idea that may work well is to use the population (which is stratified) to produce a proposal that approximates the target. That is, to use adaptive methods to construct a proposal (note that if this is done in the MCMC framework then if adaptation is only done finitely often then convergence is still achieved; see Roberts & Rosenthal (2005)).

A related but differing approach is used by Kou, Zhou & Wong (2005). They adopt the sequence of densities:

$$\begin{aligned} \pi_i(x) &\propto \exp\{-\zeta_i g_i(x)\} \\ g_i(x) &= g(x) \vee G_i \\ g(x) &= -\log\{f(x)\} \end{aligned} \tag{2.3}$$

where $\pi(x) = \frac{1}{Z} f(x)$ $(z = \int_E f(x)dx)$, $\inf_{x\in E}\{g(x)\} \geq G_1 < \cdots < G_N < \infty$ is a stratification of the energy space. If used in a population MCMC framework this approach will lead to diverse samples with respect to the energy (which may not mean the samples are diverse with respect to the state space). We discuss the method of Kou et al. (2005) below.

We note that any of the densities mentioned above may be applied simultaneously. We have often found that combining tempered densities with stratified/partitioned chains leads to satisfactory performance. The identical approach is less useful in the MCMC framework, unless clever population moves may be formulated with reasonable computational cost.

2·4 *Number of Distributions*

The choice of the number of distributions is not always clear. Our guidance is as follows. An obvious constraint is being able to store the number of chains on a computer (i.e. having enough memory). Given this constraint, we want enough chains so that we have a lot of information to improve exploration around the space, but not too many chains so that convergence to π^* takes too long in terms of CPU time. Additionally, the complexity of the problem will determine whether a large or small number of chains are needed. For example, for high dimensional problems the target density is likely to be multimodal and very complex which suggests that a large number of chains are needed so that we can successfully traverse the state space.

We have often use the criterion that samples from the target of interest are reasonably similar for long runs of the algorithm. Kou et al. (2005) suggest that the number of chains (for equi-energy sampling) should be roughly proportional to the dimensionality of the target density.

2·5 *Equi-Energy Sampler*

One population method, which does not fall exactly into the class of population-based MCMC is the equi-energy sampler of Kou et al. (2005). This method generates a non-Markovian stochastic process with target densities (2.3). The method proceeds by sampling from a Metropolis-Hastings kernel with stationary distribution π_N. Once convergence is reached we store samples and start another 'chain' (after $C > 0$ steps of the π_N chain) which targets π_{N-1} and updates at each time step by either using a Metropolis-Hastings kernel or by proposing to exchange the current state of the chain with a value stored of the chain targeting π_N within the same energy band (i.e. if $x^i_{N-1} \in [G_l, G_{l+1}]$ at time i we propose to swap with a stored value in this band). The process continues until we target π_1 which is the target density of interest.

The advantage of this method over population-based MCMC discussed above is that we will retain information of where we have been and be able to make large moves between separated modes. This is at the cost of increased storage and that the method requires an estimate of the posterior mode is computed before simulation. Also a sensible partitioning of the energy space is required; see Kou et al. (2005) for details. We note, also, that adaptive methods may be used very naturally for population-based MCMC (given that it can be shown to be ergodic, see Section 5 for further discussion) which will be less computationally expensive than storing samples and potentially just as effective.

We believe that this method is very important and worthy of detailed consideration by applied statisticians. However, a substantial theoretical investigation is needed as there is limited discussion in Kou et al. (2005). Due to space constraints we do not consider this method further.

2·6 *A Typical Algorithm*

In Algorithm 2.1 we give a typical algorithm used in population-based MCMC: it may be used as a basic template to build more complex algorithms. Note that assuming $\bigvee_{i=1}^{N} \mathcal{E}$ is countably generated the algorithm will converge to π^* (in total variation distance) (see Theorem 4 of Roberts & Rosenthal (2004)).

2·7 *Illustrative Example*

To see that population-based simulation can provide improvements over standard MCMC methods, we return to the Hidalgo stamp data example.

The algorithm followed the framework of Algorithm 2.1, except that the only population move used was an exchange move with delayed rejection. The move proposes to swap any pair of chains (chosen with uniform probability) and if rejected proposes to swap chains that are adjacent in terms of the temperature parameter; see Jasra et al. (2005b) for further details. The mutation step is a composition of the following kernels:

- Update β via a Gibbs kernel.
- Update $\boldsymbol{\mu}$ via a MH kernel with additive normal random walk proposal.

```
Algorithm 2.1 A population-based MCMC algorithm.
0. (INTIALIZATION)
    • Initialize the chain $x_{1:N}$, $X_i \sim \nu$, $\nu$ a probability density.
    • For $t = 1, \ldots, T$ sweep over the following:
1. (MUTATION)
    • Select a chain $i$ with a fixed (time homogeneous) probability and then update $x_i$ using a
      $\pi_i-$irreducible, aperiodic Markov kernel, which admits $\pi_i$ as its invariant distribuion.
2. Make a random choice between performing steps 3 or 4.
3. (CROSSOVER)
    • Perform the crossover move in Section 2·2.
4. (EXCHANGE)
    • Perform the exchange move in Section 2·2.
```

- Update $\boldsymbol{\lambda}$ via a MH kernel with multiplicative log-normal random walk proposal.
- Update $\mathbf{w}$ via a MH kernel with additive normal random walk proposal on the logit scale.
- Update $(\boldsymbol{\mu}, \boldsymbol{\lambda}, \mathbf{w}, k)$ via a birth/death reversible jump kernel. The move is similar to that of Richardson & Green (1997).

