Stat Comput (2007) 17: 263–279
DOI 10.1007/s11222-007-9028-9

On population-based simulation for static inference

Ajay Jasra · David A. Stephens · Christopher C. Holmes

Received: 12 October 2005 / Accepted: 20 March 2007 / Published online: 27 July 2007

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 $\{X_n\}_{n=1,\ldots,N}$ in parallel in order to simulate from some target density π (or potentially sequence of target densities). Population-based simulation is important as many challenging sampling problems in applied statistics cannot be dealt with successfully by conventional Markov chain Monte Carlo (MCMC) methods. We summarize population-based MCMC (Geyer, Computing Science and Statistics: The 23rd Symposium on the Interface, pp. 156–163, 1991; Liang and Wong, J. Am. Stat. Assoc. 96, 653–666, 2001) and sequential Monte Carlo samplers (SMC) (Del Moral, Doucet and Jasra, J. Roy. Stat. Soc. Ser. B 68, 411–436, 2006a), providing a comparison of the approaches. We give numerical examples from Bayesian mixture modelling (Richardson and Green, J. Roy. Stat. Soc. Ser. B 59, 731–792, 1997).

Keywords Markov chain Monte Carlo · Sequential Monte Carlo · Bayesian mixture models · Adaptive methods

A. Jasra (✉)
Department of Mathematics, Imperial College London, SW7 2AZ, London, UK
e-mail: ajay.jasra@ic.ac.uk

D.A. Stephens
Department of Mathematics and Statistics, McGill University, H3A 2K6, Montreal, Canada

C.C. Holmes
Department of Statistics, University of Oxford, OX1 3TG, Oxford, UK

1 Introduction

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

$$\mathbb{E}_\pi[h(X)] = \int_E h(x)\pi(x)dx, \qquad \pi(x) = \frac{\gamma(x)}{Z}, \tag{1.1}$$

where π is a probability density with respect to some σ-finite measure dx on measurable space $(E, \mathcal{E})$ and $\gamma : E \to \mathbb{R}^+$ may be evaluated pointwise, with $0 < Z < \infty$ unknown. 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 et al. 1953; Hastings 1970)) which generates samples from an ergodic Markov kernel $K : E \times \mathcal{E} \to [0, 1]$ with invariant measure π and estimates (1.1) via:

$$S_T(h) = \frac{1}{T}\sum_{i=1}^{T} h(x_i), \tag{1.2}$$

where $x_{1:T} \triangleq (x_1, \ldots, x_T)$ (resp. $X_{1:T} \triangleq (X_1, \ldots, X_T)$) and $x_{1:T}$ have been drawn from K. Note that $dx_{1:T} \triangleq dx_1 \times \cdots \times dx_T$ and throughout this paper all probability measures are assumed to be absolutely continuous with respect to some σ-finite measure dx.

However, in many modern areas of applied statistics, for example gene clustering (Heard et al. 2006) and admixture modelling in population genetics (Pritchard et al. 2001),

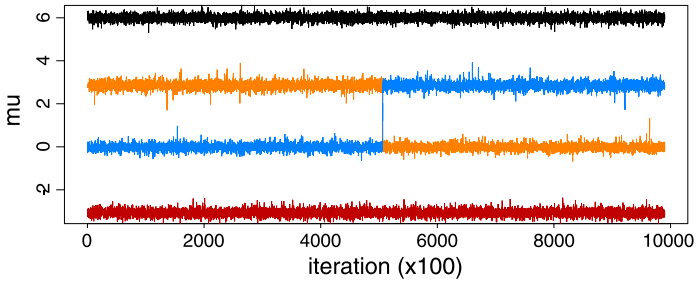


Fig. 1 Sampled means from the MCMC algorithm for mixtures. The algorithm was run for 1 million iterations with a 10000 iteration burn-in and every 100th sample post burn-in is displayed. The CPU time was 336 seconds

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 and/or high dimensional target densities, and leading to poorly mixing MCMC algorithms. We illustrate this problem with the following example from Bayesian mixture modelling.

1.1 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 and Peel (2000) for an overview. It should be noted that, in our experience, population-based methods are often needed in the Bayesian analysis of finite mixture models and thus provide a non-trivial (if well understood) example.

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

$$p(y_l|\phi_{1:k}, w_{1:k-1}, k) = \sum_{j=1}^{k} w_j f(y_l; \phi_j),$$

where $\phi_{1:k}$ are component specific parameters, the weights $w_{1:k-1} = (w_1, \ldots, w_{k-1})$ are such that $\sum_{j=1}^{k-1} w_j \leq 1$, $w_j \geq 0 \, \forall j = 1, \ldots, k-1$, $w_k = 1 - \sum_{j=1}^{k-1} w_j$, $p(\cdot)$ denotes an arbitrary probability 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 $j = 1, \ldots, k$) are the following (similar to Richardson and Green 1997): $\mu_j \sim \mathcal{N}(\xi, \kappa^{-1})$, $\lambda_j \sim \mathcal{G}a(\alpha, \beta)$ and $w_{1:k-1} \sim \mathcal{D}(\delta)$. 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. The prior parameter setting is as Richardson and 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 et al. (2005a) for a review.

To illustrate the aspects discussed above we consider the simulated data in Jasra et al. (2005a): 100 simulated data points from an equally weighted mixture of four (i.e. $k = 4$) normal densities with means at $(-3, 0, 3, 6)$ and standard deviations 0.55.

We ran a cycle of random walk MH steps updating, in one block, $\mu_{1:k}$, then $\lambda_{1:k}$ and finally $w_{1:k-1}$. The proposal variances were adjusted to yield an average acceptance rate in the range $(0.3, 0.6)$. The algorithm was run for 1 million iterations, with a 10000 iteration burn in; the sampled means are plotted in Fig. 1 (every 100th post burn-in). In Fig. 1 we can observe that the sampler is unable to explore all of the 4! symmetric modes. We remark here that we are unconcerned with whether exploring the symmetric modes is important; we simply seek to explore the target distribution *correctly*.

This situation is typical of many standard MCMC kernels when attempting to simulate from probability measures with multimodal or very high dimensional densities. We note that the difficulties presented here may be alleviated via adaptive methods (Andrieu and Robert 2001). That is, to update the transition kernel on the basis of realized values of the chain. Such adaptive procedures include: adapting proposal variances in random walk Metropolis-Hastings steps via the Robbins-Monro algorithm and approximating the target and using this approximation in a Metropolis-Hastings independence sampler; see Andrieu and Moulines (2006). However, 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 and Mira 2001); see Jasra et al. 2005a for examples), it seems clear that alternative methods are needed. Note that one single chain method, simulated tempering (Marinari and Parisi 1992; Geyer and Thompson 1995), can perform better than some population MCMC schemes discussed below (see Zheng 2003), *if the pseudo prior can be estimated accurately* (this is typically a high dimensional integral which can be estimated on the fly using stochastic approximation; see Atchadé and Liu 2004). However, in some of the applications for which population methods are required (e.g. Bioinformatics), the two fold problem of estimating the pseudo prior and exploring the state-space will be more difficult

than using a population of samples; we would not expect to be able to solve both problems simultaneously (that is, we seek to estimate marginal quantities and explore the space).

1.2 Population-based simulation and recent literature

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 of population algorithms; one which relies solely upon MCMC methodology and another which uses importance sampling/resampling ideas (Doucet et al. 2001; Liu 2001). It should be stressed that whilst there are substantial similarities of both approaches (the simulation of samples in parallel, sampling from families of distributions), there are many key differences; for example the theoretical convergence (the iteration (MCMC) against number of parallel samples (importance sampling)). The differences in the approaches should become apparent as the reader proceeds though the paper.