We optimized the Metropolis-Hastings kernels used in fixed dimensional simulation (in terms of acceptance rates) for the respective distributions. We took a population size of $N = 25$ and tempered densities (only powering up the likelihood), the temperature parameters were decreasing (from 1) by 0.04 for each distribution.

We ran a population-based reversible jump algorithm for 1.2 million sweeps and we allowed a 1 million iteration burn-in and thinned to every 20th sample. The code (programmed in C) took approximately 12 minutes to run on a Pentium 4, 3 Ghz machine (vs 4 minutes for 500000 iterations of the reversible jump algorithm). The exchange move was accepted 28% of the time at the first stage of the delayed rejection and 79% at the second. Based upon the criterion of acceptance rate, the temperature parameter settings are appropriate for this problem.

In Figure 2 we can observe the sampled k. In comparison to Figure 1 (b) we see that the population-based MCMC algorithm has been able to traverse the state space. The use of a large population, tempered densities and exchange moves has lead to improved exploration of the space.

3. Sequential Monte Carlo Samplers

3·1 *The Method*

We now describe the SMC sampler method of Del Moral et al. (2005). Consider an identical situation to population-based MCMC, that is, we have a sequence of related densities $\{\pi_n\}$, of which at least one is the target of interest (denote this π_n, note that we use n as opposed to N, since this represents a time parameter instead of the number of particles (note that the number of particles is substantially bigger for SMC methods when compared to MCMC). The method begins by generating each particle $x_1^{(i)}$, $i = 1, \dots, N$ from an initial distribution ν say then computing the importance weights:

$$W_1(x_1^{(i)}) = \frac{\pi_1(x_1^{(i)})}{\nu(x_1^{(i)})}$$

we note that we that do not require the normalizing constants of π_1 and ν as they cancel when calculating expectations. We may calculate expectations with respect to π_1 as:

$$\mathbb{E}_{\pi_1}\big[h(X)\big] \approx \frac{\sum_{i=1}^N W_1(x_1^{(i)})h(x_1^{(i)})}{\sum_{i=1}^N W_1(x_1^{(i)})}.$$

Now to produce samples for π_2 we sample each particle from a Markov kernel $X_2^{(i)}|x_1^{(i)} \sim K_1(x_1^{(i)}, \cdot)$ and reweight by using the incremental weights ω:

$$\begin{aligned} \omega_2(x_{1:2}^{(i)}) &= \frac{\pi_2(x_2^{(i)})\nu(x_1^{(i)})}{\pi_1(x_1^{(i)}) \int_E \nu(x)K_1(x, x_2^{(i)})dx} \\ W_2(x_{1:2}^{(i)}) &= W_1(x_1^{(i)})\omega_2(x_{1:2}^{(i)}). \end{aligned} \tag{3.1}$$

The problem is that the integral in (3.1) can seldom be calculated. To remove this problem, Del Moral et al. (2005) (and Neal (2001)) extend the state space to $(E \times E, \mathcal{E} \times \mathcal{E})$ and define new target density $\widetilde{\pi}_2(x_{1:2})$ such that:

$$\pi_2(x_2) = \int_E \widetilde{\pi}_2(x_{1:2})dx_1$$

i.e. it admits π_2 as its marginal. Then we may calculate (3.1) as:

$$\omega_2(x_{1:2}^{(i)}) = \frac{\widetilde{\pi}_2(x_{1:2}^{(i)})}{\pi_1(x_1^{(i)})K_1(x, x_2^{(i)})}.$$

We then take $\widetilde{\pi}_l$ to admit marginal π_l, $l = 2, \dots, n$. The algorithm continues by sampling from (potentially time inhomogeneous) Markov kernels $K_2, \dots, K_{n-1}$.

It is a well-known problem in sequential importance sampling (see Liu (2001) for example) that as the algorithm progresses the weights will degenerate to zero, except for a single particle which has weight approximately 1. (Note, that this will mean that this particle is most relevant among the current particles, but may have low target density overall). To deal with this, the method of resampling or *selection* is applied.

Resampling is a method which seeks to remove the lowest weighted particles, in the hope that future samples lie in regions of high target density, as well as reducing the CPU time spent on updating lowly weighted particles. One of the most simple methods of resampling is the multinomial approach, where particles are resampled with replacement from a multinomial distribution with probabilities equal to $W_t(x^{(i)})/\sum_{i=1}^{N} W_t(x^{(i)})$. The multinomial method will introduce unnecessary noise into the algorithm (that is the variance of the number of replicates of particle i can be reduced) so other resampling techniques are available; see Doucet et al. (2001) for further details.

3·2 *The Algorithm*

In Algorithm 3.1 we provide the basic sampling scheme. We note that the resampling criteria is based upon the effective sample size (ESS) (Liu, 2001). Resampling may occur at deterministic times, as opposed to that given in Algorithm 3.1.

The theoretical justification for an SMC algorithm is essentially asymptotic (in the number of particles); see Chopin (2004) for example. Note, also, that the selection/mutation format of the algorithm can be interpreted as Feynman-Kac path flow (Del Moral, 2004), which provides an elegant framework for the theoretical analysis of SMC methods.