The first method, which we refer to as population-based MCMC (note that this is different from *population Monte Carlo* which correspond to importance sampling algorithms such as in Cappé et al. 2004), works by building a new target $\pi^*(x_{1:N})dx_{1:N}$ on the product space $(E^N, \bigvee_{i=1}^N \mathcal{E})$ such that π^* admits π as a marginal. To our knowledge, this method was originally developed by (see also Iba 2001 for an introduction) 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 via an exchange step (see Sect. 2.2). This approach was independently developed by Hukushima and Nemoto (1996) who described this as 'Exchange Monte Carlo'. Another population method, adaptive direction sampling, was devised by Gilks et al. (1994). Further advances came in Liang and 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. (2006a). Sequential Monte Carlo methods have a rich history, originating from the initial work of Hammersely and Morton (1954); see Doucet et al. (2000) and Liu (2001) 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, efficiently, using MCMC. However, as noted by Chopin (2002) and Del Moral and Doucet (2003), SMC methods may also be used to simulate from a single, static target. This idea also appeared in Jarzynski (1997) and independently in Neal (2001). 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 and Berzuini 2001), the sequential particle filter of Chopin (2002), the dynamically weighted importance sampling (DWIS) method (Liang 2002) and population Monte Carlo (Cappé et al. 2004) but since the SMC sampler approach contains all of these methods as a special case (with slight exception of DWIS) 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 Sect. 2 we review population-based MCMC, in terms of the move types and the sequence of densities. We also return to the mixture example to provide some guidelines on how to construct population MCMC algorithms. In Sect. 3 we consider SMC samplers, detailing some important implementation issues. In Sect. 4 we provide a comparison of these methodologies. Finally, in Sect. 5 we conclude the paper, giving some discussion on potential new research areas.

2 Population-based MCMC

2.1 The method

Population-based MCMC may be described as follows. In order to sample from a target density π, we define a new target measure:

$$\pi^*(x_{1:N})dx_{1:N} = \left[\prod_{n=1}^{N} \pi_n(x_n)\right] dx_{1:N}, \tag{2.3}$$

where we assume that $\pi \equiv \pi_n$ for at least one $n \in \mathbb{T}_N$.

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 for example Roberts and Rosenthal (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 (although importance sampling techniques may be adopted).

Two main elements provide the motivation of population MCMC algorithms:

- The sequence of densities $\{\pi_N\}$ will be selected so that they are all related and, in general, easier to simulate than π. This can provide valuable information for simulating from π.
- The usage of a population of samples will allow more global moves (than single chain MCMC) to be constructed resulting in faster mixing MCMC algorithms.

We now discuss some population moves and the sequence of distributions $\{\pi_n\}_{n\in\mathbb{T}_N}$.

2.2 Population moves

The different types of population moves that have been used are now described. We use the GA terminology of Liang and Wong (2001) and Del Moral and Doucet (2003) (for example). A similar coverage may be found in Liu (2001).

Mutation This move seeks to update a single member of the population via a Markov kernel. That is, $X'_n|x_n \sim K(x_n,\cdot)$ (where $'$ denotes a new value of the population member). For example we may update $x_{1:N}$ via:

$$K(x_{1:N}, dx'_{1:N}) = \prod_{n=1}^{N} K_n(x_n, dx'_n),$$

where K_n is a Markov kernel that is π_n-stationary. The purpose of this move is to allow for 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 n and q; this is accepted with probability $\min\{1, A\}$:

$$A = \frac{\pi_n(x_q)\pi_q(x_n)}{\pi_n(x_n)\pi_q(x_q)}, \tag{2.4}$$

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. 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 and 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 and Wong (2001). Suppose $x_n = (x_{1n}, \ldots, x_{dn})$ $\forall n$, then the crossover selects a position in the vector to crossover information. That is, if we propose to crossover the lth position in the vector for chains n and q we have:

$$x'_n = (x_{1n}, \ldots, x_{(l-1)n}, x_{lq}, \ldots, x_{dq}),$$
$$x'_q = (x_{1q}, \ldots, x_{(l-1)q}, x_{ln}, \ldots, x_{dn}).$$

This move is accepted with probability $\min\{1, A\}$, with A as for (2.4) 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. One potential extension, mentioned by a referee, is to allow for a circular crossover, that is to allow break points at different parts of x_n; we investigate this in Sect. 2.7.

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 one of the ideas of population-based simulation is to use the extra information of the population to improve the exploration ability of the algorithm. If all of the chains are stuck in a single mode say, then the advantage is lost.

We note that further move types may be found in Goswami and Liu (2007). 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 $\pi_n \equiv \pi$ $\forall n \in \mathbb{T}_N$. This approach is used by Warnes (2001). 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_n using a proposal that is a normal kernel estimate of π, fitted to the other samples (at the current iteration). Robert and Casella (2004) note that this may not work well for high dimensional targets due to the poor performance of density estimation in large spaces.

Tempered Suppose that

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

for $\zeta \in \Xi$ with Ξ a set of values on $(0, 1]$. Then we may take: $\pi_n(x) \propto [\pi(x)]^{\zeta_n}$ $\zeta_n \in \Xi$ and that $\zeta_n = 1$ for at least one $n \in \mathbb{T}_N$. This approach is used by Liang and Wong (2001) and Jasra et al. (2005b). The idea is that the distributions at high *temperatures*, that is, ζ close to zero, are easily sampled and can improve the mixing of the entire algorithm. 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 for 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 and Liu (2007). A related approach, suggested by Gelman and Meng (1998), is to take

$$\pi_n(x) \propto [\pi(x)]^{\zeta_n} [p(x)]^{1-\zeta_n},$$

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

As noted by Geyer and Thompson (1995), the tempering procedure may not always improve mixing, for example in the witches hat density (Matthews 1993). However, one potential way to use the tempering idea is via *multiple tempering schemes* across different regions of the target space. That is, partition the state space and allow for separate tempering strategies within each region. For example, in a complex region of the space we heat the targets (and hence they are more easily sampled) and in well defined regions the targets are cooled. We aim to investigate such a method in future work.

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

$$\pi_n(\theta|y_{1:n}) \propto p(y_{1:n}|\theta)\pi(\theta),$$

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 Sect. 3.5). 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 Sect. 3.5.

Different dimensions In this case, we define a sequence of densities $\{\pi_n\}_{n\in\mathbb{T}_N}$ on state-space $E = E_1 \times \cdots \times E_N$ with $\dim(E_1) < \cdots < \dim E_N$. This has been presented in Liang (2003) (although it had appeared previously in the computational physics literature; e.g. Ron et al. 2002), where the idea is that the information in lower dimensional spaces (which can easily be learnt), can be crossed over with that in high dimensional spaces to improve exploration of state-space (Liang 2003 terms this move extrapolation and projection). As an example, Liang (2003), when simulating from the Ising model, uses Ising models with lower linear sizes (fewer terms in the energy function than the target of interest) as the sequence of densities. In our experience, with complicated statistical models, it can be very difficult to crossover information between different dimensional problems, but Liang (2003) successfully demonstrates the method on the witches hat problem and the Ising model.

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

$$\pi^*(x_{1:N}) \propto \pi(x_N) \prod_{n=1}^{N-1} \pi(x_n)\mathbb{I}_{A_n}(x),$$

where $\mathbb{I}_A(\cdot)$ is the indicator function. The objective is that in each A_n 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 Atchadé and 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 for 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 to 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 and Rosenthal 2005).

A related but differing approach is used by Kou et al. (2006). They adopt the sequence of densities:

$$\begin{aligned} \pi_n(x) &\propto \exp\{-\zeta_n g_n(x)\}, \\ g_n(x) &= g(x) \vee G_i, \\ g(x) &= -\log\{\gamma(x)\}, \end{aligned} \tag{2.5}$$

where $\pi(x) = \frac{1}{Z}\gamma(x)$, $\inf_{x\in E}\{g(x)\} \geq G_1 < \cdots < G_N < \infty$ 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. (2006) below.

A similar stratification idea is the multicanonical sampler (Mitsutake et al. 2003), see also Atchadé and Liu (2006). In this approach:

$$\pi_n(x) \propto \sum_{j=1}^{N_e} \frac{1}{Z_{n,j}} \frac{\pi(x)^{\zeta_n}}{Z_n} \mathbb{I}_{G_j}(x)$$

with $Z_n = \int \pi(x)^{\zeta_n} dx$, $Z_{n,j} = \int \frac{\pi(x)^{\zeta_n}}{Z_n} \mathbb{I}_{G_j}(x) dx$ and $\bigcup_i^{N_e} G_i$ is a partition of the energy space. This class of algorithm is complicated by the fact that the $\{Z_{n,j}\}$ are unknown and often learnt on the fly (e.g. by the Wang-Landau algorithm); see Atchadé and Liu (2006). Atchadé and Liu (2006) report that this is similar to the equi-energy sampler, in terms of efficiency.

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 in some sense 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 (this latter point will not be such a problem if fast mixing, global population moves can be constructed). 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 used the criterion that samples from the target of interest are reasonably similar for long runs of the algorithm. Kou et al. (2006) suggest that the number of chains (for equi-energy sampling) should be roughly proportional to the dimensionality of the target density. See Sect. 2.7 for some further discussion.

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. (2006) (see also the methods of Mitsutake et al. (2003), Andrieu et al. (2007a) and Brockwell et al. (2007)). This method generates a non-Markovian stochastic process with target densities (2.5). 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 $x_{N-1}^i \in [G_l, G_{l+1}]$ 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 having an estimate of the posterior mode, prior to simulation. Also a sensible partitioning of the energy space is required; see Kou et al. (2006) 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 Sect. 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. (2006); see Andrieu et al. (2007b). It should be noted that such an algorithm can be interpreted within the framework of *non-linear* MCMC (Andrieu et al. 2007a) and may be studied using the techniques considered there. Due to space constraints we do not consider this method further.

2.6 A typical algorithm

In Algorithm 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 and Rosenthal 2004) from π^*-a.e. starting points. We remark that there are no theoretical difficulties in defining a trans-dimensional (Green 1995) version of the algorithm; see Jasra et al. (2005b).

Algorithm 1 (A population-based MCMC algorithm)

0. (INTIALIZATION)

- *Initialize the chain* $x_{1:N}$, $X_n \sim \nu$, ν *a probability density.*
- *For* $t = 1, \ldots, T$ *sweep over the following:*

1. (MUTATION)
   - *Select a chain n with a fixed (time homogeneous) probability and then update x_n using a π_n-irreducible, aperiodic Markov kernel, which admits π_n as its invariant distribution.*
2. *Make a random choice between performing steps 3 or 4.*
3. (CROSSOVER)
   - *Perform the crossover move in Sect. 2.2.*
4. (EXCHANGE)
   - *Perform the exchange move in Sect. 2.2.*

end

2.7 Example

To see that population-based simulation can provide improvements over standard MCMC methods and to investigate some of the ideas discussed above, we return to the mixture example in Sect. 1.1.

The algorithm followed the framework of Algorithm 1, except that the only population move used was an exchange move; we investigate the crossover move later in the Section. We will adopt (with $l(y_{1:m};\phi_{1:k},w_{1:k-1})$ the mixture likelihood) densities of the form:

$$\pi_n(\phi_{1:k},w_{1:k-1}|y_{1:m}) \propto l(y_{1:m};\phi_{1:k},w_{1:k-1})^{\zeta_n}\,p(\phi_{1:k},w_{1:k-1})$$

that is, tempered densities. We firstly investigate, empirically, the performance of various heating schemes, then the size of the population, finally we look at the crossover move. The performance criteria we use is the point estimates of the component specific means $\mu_{1:k}$; which are the same in the posterior and approximately equal to 1.5. The MH proposal variances were set as:

$$\sigma_n = \frac{\sigma_1}{\gamma_n+1}, \quad n=2,\ldots,N,$$

with $\sigma_1=0.4$ (means), $\sigma_1=0.55$ (precisions) and $\sigma_1=0.6$ (weights) (these were the settings for the first example). We found that this lead to acceptance rates in the range $(0.3,0.5)$ (averaged over each chain).

2.7.1 *Various tempering approaches*

In this section we run the algorithm with $N=20$ using the following tempering schemes (note $\zeta_1=1$):

- Uniformly Spaced (A):

$$\zeta_n=\zeta_{n-1}-\frac{1}{N}, \quad n=2,\ldots,N.$$

- Logarithmic Decay (B):

$$\zeta_n=\frac{\log(\zeta_{n-1}+1)}{\log(K)}, \quad n=2,\ldots,N,$$

$K>0$.

Table 1 Estimates of means from mixture comparison for various tempering strategies. We ran each sampler for 1 million iterations, 4 times and heating schedule (C) was used. The estimates are presented in increasing order, for presentation purposes

Sampler details	Component			
	1	2	3	4
A	1.28	1.41	1.49	1.58
B	1.02	1.03	1.54	1.71
C1	1.34	1.39	1.51	1.52
C2	0.83	1.19	1.74	2.01

- Power Decay (C):

$$\zeta_n=(\zeta_{n-1}-K)^{\widetilde{\alpha}}, \quad n=2,\ldots,N,$$

$K\in(0,1)$ and $\widetilde{\alpha}>1$.

We ran four versions of the algorithm (A), (B) with $K=2.25$, (C1) with $\widetilde{\alpha}=3/2$, $K=0.001$ and (C2) with $\widetilde{\alpha}=6/5$, $K=0.001$. The first algorithm (A) provides a spread of densities from the very flat to the target, the second (B) with a quickly heating scheme (that is, the target is isolated from the flat densities), the third (C1) with a slowly heating sequence and (C2) likewise, except with many densities similar to the target. We remark that it is possible to derive automatic temperature selection; see Iba (2001) for an approach that uses pilot simulations.

The algorithm was run four times for 1 million iterations, allowing for a 10000 iteration burn in, the means were estimated on the basis of every 100th sample post burn-in, the CPU time was approximately 338 seconds for each run. The exchange move was accepted 23% of the time for (A), 59% (B), 33% (C1), 87% (C2). In Table 1 we can observe the estimated means. On the basis of the estimated means, (A) and (C1) have provided the best estimates of the means, with values all close to 1.5. We explain this as follows. For scheme (C1) the densities are slowly evolving to a very simple one, which means:

1. There is large amount of information that is relevant to sampling the target.
2. The sequence of densities allows for an efficient bridge to the easiest to sample density, which can provide valuable information in sampling π.

Both of these points are exemplified by the exchange acceptance rate which shows that we can share information in the population efficiently. For algorithm (A) we note that we have the lowest exchange acceptance rate, which indicates that this approach is best used with N large (larger than 20 for this example); this will reduce the 'gaps' between densities. As a result, we conclude that we can more efficiently adopt other tempering strategies (that is, produce better results with fewer densities). We note, however, that points (1)

and (2) above seem to be satisfied for algorithm (A) which is why it has performed well. Heating schedule (B) is moderately successful: the fact that the target is quite 'far' from the other densities indicates that the algorithm suffers from a slight bottleneck. The algorithm (C2) is the worst; the exchange probabilities are quite high, and the mean estimates are very poor. That this occurs is unsurprising; there are not enough densities at high temperatures to provide a fast exploration of the state-space.

On the basis of this experiment, it appears that (A) and (C1) are the best strategies to adopt. In the case of (C1), the densities evolve very slowly to one which is easy to sample. (A) has provided consistent results (across various runs) and can be a useful default setting of the temperature parameters.

2.7.2 *The number of distributions*

We now investigate the number of distributions. We ran our experiment using heating scheme (C) with $N = 5$ ($K = 0.001$, $\widetilde{\alpha} = 6$), $N = 10$ ($K = 0.001$, $\widetilde{\alpha} = 2$), $N = 15$ ($K = 0.001$, $\widetilde{\alpha} = 1.75$) and $N = 25$ ($K = 0.001$, $\widetilde{\alpha} = 1.35$). The algorithms were all run for 1 million iterations; the CPU times were quite similar (as the cost of the algorithm, per iteration, does not increase with N). The results are summarized in Table 2.