3·3 *Specifying the Auxiliary Distributions*

When constructing an SMC sampler we need to specify the auxiliary distributions $\widetilde{\pi}_t$ $(t \geq 2)$. The approach used by Del Moral et al. (2005) is to set:

$$\widetilde{\pi}_t(x_{1:t}) = \pi_t(x_t)\prod_{j=2}^{t} L_{j-1}(x_j, x_{j-1}) \tag{3.2}$$

where L_{j-1} is a *backwards* Markov kernel. Thus the incremental weights become:

$$\omega_t(x_{1:t}^{(i)}) = \frac{\pi_t(x_t^{(i)})L_{t-1}(x_t^{(i)}, x_{t-1}^{(i)})}{\pi_{t-1}(x_{t-1}^{(i)})K_{t-1}(x_{t-1}^{(i)}, x_t^{(i)})}.$$

Del Moral et al. (2005) show that the optimal backwards kernel L_{t-1}^{opt} (in terms of minimizing

Algorithm 3.1 A sequential Monte Carlo sampler.

```
0.(INITIALIZATION)
    • Set $t = 1$.
    • For $i = 1, \ldots, N$ draw $X_1^{(i)} \sim \nu$.
    • Set
        $$W_1(x_1^{(i)}) = \frac{\pi_1(x_1^{(i)})}{\nu(x_1^{(i)})}.$$
Iterate steps 1 and 2.
1.(SELECTION)
    • If $\left((\sum_i (W_t(x_{1:t}^{(i)}))^2)/(\sum_j W_t(x_{1:t}^{(j)}))^2\right)^{-1} < L$ ($L$ is some user set threshold), resample the
      particles and set all weights equal to $1$.
2.(MUTATION)
    • Set $t = t + 1$, if $t = n + 1$ stop.
    • For $i = 1, \ldots, N$ draw $X_t^{(i)} \sim K_{t-1}(x_{t-1}^{(i)}, \cdot)$.
    • Reweight
        $$\omega_t(x_{1:t}^{(i)}) = \frac{\widetilde{\pi}_t(x_{1:t}^{(i)})}{\widetilde{\pi}_{t-1}(x_{1:t-1}^{(i)})K_{t-1}(x_{t-1}^{(t)}, x_t^{(i)})}$$
        $$W_t(x_{1:t}^{(i)}) = W_{t-1}(x_{1:t-1}^{(i)})\omega_t(x_{1:t}^{(i)}).$$
```

the variance of the importance weights) should be taken as:

$$\begin{aligned} L_{t-1}^{\text{opt}}(x_t, x_{t-1}) &= \frac{\nu_{t-1}(x_{t-1})K_{t-1}(x_{t-1}, x_t)}{\nu_t(x_t)} \\ \nu_t(x_t) &= \int \nu(x_1)K_1(x_1, x_2)\ldots K_{t-1}(x_{t-1}, x_t)dx_{1:t-1}. \end{aligned}$$

That is, the importance weights are of the form we started with (i.e. calculating an intractable integral). Since such a backwards kernel cannot be calculated (otherwise we would not have resorted to extending the state space), Del Moral et al. (2005) suggest:

$$L_{l-1}(x_t, x_{t-1}) = \frac{\pi_{t-1}(x_t)K_{t-1}(x_{t-1}, x_t)}{\int_E \pi_{t-1}(x_{t-1})K_{t-1}(x_{t-1}, x_t)dx_{t-1}}. \tag{3.3}$$

The usage of an MCMC kernel with invariant distribution π_{t-1} will mean that the integral in equation (3.3) can be easily calculated. However, we can use non-MCMC and even time adaptive kernels; see Cappé et al. (2004) for an example with the latter. For problems in mixture modelling, we have found that non-MCMC kernels are hard to construct so that they do not lead to fast impoverishment (importance weights degenerating to zero).

To select the densities, as long as $\pi_n \equiv \pi$, any of the choices in Section 2·3 may be chosen (assuming that the incremental weights are well-defined). We note that it is not always easy to specify the densities; we demonstrate this in the following example.

3·4 *Illustrative Example*

We now provide an example on how to construct and apply an SMC sampler. We use the mixture example of Section 1·1 along with the simulated data; 100 data points generated from an equally weighted mixture of four normal distributions with means at -3,0,3 and 6 respectively, and variance 0.55^2. We consider two tempering approaches: tempered densities and data point tempered.

3.4.1 Two Tempering Approaches: Algorithm Details

Tempered Densities. We take the target distributions to be:

$$\pi_t(\boldsymbol{\phi}, \mathbf{w}, \beta | y_{1:m}) \propto l(y_{1:m}; \boldsymbol{\phi}, \mathbf{w})^{\zeta_t} p(\boldsymbol{\phi}, \mathbf{w}, \beta) \quad t \in \{1, \ldots, n\}$$

where $l(\cdot)$ is the likelihood function, $\zeta_t \in [0, 1]$ is a temperature to be defined below (k is assumed to be fixed) and denote $\boldsymbol{\theta} = (\boldsymbol{\phi}, \mathbf{w}, \beta)$.

We will apply the MCMC kernels in Section 2·7 (no reversible jump kernel is needed) in the following way. At time even time points apply the Gibbs kernel. At odd time points (3 and upwards) apply the cycle of kernels for $\boldsymbol{\mu}$ etc. The initial distribution is the prior.

For the temperature parameters we take the sequence

$$\begin{aligned}\zeta_1 &= 0\\ \zeta_{2t} &= \frac{1}{n}+\zeta_{2(t-1)} \quad t=1,\dots,n\\ \zeta_{2t+1} &= \zeta_{2t} \quad t=1,\dots,n-1\end{aligned}$$

where $\zeta_0 \triangleq 0$. That is, we do not change the distribution, when applying the cycle updating $\boldsymbol{\mu}, \boldsymbol{\lambda}$ etc. This ensures regular updates at the cost of increasing the variance of the importance weights.