In Table 2 we can observe that the estimates of the means generally improve with N, until we reach $N = 25$ when they become worse. This can be explained, for this example, as follows. The improved sampling from π is induced by the fact that for larger N a large amount of information about the target density π can be stored, simultaneously. In addition, the larger number of densities can make it easier to specify temperature parameters with fewer gaps. However, as the algorithm that we implement is of fixed computational complexity, as the number of chains increase, the increase in the theoretical convergence time of the Markov chain (see Sect. 5 for more discussion) is not counter-balanced by the former two points (for $N = 25$).

On the basis of the results seen here, we would recommend taking a reasonably large population relative to storage constraints. As noted in Sect. 2.4 this can improve the sampling if good population moves may be constructed. See Geyer and Thompson (1995) for some discussion on the number of distributions in the context of simulated tempering.

2.7.3 *The crossover move*

We now investigate the usefulness of the crossover move. To provide a sensible discussion we change the dataset to the well-known Galaxy data (see Richardson and Green 1997 and the references therein) and set $k = 6$; we explain why below.

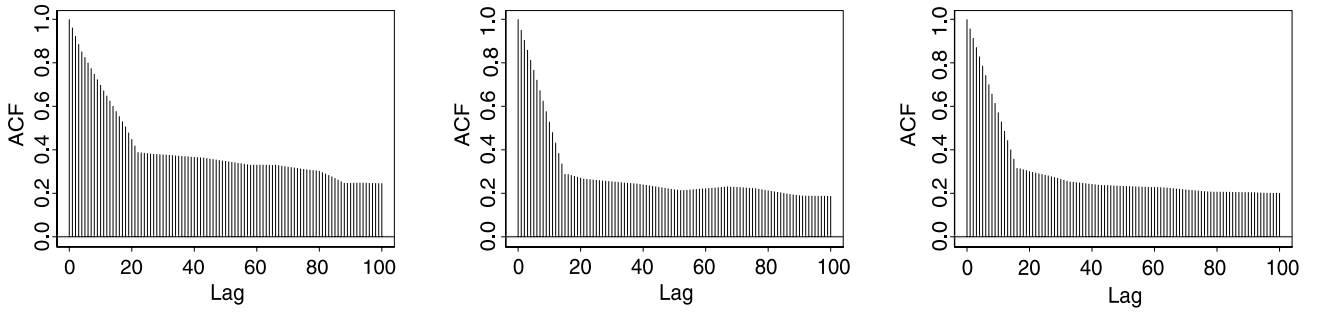
Table 2 Estimates of means from mixture comparison for various population sizes. We ran each sampler for 1 million iterations, 4 times. The estimates are presented in increasing order, for presentation purposes

N	Component			
	1	2	3	4
5	1.17	1.37	1.51	1.96
10	0.45	1.61	1.85	1.93
15	1.29	1.46	1.64	1.62
25	1.04	1.12	1.73	1.87

The algorithm is now modified to allow with probability $1/2$ to perform a crossover move, instead of an exchange step. This is performed as follows. Firstly, we select two chains with uniform probability. Secondly, we permute the component specific parameters so that we order on the component means. Thirdly, we draw a component $j = 1, \ldots, k$, with probability proportional to $1/j$, to crossover, swapping the component means and precisions to the left of j inclusive. We then accept or reject this move via the Hastings ratio and then permute the labels at random after the move; this ensures invariance of the composition of moves. Another potential way to crossover, is to try a circular crossover, that is, to allow for break points at different points in the component. We allowed the first and last components to be swapped in this move (for ease of programming and to investigate, potentially, the best case scenario in terms of being a local move). It is now apparent why we have changed the data. Performing this move when the data is as for the first part will yield very high acceptance rates due to the geometry of the parameter space (i.e. on permuting the means, the states will be virtually identical).

To investigate the performance of the crossover move, we ran three algorithms for 100000 iterations, one with only ordinary crossover (E1), circular crossover (E2) and only exchange (E3). We set $N = 10$ and adopted heating scheme (C) with $K = 0.001$, $\widetilde{\alpha} = 2$. To compare performances we consider the acceptance rates of crossover moves (0.49 (E1)) and (0.16 (E2)) as well as the autocorrelations of the (unnormalized) log likelihood (Fig. 2) (this is used over estimation of the means due to the permutation move used in crossover and the fact that this quantity is invariant to permutation).

From the acceptance rate of the ordinary crossover move, it appears that this is more effective. However, it should be noted that, in this move we will generally attempt to crossover less information; the acceptance rate of the circular crossover is still quite good. In Fig. 2 we can observe the autocorrelations of unnormalized log-likelihood of the chain of interest for every iteration (note the autocorrelation is quite high due to the fact that at some iterations we do not update the particular chain). We can see that the addition of a crossover move can vastly reduce the autocorrelations,



(a) Exchange. (b) Crossover and Exchange. (c) Circular Crossover and Exchange.

Fig. 2 Autocorrelations of the likelihoods. This is from the crossover comparison with the galaxy data. The algorithms were run for 100000 iterations

however, at the cost of an increased CPU time (55 seconds against 42 seconds). We conclude, as noted by Liang and Wong (2001), that the crossover can improve the exploration ability of parallel tempering, at the cost of increased computational cost and coding effort.

3 Sequential Monte Carlo methods

3.1 The method

We now describe a generic sequential Monte Carlo method, SMC samplers, which is appropriate to sample from a specific distribution π. Consider an identical situation to population-based MCMC, that is, we have a sequence of related densities $\{\pi_n\}_{n\in\mathbb{T}_p}$ on $(E,\mathcal{E})$ of which at least one is the target of interest (denote this π_p, note that we use p as opposed to N, since this represents a time parameter instead of the number of particles). The method begins by generating each particle $x_1^{(i)}$, $i=1,\ldots,N$ 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 estimating expectations; that is:

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

is a consistent estimator for $\mathbb{E}_{\pi_1}[h(X)]$.

Now to produce samples for π_2 we sample each particle from a Markov kernel $X_2^{(i)}|x_1^{(i)} \sim K_2(x_1^{(i)},\cdot)$ and reweight by using the incremental weights ω:

$$\begin{aligned}\omega_2(x_{1:2}^{(i)}) &= \frac{\pi_2(x_2^{(i)})\nu(x_1^{(i)})}{\pi_1(x_1^{(i)})\int_E \nu(x)K_2(x,x_2^{(i)})dx},\\ W_2(x_{1:2}^{(i)}) &= W_1(x_1^{(i)})\omega_2(x_{1:2}^{(i)}).\end{aligned} \tag{3.6}$$

The problem is that the integral in (3.6) can seldom be calculated. To remove this problem, Del Moral et al. (2006a) (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 (3.6) as:

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

We then take $\widetilde{\pi}_n$ to admit marginal π_n, $n=3,\ldots,p$. The algorithm continues by sampling from (potentially time inhomogeneous) Markov kernels $K_3,\ldots,K_n$; the algorithm is described fully in Algorithm 2. Note that $\widetilde{\pi}_1=\pi_1$.

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 $W_n(x^{(i)})/\sum_{i=1}^N W_n(x^{(i)})$. 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: for example stochastic remainder selection (Baker 1985) (leading to residual resampling (Liu and Chen 1998)) and systematic resampling (Whitley 1994). See Douc et al. (2005) for some comparisons of various resampling schemes in the context of particle filters. Note also, the pruned-enriched Rosenbluth method (Grassberger 1997) may be used. This is a different procedure to resampling. However, resampling is arguably easier to perform as, with the exception of an effective sample size (ESS) threshold (see below), it does not require tuning parameters to be set.

3.2 The algorithm

In Algorithm 2 we provide the basic sampling scheme. We note that the resampling criteria is based upon the effective sample size (Liu 2001). Resampling may occur at deterministic times, as opposed to that given in Algorithm 2. The position of the resampling step will depend upon the algebraic form of the incremental weights; if they are independent of the sampled state (as occurs below) then it is possible to resample the particles before they are mutated (this will improve the diversity of the population). Note that it is possible to derive continuous time versions of this algorithm; see Rousset (2006) and Rousset and Stoltz (2006).

One point of interest is that we are able to estimate ratios of normalizing constants as a by-product of the algorithm; see Neal (2001) and Del Moral et al. (2006a) for more details. Also see: Neal (2005), Johansen et al. (2006) and Rousset and Stoltz (2006) for more advanced techniques.