The backwards Markov kernels are taken to be the suboptimal choice (3.3). That is,

$$L_{t-1}(x', x) = \frac{\pi_{t-1}(x)K_{t-1}(x,x')}{\pi_{t-1}(x')}.$$

Thus, using the invariance of the MCMC kernels, for $t = 4, \dots, 2n$ we have (unnormalized) incremental weight:

$$\omega_t(\boldsymbol{\theta}_{1:t}) = \frac{l(y_{1:m}; \boldsymbol{\phi}_{t-1}, \mathbf{w}_{t-1})^{\zeta_t}}{l(y_{1:m}; \boldsymbol{\phi}_{t-1}, \mathbf{w}_{t-1})^{\zeta_{t-1}}}$$

for odd $t \geq 1$ the incremental weight is 1 and for $t = 2$ we have

$$\omega_2(\boldsymbol{\theta}_{1:2}) = l(y_{1:m}; \boldsymbol{\phi}_{t-1}, \mathbf{w}_{t-1})^{\zeta_2}.$$

Data Point Tempering. To apply SMC under the 'adding data points' tempering (Chopin (2002) notes that this may have a beneficial tempering effect) we perform the same algorithm (densities changing by adding the data in some order) except we have unnormalized incremental weight for $t = 2, 4, \dots, 2n$:

$$\omega_l(\boldsymbol{\theta}_{1:l}) = \frac{l(y_{1:t/2}; \boldsymbol{\phi}_{t-1}, \mathbf{w}_{t-1})}{l(y_{1:t/2-1}; \boldsymbol{\phi}_{t-1}, \mathbf{w}_{t-1})}$$

and 1 for odd t ($l(y_{1:0}; \boldsymbol{\phi}, \mathbf{w}) \triangleq 1$). That is, at the even times we add a single data point which means that there is an extra term in the likelihood.

Population Size. The choice of population size is dependent upon the Markov kernels applied. If the kernels mix well it is often a good idea to run a large number of samples for a short period, conversely, if the kernels mix badly then it is best to have a small sample size and to run the algorithm for a long time. For this example, the mixing of the kernels is reasonable for the full posterior. However, the usage of tempering allows the kernels to be more effective. Thus, we might expect a good representation of the target under the population size chosen.

For illustration we chose population sizes of 1000 and 1500 particles for the tempered and data point tempered approaches respectively. We observed extremely poor performance for the adding data method with population size 1500 or smaller, so we included a larger set of particles for this approach (also to make a reasonable comparison in terms of CPU time between the tempering methods).

We applied the stratified resampling approach (see Chopin (2004) for example) after the Gibbs step, with threshold half the sample size.

We had $n = 200$ different densities for the tempering method. This is more than is needed for satisfactory performance. For the adding data approach, the order in which the data points are added may influence the performance of the algorithm (as noted by Chopin (2002)). For our implementation, the first four data points represented each of the modes in the data.

3.4.2 Performance of Tempering Schemes on the Simulated Data

The CPU times were 78 and 50 seconds, and in Figures 3, 4 and 5 we can see the performance of the two SMC samplers.

In Figure 3 we can observe the sampled means from the SMC samplers. For the tempered densities (Figure 3 (a)) we can see that the SMC sampler is able to visit all 4! symmetric modes in the target density and thus performs well. For the adding data point approach (Figure 3 (b)) the sampler has only represented a single mode. We note, however, that for future batches of the algorithm, it is likely that another symmetric mode would be represented, thus using this approach would allow us to visit the entire state space.

In Figure 4 the importance weights are plotted against sampled means. We see that the variability of the weights for the tempered densities (Figure 4 (a)) is slightly higher and more skewed than for the adding data points (Figure 4 (b)). This is because the tempered densities approach has not resampled after time 280 (Figure 5 (a)).

Figure 5 displays the ESS. For the tempered densities (Figure 5 (a)) we see the typical weight degeneracy problem, and the resampling allows us to remove the problem successfully. The adding data points plot (Figure 5 (b)) shows a different trend. Initially the ESS is very low, which corresponds to the fact that the target is changing very quickly. As the amount of data increases the average value of the ESS appears to increase. However, the resampling for adding data is less effective than for the tempered densities.

3.4.3 Link between Temperatures and Resampling

We now discuss the importance of the temperature parameters $\{\zeta_n\}$. In Figure 5 (a) we observe that the resampling rate is reasonably constant until the last 60 densities, where we do not resample at all (the target densities change less as we reach the target of interest, despite the uniform cooling schedule). The main factor affecting this is the choice of temperature parameter (note that the resampling threshold is clearly another element which affects resampling). If the rate of resampling is reasonably consistent it implies that the evolution of densities provides a smooth path for the particles to adapt to (clearly this comment applies to a non-deterministic resampling schedule). Conversely, if resampling occurs irregularly, it often means that our choice of temperature parameters leads to large discrepancy between consecutive densities in certain regions of 'time'. Therefore, we unnecessarily remove particle diversity, which is of particular importance in multimodal situations.

To illustrate, we ran the tempered densities for $n = 50$, with a uniform cooling schedule and one which is piecewise linear; see Figure 6. In Figure 6 (a) we see the same pattern for the resampling times as in Figure 5 (a), i.e. regular resampling except as we approach the final density. We attempt to alleviate this by increasing the rate of cooling in Figure 5 (b), after $t = 50$. This helps to provide a slightly more regular resampling schedule.

We note, in extreme situations, where resampling occurs too often (e.g. every iteration) we can reduce the cooling rate and increase the number of densities. This is at the cost of further extending the state space. Our experience is that the choice of temperature parameters is often critical to efficient performance of algorithm, in terms of resampling.

3.4.4 Conclusions

In this example we have demonstrated two tempering procedures for an SMC sampler, when simulating from a mixture posterior. We found that the tempered densities approach seemed to sample the space more effectively, at the cost of increased CPU time.