The theoretical justification for an SMC algorithm is essentially asymptotic (in the number of particles); see Crisan and Doucet (2000); Del Moral and Miclo (2000); Chopin (2004); Del Moral (2004); Künsch (2005) and Douc and Moulines (2006). Note, also, that the selection/mutation format of the algorithm can be interpreted as a particle approximation of a Feynman-Kac formulae (Del Moral 2004), which provides an elegant framework for the theoretical analysis of SMC methods.

Algorithm 2 (A sequential Monte Carlo sampler)

0. (INITIALIZATION)
   - Set $n = 1$.
   - 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 $((\sum_i (W_n(x_{1:n}^{(i)}))^2)/(\sum_j W_n(x_{1:n}^{(j)}))^2)^{-1} < L$ (L is some user set threshold), resample the particles and set all weights equal to 1.
2. (MUTATION)
   - Set $n = n + 1$, if $n = p + 1$ stop.
   - For $i = 1, \ldots, N$ draw $X_n^{(i)} \sim K_n(x_{n-1}^{(i)}, \cdot)$.
   - Reweight

$$\omega_n(x_{1:n}^{(i)}) = \frac{\widetilde{\pi}_n(x_{1:n}^{(i)})}{\widetilde{\pi}_{n-1}(x_{1:n-1}^{(i)})K_n(x_{n-1}^{(i)}, x_n^{(i)})},$$

$$W_n(x_{1:n}^{(i)}) = W_{n-1}(x_{1:n-1}^{(i)})\omega_n(x_{1:n}^{(i)}).$$

end

3.3 Specifying the auxiliary distributions

When constructing an SMC sampler we need to specify the auxiliary distributions $\{\widetilde{\pi}_n\}_{n\in\mathbb{T}_p}$. The approach used by Del Moral et al. (2006a) (also Jarzynski 1997 and Neal 2001) is to set:

$$\widetilde{\pi}_n(x_{1:n}) = \pi_n(x_n)\prod_{j=2}^{n} L_{j-1}(x_j, x_{j-1}), \tag{3.7}$$

where L_{n-1} is a *backward* in time Markov kernel. Thus the incremental weights become:

$$\omega_n(x_{1:n}^{(i)}) = \frac{\pi_n(x_n^{(i)})L_{n-1}(x_n^{(i)}, x_{n-1}^{(i)})}{\pi_{n-1}(x_{n-1}^{(i)})K_n(x_{n-1}^{(i)}, x_n^{(i)})}.$$

Del Moral et al. (2006a) show that the optimal backwards kernel $\{L_{n-1}^{\text{opt}}\}$ (in terms of minimizing the variance of the importance weights) should be taken as:

$$L_{n-1}^{\text{opt}}(x_n, x_{n-1}) = \frac{\nu_{n-1}(x_{n-1})K_n(x_{n-1}, x_n)}{\nu_n(x_n)},$$

$$\nu_n(x_n) = \int \nu(x_1)K_2(x_1, x_2)\cdots K_n(x_{n-1}, x_n)dx_{1:n-1}.$$

That is, the importance weights are of the form we started with (i.e. calculating an intractable integral). Due to the fact that such backward kernels cannot be calculated (otherwise we would not have resorted to extending the state space), Del Moral et al. (2006a) suggest a variety of alternatives, including (also in Neal 2001):

$$L_{n-1}(x_n, x_{n-1}) = \frac{\pi_n(x_{n-1})K_n(x_{n-1}, x_n)}{\pi_n(x_n)} \tag{3.8}$$

with K_n an MCMC kernel with invariant distribution π_n; this will mean that the integral in (3.8) 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). Additional move types, including Gibbs steps, can be found in Del Moral et al. (2006b).

To select the densities, as long as $\pi_p \equiv \pi$, any of the choices in Sect. 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 Sect. 3.5.

In addition, it is also possible to derive algorithms such that the kernel is a mixture of moves:

$$K_n(x_{n-1}, x_n) = \sum_{i=1}^{D} \alpha_i(x_{n-1}) K_{n,i}(x_{n-1}, x_n)$$

with associated backward kernel:

$$L_{n-1}(x_n, x_{n-1}) = \sum_{i=1}^{D} \beta_i(x_n) L_{n-1,i}(x_n, x_{n-1}).$$

It can be computationally expensive to evaluate the mixture of kernels, and so we can extend the state space to $E^n \times \mathbb{T}_D^n$ and simulate an indicator at time n (with probability $\alpha_i(x_{n-1})$) and compute the incremental weight:

$$\omega_n(x_{1:n}^{(i)}, j) = \frac{\pi_n(x_n^{(i)})\beta_j(x_n^{(i)})L_{n-1,j}(x_n^{(i)}, x_{n-1}^{(i)})}{\pi_{n-1}(x_{n-1}^{(i)})\alpha_j(x_{n-1}^{(i)})K_{n,j}(x_{n-1}^{(i)}, x_n^{(i)})}. \quad (3.9)$$

3.4 Relation to other methods

In this section we show that some other SMC methods fall into the framework of Del Moral et al. (2006a).

The annealed importance sampler (AIS) of Neal (2001) corresponds to the case with K_n is an MCMC kernel with invariant distribution π_n with backward kernel $\pi_n(x_{n-1})K_n(x_{n-1}, x_n)/\pi_n(x_n)$. Note that this method does not use resampling and this can often make it more difficult to specify simulation parameters; see Del Moral et al. (2006a) for a comparison of SMC samplers and AIS. A point of interest, is that this method cannot be adopted if the support of each density is nested; see Del Moral et al. (2006a, 2006b) for further details.

The population Monte Carlo method of Cappé et al. 2004 occurs in the situation where we have a homogeneous sequence of targets π, $L(x', x) = \pi(x)$ and $K(x, x') = q(x')$, with q some importance distribution, which is typically adapted on the fly. In the next generation of this methodology, the D-kernel approach (Douc et al. 2006a, 2006b) perform a similar method, where the importance function is a time adaptive mixture of Markov kernels:

$$K_n(x_{n-1}, x_n) = \sum_{i=1}^{D} \alpha_{n,i} K_i(x_{n-1}, x_n).$$

In this case we have $\pi_n \equiv \pi$, $L(x, x') = \pi(x')$ and K_n as above is a time adaptive kernel (in the case of Douc et al. 2006b) whose mixture weights are updated during iterations. For the algorithm of Douc et al. (2006a), there is an auxiliary variable version of this algorithm as in (3.9). The objective of the D-kernel method is to allow the algorithm to select an importance distribution which is optimal, in terms of minimizing the asymptotic variance of self-normalized estimates (in Douc et al. 2006b); in Douc et al. (2006a) it is, primarily, the Kullback divergence between the target and the proposal (on a product space). We note a potential drawback of this methodology is that the kernel needs to be evaluated pointwise, in order to calculate the importance weights. This immediately removes MH kernels (and thus many trans-dimensional problems) and will restrict the potential user to kernels, among others, such as Gibbs updates or random walk moves (also prior approximations of π). In the case of the former, it will not always be the case that we are able, in complex problems, to construct a collection (or a single) of Gibbs samplers (e.g. different blocking strategies); we note, however, if indeed we are able to do so, the D-kernel procedure may be very useful. In the setting of the latter, it can be very difficult to select kernels, in high dimensions, so that the variance of the importance weights is finite; that is, such a method is not always going to be useful in problems where population-based methods are really needed. As noted by one of the referees, the D-kernel approach is time adaptive as opposed to sequential. This means that the algorithm seeks to use the experience in the past to provide a correction of the methodology which is not present in the SMC samplers we have presented. However, we argue that such a procedure can fall directly into an SMC sampler framework. In the static case, which is our concern, we can set $\pi_{n-1} \approx \pi_n$ and thus information at time $n-1$ is relevant for determining the properties of K_n; we are free to use all of the adaptive MCMC procedures (Robbins-Monro, MH independence sampler with proposal approximating the target (Andrieu and Moulines 2006)) in the literature. Further this can be effective, as the tempering procedure allows us to select good kernels initially and thus correct potentially poor kernels in the future.