Improved performance for the adding data point method can be achieved by adding (for univariate mixture data) data in the (mixture) proportion of which they occur in the data. That is, for this example, we may add a data point from the mode at -3,0,3 then 6 and continue in this order. A drawback of this approach is that it requires that we have some knowledge of the model generating the data (i.e. exactly what we wish to infer). Additionally, it is likely that the target density is similar to a density that has been tempered. As a result, the extra design that is needed for this approach is only worth doing for huge data sets (i.e. we trade off the time spent to find a good order of the data with the CPU time that will be saved).

In general, we would recommend that the first tempering procedure be applied, when it is not too expensive to calculate the likelihood. This is because of the dynamic nature of the adding data point method can mean poor initial performance (without careful thought) and the tempering effect is not as effective as for the former method. However, for problems in which data exhibit a natural order (e.g. hidden Markov models) data point tempering may be far more effective.

4. Comparison of the Population Schemes

In this Section we consider a comparison of population-based MCMC with SMC samplers. Since the methods differ in many aspects, we provide a large amount of qualitative discussion corresponding to our experience in applying both methodologies.

4·1 *Computational Cost*

Our first observation concerns the CPU times associated with the usage of both methods. We have often found that, when a large number of distributions are required so that the SMC method operates correctly, population-based MCMC is substantially cheaper. That is, since population-based MCMC uses a smaller population and more updates per chain, the storage requirements are substantially reduced. This must be counter-balanced by the fact that for SMC there is no burn-in and often (given reasonable performance of the SMC sampler) the samples appear to be close to i.i.d, and for MCMC, both of these require extra iterations (i.e. a burn-in period and thinning of sample realizations).

4·2 *Specification of Simulation Parameters*

We now consider how difficult (or easy) it is to specify simulation parameters for both methods, in order that sampler performance is adequate.

In terms of temperature specification for SMC samplers, as shown in Section 3·4, this can often be vital to the performance. That is, the rate of resampling is heavily dependent upon the temperature parameters and tuning these for efficient performance is often time consuming. Conversely, for population-based MCMC, it is not too difficult to specify temperature parameters so that reasonable acceptance rates are found. However, this does not mean that the algorithm will necessarily perform well: it is more difficult to judge what cooling schedule yields satisfactory performance for MCMC. We note, that for both MCMC and SMC we should not rely solely upon the ideal exchange acceptance rate/resampling rate to indicate upon good performance of the algorithm. We recommend checking sampled parameters and having multiple runs to help ensure that the sampler approximates the target density correctly.

There is more freedom to choose a tempering procedure for SMC than for MCMC. For example, data point tempering can perform very badly for MCMC methods.

4·3 *Markov Kernels*

We now compare the performance of MCMC and SMC methods under similar Markov kernels.

In terms of Markov kernels there is substantially more freedom in specifying these for the SMC method. Kernels need not be reversible or even Markov (and hence time adaptive) for the SMC method. Despite this fact, it is not always simple to construct an efficient backwards Markov kernel for such moves. For example, if we wish to calculate the sub-optimal choice (3.3), we need to evaluate an integral which may not be available in closed form.

In our experience, when the Markov kernels used in MCMC and SMC allow easy movement around the state-space, then there is little to choose between the methods (since performance and CPU times are similar, unless we run a very large number particles for SMC). However, for reasonably mixing Markov kernels (i.e. that are slow to move around the space, but in a reasonable amount of CPU time correctly sample the target) we have found that population-based MCMC performs better. We now demonstrate this in the following example.

4·4 *Example: Hidalgo Stamp Data Revisited*

We return the Hidalgo stamp data example in Sections 1·1 and 2·7. We ran the SMC sampler in Section 3·4 20 times with a population size of 500 (with the reversible jump kernel included). We took $n = 750$ and used the stratified resampling method with threshold 250 samples (applied after the Gibbs update).

The MH kernels had a decreasing value of the proposal variance and these were set so that the average acceptance rate was in the range $(0.15, 0.6)$ (as noted by Chopin (2002) this will not necessarily mean that the kernels mix well, but it is not clear how to construct a good global MCMC move (or Markov kernel) which is why we have resorted to population-based simulation in the first place).

The temperature sequence was taken to be piecewise linear with ζ increasing uniformly by 1/15000 (i.e. $\zeta_1 = 1/15000$, $\zeta_2 = 2/15000$ etc.) for the first 150 distributions, 1/2500 for the next 125, 1/750 for the next 225 and finally 1/250 for the last 250 distributions. This choice was made to help ensure that the resampling occurred quite consistently.

In Figure 7 (a) we can observe the sampled k. The performance of the sampler appears to be poor in comparison to Figure 2. We have seen that the SMC algorithm, as we have implemented it, has performed badly. We were unable to find a set of temperatures or MCMC kernels so that the sampler worked correctly. Jasra (2005) shows that an interacting SMC method can get around the problems here - with a minor modification of the existing code.

4·5 *Summary*

Overall, our thoughts on the relative advantages/disadvantages of population-based MCMC against sequential Monte Carlo samplers are as follows:

(A) *MCMC can often be less computationally expensive.*

(B) *MCMC is often easier to calibrate (in terms of simulation parameters).*

(C) *SMC requires no burn-in.*

(D) *SMC is a richer method, allowing us the freedom to* design *good samplers.*

One area where we feel SMC is superior to MCMC is in problems with a natural ordering of the data; for example in time series data. For example in hidden Markov models (e.g. Robert, Rydén & Titterington (2000)) SMC samplers will be able to take advantage of the Markov nature of the hidden sequence and use data point tempering which is very computationally efficient.

We feel that the freedom of SMC is its main advantage (for example, the interacting SMC approach has a more satisfactory partitioning method (than for population-based MCMC)). However, population-based MCMC provides a principled way to draw from a target density, that is often easier to implement than SMC. In more simple terms, the comparison between population-based MCMC and SMC is analogous to that of the Gibbs sampler and Metropolis-Hastings: population-based MCMC methods are less flexible but easier to use.