The approach of Liang (2002) falls under the case $\pi_n \equiv \pi$ $\forall n$, $K_n(x, x') = L_n(x', x) = K(x, x')$ (as applied in practice), with the exception that the weights are updated via dynamic weighting (Wong and Liang 1997). One potential drawback is that the weights, on unbounded state-spaces, are likely to have infinite variance (although all the examples in

Liang 2002 are indeed on compact spaces); this is not necessarily the case for SMC samplers. In addition, the *asymptotic* variance (of sample path averages in the central limit theorem) for SMC can be shown to be upper-bounded for any finite time horizon for the case of AIS (with resampling) assuming geometric ergodicity of the Markov kernels (Jasra and Doucet 2006); see Chopin (2004) and Künsch (2005) for analysis in the context of filtering and Sect. 5 for a discussion on the bounds on the actual Monte Carlo variance.

3.5 Example

We now provide an example on how to construct and apply a SMC sampler; we use the mixture example of Sect. 1.1. We consider two tempering approaches: tempered densities and data point tempered. Note that Fearnhead and Meligkotsidou (2007) have a method for simulating from mixture models using particle approaches; we do not consider that here.

3.5.1 Two tempering approaches: algorithm details

Tempered densities We take the target distributions to be:

$$\pi_n(\phi_{1:k}, w_{1:k-1}|y_{1:m}) \propto l(y_{1:m}; \phi_{1:k}, w_{1:k-1})^{\zeta_n} p(\phi_{1:k}, w_{1:k-1}), \quad n \in \mathbb{T}_p.$$

We will adopt the MCMC kernels in Sect. 2.7, adapted to the sequence of densities used for SMC. We allowed, with probability 1/2 for an application of the kernel to be 10 iterates of the sweep over $\mu_{1:k}$, $\lambda_{1:k}$ and $w_{1:k-1}$. The backward kernel (3.8) is adopted and $p = 100$. Thus, using the invariance of the MCMC kernels, for $n \in \mathbb{T}_p$ we have incremental weight (adopting the convention $\zeta_0 = 0$):

$$\omega_n(\phi_{1:k,n-1}, w_{1:k-1,n-1}) = l(y_{1:m}; \phi_{1:k,n-1}, w_{1:k-1,n-1})^{\zeta_n - \zeta_{n-1}}.$$

For the temperature parameters, $\{\zeta_n\}$ increased from 0 to 15/100 for the first 20 time points then from 15/100 to 40/100 for the next 40 and finally from 40/100 to 1 for the last 40 time points. We note that the specific cooling scheme (e.g. logarithmic, power decay) can more easily be chosen (than population MCMC), in a problem specific way via ESS plots (see the next section). We recommend, generally, that the densities evolve slowly, initially, and then more quickly. The automatic construction of a temperature sequence is the subject of current research work.

One point to be discussed is the choice of p. This is a little clearer in the SMC context. If p is quite large, then we can allow the consecutive densities to be such that $\pi_{n-1} \approx \pi_n$; indeed it can be the case that resampling is not even needed (as the weight degeneracy is quite slow). This is at the cost of an increase in CPU time. The objective is then to set $\pi_{n-1} \approx \pi_n$, but not have p so large that CPU times take too long; this can be guided by the rate of resampling as discussed below. A thorough discussion of the choice of p can be found, for the example below, can be found in Del Moral et al. (2006a).

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 incremental weight for $n = 2, \ldots, m$:

$$\omega_l(\phi_{1:k,n-1}, w_{1:k-1,n-1}) = l(y_n; \phi_{1:k,n-1}, w_{1:k-1,n-1}).$$

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.

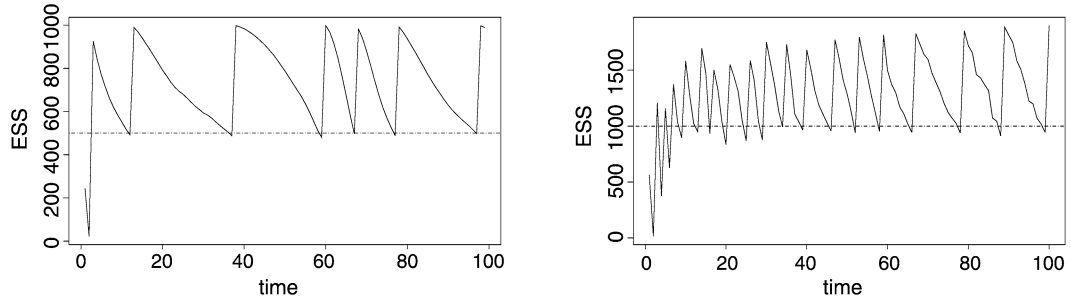
For illustration we chose population sizes of 1000 and 2000 particles for the tempered and data point tempered approaches respectively. We observed extremely poor performance for the adding data method with population size 1500 or smaller, so we included a larger set of particles for this approach (also to make a reasonable comparison in terms of CPU time between the tempering methods).

We applied the systematic resampling approach before mutating the particles, resampling upon the basis of when the ESS drops below $N/2$. We note that the order at which the data appears is important in the data point tempered approach, and we ordered the data so that information came from each of the modes $(-3, 0, 3, 6)$ in turn; we discuss this further below.

3.5.2 Performance of tempering schemes on the simulated data

The CPU times were 185 and 197 seconds for the tempered and data point tempered procedures. In Table 3 and in Fig. 3 we can observe the performance of the two SMC samplers.

In Table 3 we can see the estimated means. It is clear that the tempered approach has allowed for a significantly better estimation of these quantities. This is because the values of the means are reasonably similar for the tempered densities. The data point tempered approach has been unable to explore each of the 4! symmetric modes in the correct proportion.



(a) Tempered Densities. (b) Data Point Tempered.

Fig. 3 Effective sample size plots from the SMC samplers. We fitted a four component normal mixture to the data, the output is a single population of 1000 samples from an SMC sampler using 100 tempered densities (**a**) and a single population of 2000 samples from an SMC sampler which had densities change by adding data points (**b**). For both plots, the *horizontal line* is the resampling threshold

Table 3 Estimates of means from mixture comparison for the two tempering procedures. We ran each sampler for 100 time-points, 4 times with 1000 (tempering) and 2000 (data point tempering) particles respectively. The estimates are presented in increasing order, for presentation purposes

N	Component			
	1	2	3	4
Tempered	1.24	1.39	1.51	1.61
Data Point Tempered	0.88	1.20	1.82	1.87

Figure 3 displays the ESS. For the tempered densities (Fig. 3(a)) we see the typical weight degeneracy problem, and the resampling allows us to remove the problem successfully (note that the plot indicates that initializing the algorithm from the prior is a very poor choice; see Del Moral et al. 2006b for alternative approaches). The adding data points plot (Fig. 3(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.

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. 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 does not appear to be 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 (see Chopin 2007 for some examples in hidden Markov modelling).

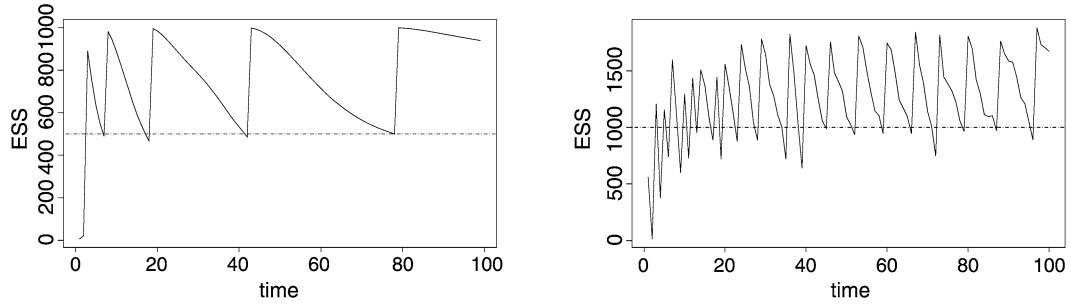
3.5.3 Link between temperatures and data order with resampling

We now discuss the importance of the temperature parameters $\{\zeta_n\}$. In Fig. 3(a) we observe the resampling rate of the algorithm. In Fig. 3(a) the resampling rate is reasonably consistent. The main factor affecting this is the choice of temperature parameter and Markov kernel (note that the resampling threshold is clearly another element which affects resampling); in this example the latter mixes quite fast so we address the former point. 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 $p = 100$, for a cooling scheme:

$$\zeta_n = \frac{\log\{n\}}{\log\{100\}}, \quad n = 2, \ldots, 100$$

and $\zeta_1 = 1$. In Fig. 4(a) we can observe the rate of resampling (under similar Markov kernels to those in Fig. 3(a)). We see that we have resampled less (for the log cooling scheme), and appear to have a slightly more consistent rate of resampling; this can manifest itself in terms of better point estimates (although this does not occur for this run—we resample close to time p). 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.



(a) Tempered Densities.

(b) Data Point Tempered.

Fig. 4 Effective sample size plots from the SMC samplers. We fitted a four component normal mixture to the data, the output is a single population of 1000 samples from an SMC sampler using 100 tempered densities (log cooling sequence) (**a**) and a single population of 2000 samples from an SMC sampler which had densities change by adding data points (**b**) (different order to Fig. 3(b)). For both plots, the horizontal line is the resampling threshold

For the data point tempering, the order of adding data can have an impact upon the performance of the algorithm. We ran the data point tempered algorithm, with a random choice of data arriving (Fig. 4(b)). We can thus see that 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).

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 (propagation of chaos properties; see Del Moral 2004), 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 Sect. 3.5, 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 (see Liu 2001 and Iba 2001 for some discussion on what is meant by a reasonable acceptance rate) 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. It should be noted 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. It is recommended that 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, in our experience, data point tempering can perform very badly for MCMC methods.

4.3 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 backward Markov kernel for such moves. For example, if we wish to calculate the sub-optimal choice (3.8), 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 for 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, when the kernels mix slowly, it is easier to compensate for this via a long tempering sequence in SMC (which can be more effective than many iterations in population MCMC). This is demonstrated in our examples, by the fact that the SMC sampler with tempered densities (Table 3) provides comparable results to the best population MCMC (Table 1), with slightly more than half the CPU time.

4.4 Summary

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

(A) MCMC is often easier to calibrate (in terms of simulation parameters).
(B) SMC requires no burn-in.
(C) 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 et al. 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. 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 can be 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 and 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 and 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 and 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. One important point, that is unclear to us, is whether as $N \to \infty$ (or in practice large N) population MCMC improves (for reasonable population moves and not necessarily fixed computational cost per iteration; e.g. mutation for each chain at every step). It is not obvious that MCMC algorithms will fail completely (due to an increase in the size of the state-space) or improve (due to the increase of information). This may be investigated using spectral gap analysis; see Diaconis and Saloff-Coste (1993) and Eberle and Marinelli (2006) for a starting point. Additionally, it is important to produce fast mixing transition kernels. One way this may be achieved is to use adaptive methods (see Jasra et al. 2005c in the trans-dimensional case via SMC 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 and Moulines (2006) and Roberts and Rosenthal (2005) for finite dimensional analysis and Andrieu et al. (2007a) for infinite dimensional approaches.

Possible research areas in SMC is as follows. An aspect of interest is when and how to resample. For example, as noted by Chen et al. (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); see Del Moral (2004) for coverage with the earlier multilevel Feynman-Kac formulae approach. 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). One area that appears to be under utilized in the static case, is the usage of advanced strategies typically used in sequential problems, for example rejection control (Liu et al. 1998) and pilot-exploration resampling (Zhang and Liu 2002). However, due to the nature of these techniques, it may not always be computationally efficient to use the methods; for example it is conceivable that the method of Liu et al. (1998) may improve SMC methods (in the static case), but the value of the pilot exploration scheme is of less clear (due to the fact that consecutive artificial densities will be similar, providing good guidance for the future). Finally, a very important point are $\mathbb{L}_p$-bounds for centered sample path averages of unbounded, integrable functions; whilst weak laws of large numbers exist (e.g. Douc and Moulines 2006), the bounds are currently only for bounded functions (Del Moral 2004, Chap. 7).

Acknowledgements 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. We thank three referees and the editor in chief for many useful comments that have allowed us to improve, substantially, the paper. We also thank Mark Briers for some comments on earlier versions of the paper.

References

Andrieu, C., Moulines, É.: On the ergodicity properties of some adaptive MCMC algorithms. Ann. Appl. Probab. **16**, 1462–1505 (2006)

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

Andrieu, C., Jasra, A., Doucet, A., Del Moral, P.: Non-linear Markov chain Monte Carlo via self interacting approximations. Technical Report, University of Bristol (2007a)

Andrieu, C., Jasra, A., Doucet, A., Del Moral, P.: A note on the convergence of the equi-energy sampler. Technical Report, University of Bristol (2007b). Stoch. Anal. Appl. (to appear)

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

Atchadé, Y.F., Liu, J.S.: Discussion of the 'equi-energy sampler'. Ann. Stat. **34**, 1620–1628 (2006)

Baker, J.E.: Adaptive selection methods for genetic algorithms. In: Grefenstette, J. (ed.) Proc. Intl. Conf. on Genetic Algorithms and Their Appl., pp. 101–111. Erlbaum, Mahwah (1985)

Brockwell, A.E., Del Moral, P., Doucet, A.: Sequentially interacting Markov chain Monte Carlo for Bayesian computation. Technical Report, Carnagie Mellon University (2007)

Cappé, O., Guillin, A., Marin, J.M., Robert, C.P.: Population Monte Carlo. J. Comput. Graph. Stat. **13**, 907–925 (2004)

Chen, Y., Xie, J., Liu, J.S.: Stopping-time resampling for sequential Monte Carlo methods. J. Roy. Stat. Soc. Ser. B **67**, 199–217 (2005)

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

Chopin, N.: Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. Ann. Stat. **32**, 2385–2411 (2004)

Chopin, N.: Inference and model choice for time-ordered hidden Markov models. J. Roy. Stat. Soc. Ser. B (2007, to appear)

Crisan, D., Doucet, A.: Convergence of sequential Monte Carlo methods. Technical Report, University of Cambridge (2000)

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

Del Moral, P., Doucet, A.: On a class of genealogical and interacting Metropolis models. In: Azéma, J., Emery, M., Ledoux, M., Yor, M. (eds.) Séminaire de Probabilités XXXVII. Lecture Notes in Math., vol. 1832, pp. 415–446. Springer, Berlin (2003)

Del Moral, P., Miclo, L.: Branching and interacting particle systems approximations of Feynman-Kac formulae with applications to non-linear filtering. In: Séminaire de Probabilitiés XXXIV. Lecture Notes in Math., vol. 1729, pp. 1–145. Springer, Berlin (2000)

Del Moral, P., Doucet, A., Jasra, A.: Sequential Monte Carlo samplers. J. Roy. Stat. Soc. Ser. B **68**, 411–436 (2006a)

Del Moral, P., Doucet, A., Jasra, A.: Sequential Monte Carlo for Bayesian computation (with discussion). In: Bayarri, S., Berger, J.O., Bernardo, J.M., Dawid, A.P., Heckerman, D., Smith, A.F.M., West, M. (eds.) Bayesian Statistics 8 (2006b, in press)

Diaconis, P., Saloff-Coste, L.: Comparison theorems for reversible Markov chains. Ann. Appl. Probab. **3**, 696–730 (1993)

Douc, R., Moulines, É.: Limit theorems for weighted samples with applications to sequential Monte Carlo methods. Technical Report, Centre de Mathématiques Appliquées, École Polytechnique (2006). Ann. Stat. (to appear)

Douc, R., Cappé, O., Moulines, É.: Comparison of resampling schemes for particle filtering. In 4th International Symposium on Image and Signal Processing and Analysis (ISPA) (2005)