5. Discussion

In this paper we have provided a review of population-based simulation. We discussed population-based MCMC and SMC samplers and compared the approaches. We have given examples of how to use the methodology and demonstrated that it is often superior to standard (single chain) MCMC.

As outlined at the beginning of the paper, we intended to convince applied statisticians the population-based simulation can be easily implemented at reasonable computational cost. We hope that the examples included have done this. For more complex examples (e.g. the gene expression and population genetics examples alluded to at the beginning of the paper) we have shown in previous work (Jasra 2005) that population-based simulation provides significant improvement when standard methods completely fail to work; at reasonable increase in computational cost (i.e. as noted in Robert & Casella (2004) there is no free lunch: an increase in storage means longer CPU times).

Areas of future research in population-based MCMC are as follows. The theoretical analysis of such methods is seldom considered in the literature (See Madras & Zheng (2003) for an exception) and it is important to see if there is any theoretical advantages of population-based MCMC vs single chain MCMC. One important aspect is whether population algorithms converge any faster to π^* than the original algorithm to π. For population algorithms we can observe better mixing across the space for the target of interest and, given a sensible definition of iteration, will the population kernel necessarily or *ever* converge quicker? Madras & Zheng (2003) show that this can be the case for the mean-field Ising model, but it is vital to investigate if any general result may be obtained. Additionally, it is important to produce fast mixing transition kernels. One way this may be achieved is to use adaptive methods. That is, we may use the population to update kernels and improve the exploration of the state space. Such a procedure is clearly non-Markovian thus the ergodicity of such a stochastic process needs to be verified; see Andrieu & Moulines (2003) and Roberts & Rosenthal (2005) for a starting point.

Possible research areas in SMC is as follows. An aspect of interest is when and how to resample. For example, as noted by Chen, Xie & Liu (2005) the time parameter is not always an appropriate way to decide when to resample (since certain samples may have low importance weights, but as time n approaches these samples have higher weights). Additionally, Chopin (2002) notes that the weights themselves are not always a suitable criterion to judge the performance of the algorithm as they do not always take into account the effectiveness of the mutation step. It is interesting to see if there are alternative criteria (to time, ESS, form of importance weights) to decide when to resample and how well the sampler has performed (online). Another theoretical point of interest is the investigation of bounds on the total variation distance between the marginal (terminal time) particle measure and the target measure. At present such bounds (e.g. Theorem 8.3.3 of Del Moral (2004)) assume that the state space is bounded; it is important to construct bounds that do not assume this.

Acknowledgement

The first author was supported by an Engineering and Physical Sciences Research Council Studentship. The first author acknowledges many useful conversations and correspondence with Arnaud Doucet.

REFERENCES

Andrieu, C. & Moulines E. (2003). On the ergodicity properties of some adaptive MCMC algorithms, Technical Report, University of Bristol.

Andrieu, C. & Robert C. P. (2001). Controlled MCMC for optimal sampling, Technical Report, Universitié Paris Dauphine.

Atachade, Y. F. & Liu, J. S. (2004). The Wang-Landau algorithm for Monte Carlo computation in general state spaces, Technical Report, University of Ottawa.

Cappé, O., Gullin, A., Marin, J. M. & Robert, C. P. (2004). Population Monte Carlo. *Journal of Computational and Graphical Statistics*, **13**, 907-25.

Chen, Y., Xie, J., & Liu, J. S. (2005). Stopping-time resampling for sequential Monte Carlo methods. *Journal of the Royal Statistical Society Series B*, **67**, 199–217.

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika*, **89**, 539-552.

Chopin, N. (2004). Central theorem for sequential Monte Carlo methods and its application to Bayesian inference. *Annals of Statistics*, **32**, 2385-2411.

Del Moral, P. (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications.* Springer: New York.

Del Moral, P. & Doucet, A. (2003). On a class of genealogical and interacting Metropolis models. In *Séminaire de Probabilités XXXVII*, Ed. Azéma, J., Emery, M., Ledoux, M. and Yor, M., *Lecture Notes in Mathematics*, Springer: Berlin, **1832**, 415-446.

Del Moral, P., Doucet, A. & Peters, G. W. (2005). Sequential Monte Carlo samplers, Technical Report, University of Cambridge.

Doucet, A., Godsill, S. & Andrieu, C. (2000). On sequential Monte Carlo sampling for Bayesian filtering. *Statistics and Computing* **10**, 197–208.

Doucet, A., De Freitas, J. F. G. & Gordon, N. J. (2001). *Sequential Monte Carlo Methods in Practice.* Springer: New York.

Gelman, A. & Meng, X. L. (1998). Simulating normalizing constants: From importance sampling to bridge sampling to path sampling, *Statistical Science* **13**, 163–85.

Geyer, C. (1991), Markov chain maximum likelihood, In *Computing Science and Statistics: The 23rd symposium on the interface*, (E. Keramigas ed), 156-63 Fairfax: Interface Foundation.

Gilks, W. R., Roberts, G. O. & George, E. I. (1994). Adaptive direction sampling. *The Statistician* **43**, 179–89.

Gilks, W. R. & Berzuini, C. (2001). Following a moving target - Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society Series B* **63**, 127–46.

Goswami, G. R. & Liu J. S. (2005). On real parameter evolutionary Monte Carlo algorithm, Technical Report, Harvard University.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* **82**, 711–32.

Green, P. J. & Mira, A. (2001). Delayed rejection in reversible jump Metropolis-Hastings. *Biometrika* **88**, 1035–53.

Hammersley, J. M. & Morton, K. W. (1999). Poor Man's Monte Carlo. *Journal of the Royal Statistical Society Series B* **16**, 23–38.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**, 97–109.