Douc, R., Guillin, A., Marin, J.M., Robert, C.P.: Convergence of adaptive sampling schemes. Ann. Stat. (2006a, in press)

Douc, R., Guillin, A., Marin, J.M., Robert, C.P.: Minimum variance importance sampling via population Monte Carlo. Technical Report, Université Paris-Dauphine (2006b). ESIAM Probab. Stat. (to appear)

Doucet, A., Godsill, S., Andrieu, C.: On sequential Monte Carlo sampling for Bayesian filtering. Stat. Comput. **10**, 197–208 (2000)

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

Eberle, A., Marinelli, C.: Convergence of sequential Markov chain Monte Carlo methods I: Non-linear flow of probability measures. Technical Report, Universität Bonn (2006)

Fearnhead, P., Meligkotsidou, L.: Filtering methods for mixture models. J. Comput. Graph. Stat. (2007, to appear)

Gelman, A., Meng, X.L.: Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. Stat. Sci. **13**, 163–185 (1998)

Geyer, C.J.: Markov chain maximum likelihood. In: Keramigas, E. (ed.) Computing Science and Statistics: The 23rd Symposium on the Interface, pp. 156–163. Interface Foundation, Fairfax (1991)

Geyer, C.J., Thompson, E.A.: Annealing Markov chain Monte Carlo with applications to ancestral inference. J. Am. Stat. Assoc. **90**, 909–920 (1995)

Gilks, W.R., Roberts, G.O., George, E.I.: Adaptive direction sampling. The Statistician **43**, 179–189 (1994)

Gilks, W.R., Berzuini, C.: Following a moving target—Monte Carlo inference for dynamic Bayesian models. J. Roy. Stat. Soc. Ser. B **63**, 127–146 (2001)

Goswami, G.R., Liu, J.S.: On learning strategies for evolutionary Monte Carlo. Stat. Comput. **17**, 23–28 (2007)

Grassberger, P.: Pruned-enriched Rosenbluth method: simulations of θ polymers of chain length up to 1 000 000. Phys. Rev. E **56**, 3682–3693 (1997)

Green, P.J.: Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika **82**, 711–732 (1995)

Green, P.J., Mira, A.: Delayed rejection in reversible jump Metropolis-Hastings. Biometrika **88**, 1035–1053 (2001)

Hammersley, J.M., Morton, K.W.: Poor man's Monte Carlo. J. Roy. Stat. Soc. Ser. B **16**, 23–38 (1999)

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

Heard, N.A., Holmes, C.C., Stephens, D.A.: A quantitative study of gene regulation involved in the immune response of anopheline mosquitoes: an application of Bayesian hierarchical clustering of curves. J. Am. Stat. Assoc. **101**, 18–29 (2006)

Hukushima, K., Nemoto, K.: Exchange Monte Carlo method and application to spin glass simulations. J. Phys. Soc. Jpn. **65**, 1604–1608 (1996)

Iba, Y.: Population Monte Carlo algorithms. Trans. Jpn. Soc. Artif. Intell. **16**, 279–286 (2000)

Iba, Y.: Extended ensemble Monte Carlo. Int. J. Mod. Phys. **12**, 653–656 (2001)

Jarzynski, C.: Nonequilibrium equality for free energy differences. Phys. Rev. Lett. **78**, 2690–2693 (1997)

Jasra, A.: Bayesian inference for mixture models via Monte Carlo computation. PhD thesis, Imperial College London (2005)

Jasra, A., Doucet, A.: Stability of sequential Monte Carlo samplers via the Foster-Lyapunov condition. Technical Report, University of British Columbia (2006)

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

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

Jasra, A., Doucet, A., Stephens, D.A., Holmes, C.C.: Interacting sequential Monte Carlo samplers for trans-dimensional simulation. Technical Report, Imperial College London (2005c)

Johansen, A., Del Moral, P., Doucet, A.: Sequential Monte Carlo samplers for rare event estimation. Technical Report, University of Cambridge (2006)

Kou, S.C., Zhou, Q., Wong, W.H.: Equi-energy sampler with applications to statistical inference and statistical mechanics. Ann. Stat. **32**, 1581–1619 (2006)

Künsch, H.R.: Recursive Monte Carlo filters; algorithms and theoretical analysis. Ann. Stat. **33**, 1983–2021 (2005)

Liang, F.: Dynamically weighted importance sampling in Monte Carlo computation. J. Am. Stat. Assoc. **97**, 807–821 (2002)

Liang, F.: Use of sequential structure in simulation from high-dimensional systems. Phys. Rev. E **67**, 056101–056107 (2003)

Liang, F., Wong, W.H.: Real parameter evolutionary Monte Carlo with applications to Bayesian mixture models. J. Am. Stat. Assoc. **96**, 653–666 (2001)

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

Liu, J.S., Chen, R.: Sequential Monte Carlo methods for dynamic systems. J. Am. Stat. Assoc. **93**, 1032–1044 (1998)

Liu, J.S., Chen, R., Wong, W.H.: Rejection control and sequential importance sampling. J. Am. Stat. Assoc. **93**, 1022–1031 (1998)

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

Marinari, E., Parisi, G.: Simulated tempering; a new Monte Carlo scheme. Europhys. Lett. **19**, 451–458 (1992)

Matthews, P.: A slowly mixing Markov chain and its implication for Gibbs sampling. Stat. Probab. Lett. **17**, 231–236 (1993)

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

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equations of state calculations by fast computing machines. J. Chem. Phys. **21**, 1087–1092 (1953)

Mitsutake, A., Sugita, Y., Okamoto, Y.: Replica-exchange multicanonical and multicanonical replica exchange Monte Carlo simulations of peptides. I. Formula and benchmark tests. J. Chem. Phys. **118**, 6664–6676 (2003)

Neal, R.M.: Sampling from multimodal distributions using tempered transitions. Stat. Comput. **4**, 353–366 (1996)

Neal, R.M.: Annealed importance sampling. Stat. Comput. **11**, 125–139 (2001)

Neal, R.M.: Estimating ratios of normalizing constants using linked importance sampling. Technical Report, University of Toronto (2005)

Pritchard, J.K., Stephens, M., Donnelly, P.: Inference of population structure using multilocus genotype data. Genetics **155**, 945–959 (2001)

Richardson, S., Green, P.J.: On Bayesian analysis of mixture models with an unknown number of components (with discussion). J. Roy. Stat. Soc. Ser. B **59**, 731–792 (1997)

Robert, C.P., Casella, G.: Monte Carlo Statistical Methods, 2nd edn. Springer, New York (2004)

Robert, C.P., Rydén, T., Titterington, D.M.: Bayesian inference in hidden Markov models through reversible jump Markov chain Monte Carlo. J. Roy. Stat. Soc. Ser. B **62**, 57–75 (2000)

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

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

Ron, D., Swendson, R.H., Brandt, A.: Inverse Monte Carlo renormalization group transformations for critical phenomena. Phys. Rev. Lett. **89**, 275701–275705 (2002)

Rousset, M.: Continuous time population Monte Carlo and computational physics. PhD thesis, Universitié Paul Sabatier, Toulouse (2006)

Rousset, M., Stoltz, G.: Equilibrium sampling from nonequilibrium dynamics. J. Stat. Phys. **123**(6), 1251–1272 (2006)

Warnes, A.: The normal kernel coupler: an adaptive Markov chain Monte Carlo method for efficiently sampling from multimodal distributions. PhD thesis, University of Washington (2001)

Whitley, D.: A genetic algorithm tutorial. Stat. Comput. **4**, 65–85 (1994)

Wong, W.H., Liang, F.: Dynamic weighting in Monte Carlo optimization. Proc. Nat. Acad. Sci. **94**, 14220–14224 (1997)

Zhang, J.L., Liu, J.S.: A new sequential importance sampling method and its application to the two dimensional hydrophobic-hydrophilic model. J. Chem. Phys. **117**, 3492–3498 (2002)

Zheng, Z.: On swapping and simulated tempering algorithms. Stoch. Process. Appl. **104**, 131–153 (2003)