Heard, N. A., Holmes, C. C., & Stephens, D. A. (2005). A quantitative study of gene regulation involved in the immune response of anopheline mosquitoes: An application of Bayesian hierarchical clustering of curves, *Journal of the American Statistical Association*, To appear.

Hukushima, K. & Nemoto, K. (1996). Exchange Monte Carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan* **65**, 1604-08.

Iba, Y. (2000). Population Monte Carlo algorithms. *Transactions of the Japanese Society of Artificial Intelligence* **16**, 279–86.

Jasra, A. (2005). *Bayesian Inference for Mixture Models via Monte Carlo Computation*, PhD thesis (in preparation), Imperial College London.

Jasra, A., Holmes, C. C., & Stephens, D. A. (2005a). Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modelling, *Statistical Science* **20**, 50–67.

Jasra, A., Stephens, D. A., & Holmes, C. C. (2005b). Population-based reversible jump Markov chain Monte Carlo, Technical Report, Imperial College London.

Kou, S. C, Zhou, Q., & Wong, W. H. (2005). Equi-energy sampler with applications to statistical inference and statistical mechanics, Technical Report, Harvard University.

Liang, F. & Wong, W. H. (2001). Real parameter evolutionary Monte Carlo with applications to Bayesian mixture models. *Journal of the American Statistical Association* **96**, 653–66.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing.* Springer: New York.

Madras, N. & Zheng, Z. (2003). On the swapping algorithm. *Random Structures and Algorithms* **22**, 66–97.

McLachlan, G. J. & Peel, D. (2000). *Finite Mixture Models.* Wiley: Chichester.

Mengersen, K. L. & Robert, C. P. (2003). IID sampling using self-avoiding population Monte Carlo. In *Bayesian Statistics 7*, Ed. Bernardo, J. M., Bayarri, M. J., Berger, J. O., Dawid, A. P., Heckerman, D., Smith, A. F. M. & West, M., OUP: Oxford, 277-93.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics* **21**, 1087–92.

Neal, R. M. (1996). Sampling from multimodal distribtutions using tempered transitions. *Statistics and Computing* **4**, 353-66.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing* **11**, 125–39.

Pritchard, J. K., Stephens, M. & Donnelly, P. (2001). Inference of population structure using multilocus genotype data. *Genetics* **155**, 945–59.

Richardson, S. & Green, P. J. (1997). On Bayesian analysis of mixture models with an unknown number of components (with Discussion). *Journal of the Royal Statistical Society Series B* **59**, 731–92.

Robert, C. P. & Casella G. (2004). *Monte Carlo Statistical Methods.* Second edition. Springer: New York.

Robert, C. P., Rydén, T., & Titterington, D. M. (2000). Bayesian inference in hidden Markov models through reversible jump Markov chain Monte Carlo. *Journal of the Royal Statistical Society Series B* **62**, 57–75.

Roberts, G. O. & Rosenthal, J. S. (2004). General state space Markov chains and MCMC algorithms. *Probability Surveys*, **1**, 20–71.

Roberts, G. O. & Rosenthal, J. S. (2005). Coupling and ergodicity of adaptive MCMC, Technical Report, University of Lancaster.

Warnes, A. (2001). *The Normal Kernel Coupler: An Adaptive Markov Chain Monte Carlo Method for Efficiently Sampling from Multimodal Distributions*, PhD thesis, University of Washington.

Captions

Figure 1: Hidalgo stamp data (a) and sampled number of components (b). For (b), we fitted the mixture model of Richardson & Green (1997) (with an unknown unmber of components). The output is the number of components from a reversible jump MCMC algorithm plotted every 100^{th} sweep. For (b) the CPU time was approximately 4 minutes.

Figure 2: Sampled k from a population-based reversible jump MCMC algorithm; Hidalgo stamp data. The algorithm was run for 1.2 million iterations with a 1 million sweep burn-in and we recorded every 20th sample post burn-in. The CPU time was approximately 12 minutes.

Figure 3: Trace plots of the sampled means from the SMC samplers. We fitted a four component normal mixture to the data, the output is a single population of 1500 samples from an SMC sampler using 200 tempered densities (a) and a single population of 2000 samples from an SMC sampler which had densities change by adding data points (b).

Figure 4: Plot of importance weights against sampled means from the SMC samplers. We fitted a four component normal mixture to the data, the output is a single population of 1500 samples from an SMC sampler using 200 tempered densities (a) and a single population of 2000 samples from an SMC sampler which had densities change by adding data points (b).

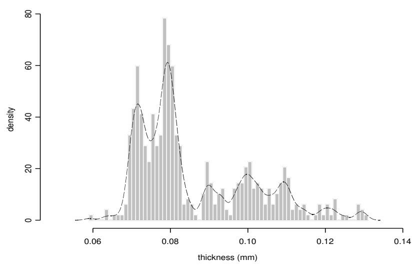
Figure 5: Effective sample size plots from the SMC samplers. We fitted a four component normal mixture to the data, the output is for a single population of 1500 samples from an SMC

sampler using 200 tempered densities (a) and a single population of 2000 samples from an SMC sampler which had densities change by adding data points (b).

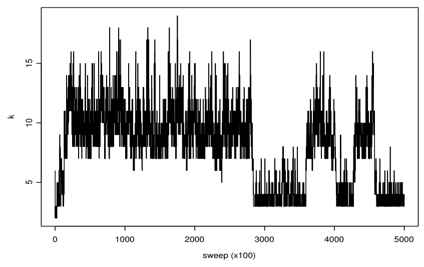
Figure 6: Effective sample size plots and temperature from the SMC samplers. We fitted a four component normal mixture to the data, the output is for a single population of 1500 samples from an SMC sampler using 50 tempered densities with uniform cooling rate (a) and (b) similarly, but with a piecewise linear cooling schedule in the figure.

Figure 7: Sampled k from the SMC sampler; Hidalgo stamp data.

Figures



(a) Data.



(b) Sampled k.

Figure 1:

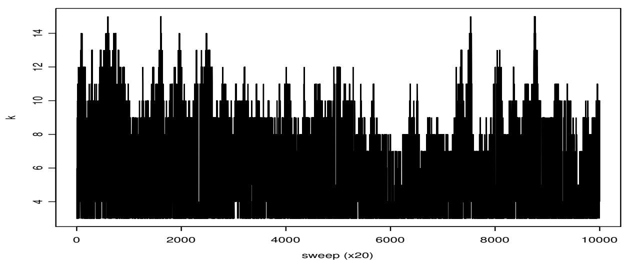
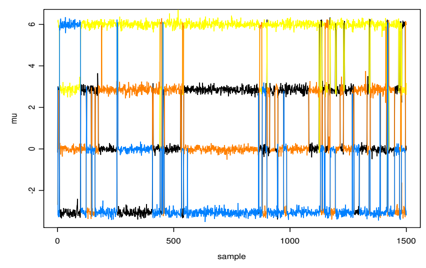
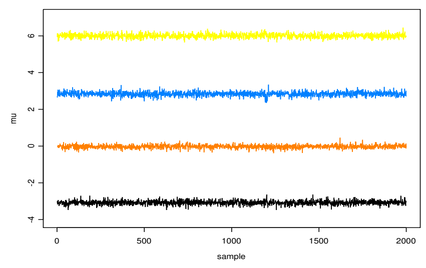


Figure 2:

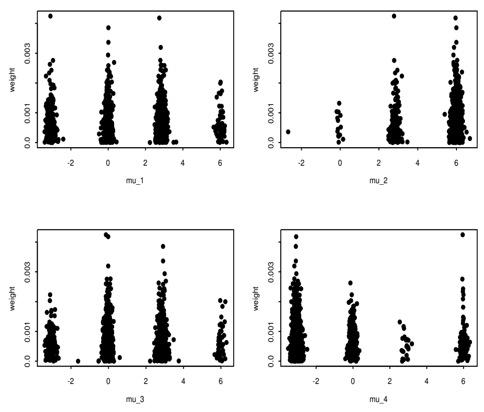


(a) Tempered Densities.

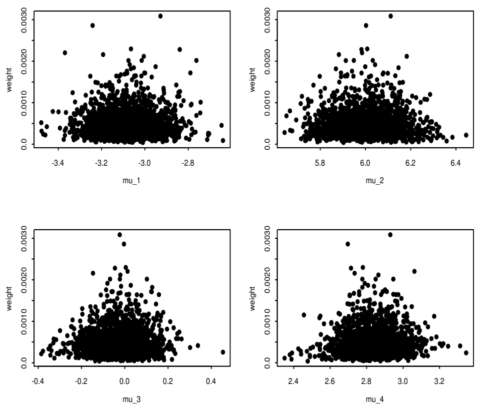


(b) Adding Data Points.

Figure 3:

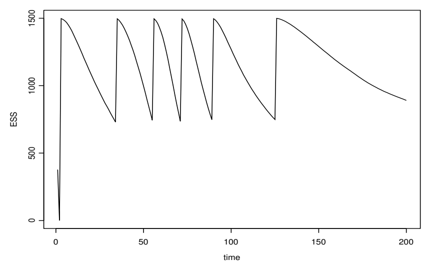


(a) Tempered Densities.

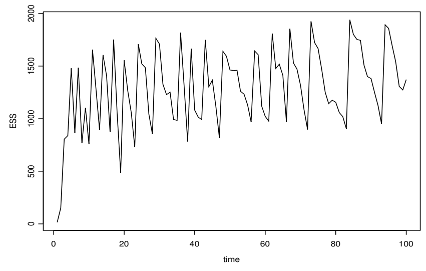


(b) Adding Data Points.

Figure 4:

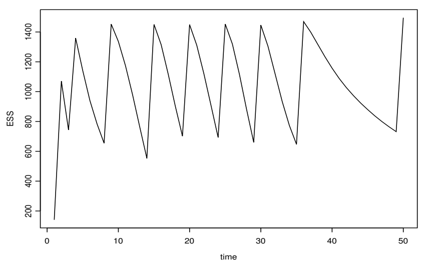


(a) Tempered Densities.

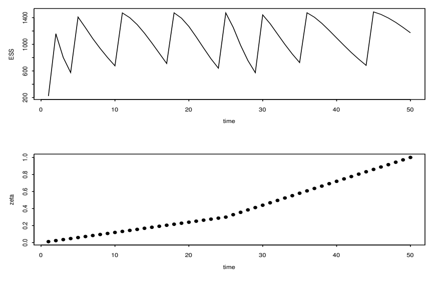


(b) Adding Data Points.

Figure 5:



(a) Uniform Cooling Schedule.



(b) Piecewise Linear Cooling Schedule.

Figure 6:

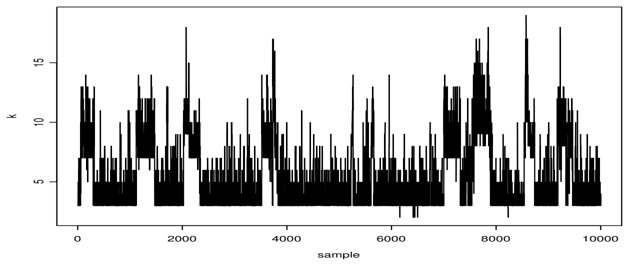


Figure 7: