Population Monte Carlo

Olivier Cappé†
CNRS / ENST Département TSI, Paris
Arnaud Guillin
Jean–Michel Marin
CEREMADE, Université Paris Dauphine, Paris
Christian P. Robert†‡
CEREMADE, Université Paris Dauphine, and CREST, INSEE, Paris

Summary. Importance sampling methods have been rather neglected by MCMC algorithms since their infancy, even though they share many common features. This paper shows that importance sampling can be iterated to produce more accurate approximations to iid sampling from a target distribution, than sequential sampling from an MCMC algorithm. We first illustrate the adaptability of the joint scheme on a toy mixture example. As a more realistic example, we then reanalyse the ion channel model of Hodgson (1999), using an importance sampling scheme based on a hidden Markov representation. The degeneracy phenomenon that usually occurs in particle systems is studied on both examples.

Keywords: Adaptive algorithm, degeneracy, hidden semi-Markov model, importance sampling, ion channel model, MCMC algorithms, mixture model, multiple scales, particle system, random walk, reversible jump, unbiasedness

1. Introduction

When reviewing the literature on MCMC methodology, an obvious feature is that it has predominantly focussed on producing *single* outputs from a given target distribution, π. This may sound a paradoxical statement when considering that one of the major applications of MCMC algorithms is the approximation of integrals

$$\mathfrak{I} = \int h(x)\pi(x)dx$$

with empirical sums

$$\hat{\mathfrak{I}} = \frac{1}{T}\sum_{t=1}^{T} h(x^{(t)})\,,$$

where $(x^{(t)})$ is a Markov chain with stationary distribution π. But the main issue is that π is considered as the limiting distribution of x_t *per se* and that the Markov correlation between

†Research partially supported by EU TMR network ERB–FMRX–CT96–0095 on *"Computational and Statistical Methods for the Analysis of Spatial Data"* and CREST, Insee, Paris.

‡*Address for correspondence:* CEREMADE, Université Paris Dauphine, 16 place du Maréchal de Lattre de Tassigny, 75775 Paris cedex 16
E-mail: xian@ceremade.dauphine.fr

the x_t's is evacuated through the ergodic theorem (Meyn and Tweedie, 1993). There only exist a few references to the use of MCMC algorithms for the production of samples from π, including Warnes (2001) and Mengersen and Robert (2002), although the concept is not very original compared with the production of a single output from the target distribution.

Another striking (and related) feature of the MCMC literature is the early attempt to dissociate itself from [pre]existing techniques such as *importance sampling*, although the latter shared with MCMC algorithms the property of simulating from the wrong distribution to produce [approximate] results from the correct distribution (see Robert and Casella, 1999, Chap. 3). It is only lately that the realisation that both approaches could be successfully coupled came upon the MCMC community, as shown for instance by MacEachern and Peruggia (2000), Liu (2001), or Liu *et al.* (2001).

One clear example of this fruitful symbiosis is given by *iterated particle systems* (Doucet *et al.*, 2001). Originally, iterated particle systems were introduced to deal with dynamic target distributions, as for instance in radar tracking, where the imperatives of on-line processing of rapidly changing target distributions prohibited to resort to repeated MCMC sampling. The basic idea consisted in recycling previous weighted samples primarily through a modification of the weights (Gordon *et al.*, 1993), later supported with additional sampling steps (Berzuini *et al.*, 1997; Pitt and Shephard, 1999). As pointed out in Chopin (2002), a particle system simplifies into a particular type of importance sampling scheme in a static—as opposed to dynamic—setup.

We therefore study in this paper a method, *population Monte Carlo*, that tries to link these different "loose ends" into a superior simulation technology. It borrows from MCMC algorithms for the construction of the proposal, from importance sampling for the construction of appropriate estimators, from SIR (Rubin, 1987) for sample equalisation, and from iterated particle systems for sample improvement. The population Monte Carlo algorithm is thus an iterated scheme that produces, at each iteration, a sample approximately simulated from a target distribution *and* (approximately) unbiased estimates of integrals under that distribution. The sample is constructed using MCMC proposal for generation and importance sampling weights for pruning the proposed sample.

We describe in Section 2 the population Monte Carlo technique, and apply these development, first to a simple mixture example in Section 3, and second to the more ambitious ion channel model that we reassess in Section 4. While reasonable in complexity, the mixture example still offers an interesting media to assess the adaptativity of the population Monte Carlo sampler and the resistance to degeneracy. The ion channel model is more challenging in that there is no close form representation of the observed likelihood, while the completion step is more delicate than in mixture settings. In particular, Section 4.6 explains why a Metropolis–Hastings algorithm based on the same proposal as population Monte Carlo does not work. Section 5 contains the overall conclusions of the paper.

2. The population Monte Carlo approach

As noted in Mengersen and Robert (2003), it is possible to construct an MCMC algorithm associated with the target distribution

$$\pi^{\otimes n}(x_1, \ldots, x_n) = \prod_{i=1}^{n} \pi(x_i)\,,$$

on the space $\mathcal{X}^n$, rather than with the distribution $\pi(x_1)$, on the space $\mathcal{X}$. All standard results and schemes for MCMC algorithms apply in this particular case, and irreducible Markov chains associated with such MCMC algorithms converge to the target $\pi^{\otimes n}$ in distribution, that is, get approximately distributed as an iid sample from π after a "sufficient" number of iterations. Mengersen and Robert (2003) point out that additional sampling devices can be used to construct the proposal distributions, like Gibbs-type component-wise repulsive proposals that exclude immediate neighbourhoods of the other points in the sample. (Just as regular Metropolis–Hastings algorithms use the wrong distribution to converge to the correct distribution, their version can use dependent proposals to simulate from an independent target.)

In the current setting, we however restrict the choice of proposals to component-wise moves, that is, given $\mathbf{x}^{(t)} = (x_1^{(t)}, \ldots, x_n^{(t)})$, the components $x_i^{(t+1)}$ are generated from a proposal $q(x|x_i^{(t)})$. (We insist upon the fact that q is a transition kernel, not the proposal density in the usual Metropolis–Hastings sense, since, for reasons that should soon become clearer, we do not implement the corresponding Metropolis–Hastings acceptance step.)

Now, it is worth noticing that, rather than considering the difficult problem of assessing the convergence of a Markov chain to its stationary distribution, especially when the original dimension is multiplied by n, we can correct *at each iteration* for the use of the wrong distribution by *importance sampling*. Indeed, we can directly associate to each point $x_i^{(t)}$ in the n-dimensional sample $\mathbf{x}^{(t)} = (x_1^{(t)}, \ldots, x_n^{(t)})$ a weight

$$\varrho_i^{(t)} = \frac{\pi(x_i^{(t)})}{q_{it}(x_i^{(t)})}\,, \qquad i = 1, \ldots, n\,,$$

where q_{it} is the proposal distribution for simulating $x_i^{(t)}$. This definition has the obvious consequence that the estimators of the form

$$\mathfrak{I}_t = \frac{1}{n} \sum_{i=1}^{n} \varrho_i^{(t)} h(x_i^{(t)})$$

are *unbiased* for every integrable function h and *every iteration* t. Note the extension of regular importance sampling results to the case where the importance distribution for x_i depends on both i and t. As already indicated in Robert and Casella (1999) in a more restrictive setting, importance sampling estimators have the interesting property that the terms $\varrho_i^{(t)} h(x_i^{(t)})$ are uncorrelated, even when the proposal q_{it} depends on the whole past of the experiment:

$$\text{var}\,(\mathfrak{I}_t) = \frac{1}{n^2} \sum_{i=1}^{n} \text{var}\left(\varrho_i^{(t)} h(x_i^{(t)})\right) \tag{1}$$

due to the cancelling effect of the weights $\varrho_i^{(t)}$.

Obviously, in most settings, the distribution of interest π is unscaled and we have to use instead

$$\varrho_i^{(t)} \propto \frac{\pi(x_i^{(t)})}{q_{it}(x_i^{(t)})}\,, \qquad i = 1, \ldots, n\,,$$

scaled so that the weights $\varrho_i^{(t)}$ sum up to 1. In this case, the above unbiasedness property and the variance decomposition are lost, although they approximately hold. In fact, the

estimation of the normalising constant of π improves with each iteration t, since the overall sum

$$\varpi_t = \frac{1}{tn} \sum_{\tau=1}^{t} \sum_{i=1}^{n} \frac{\pi(x_i^{(\tau)})}{q_{i\tau}(x_i^{(\tau)})} \tag{2}$$

is a convergent estimator of the inverse of the normalising constant. Therefore, as t increases, ϖ_t is contributing less and less to the variability of $\Im_t$ and the above properties can be considered as holding for t large enough. Note in addition that, if the sum (2) is only based on the $(t-1)$ first iterations, the variance decomposition (1) still holds, via the same conditioning argument.

As pointed out in Rubin (1987), it is preferable, rather than updating the weights at each iteration, to resample (with replacement) n values $y_i^{(t)}$ from $(x_1^{(t)}, \ldots, x_n^{(t)})$ using the weights $\varrho_i^{(t)}$ (and possibly a variance reduction device as in Carpenter *et al.*, 1998). This partially avoids the *degeneracy* phenomenon, that is, the preservation of negligible weights and corresponding irrelevant points in the sample. The sample $(y_1^{(t)}, \ldots, y_n^{(t)})$ resulting from this sampling importance resampling (SIR) step is thus akin to an iid sample extracted from the weighted empirical distribution associated with $\pi^{\otimes n}(x_1, \ldots, x_n)$.

The novelty of the method proposed in this paper is that the iterated call to importance sampling based on the current SIR sample allows for a progressive selection of the most relevant points of the sample, via a selection process akin to the one underlying Berzuini *et al.*'s (1997) method. Indeed, without this importance resampling correction, a regular Metropolis–Hastings acceptance step for each point of the n-dimensional sample produces a parallel MCMC sampler which converges to the target $\pi^{\otimes n}$ in distribution. Similarly, a regular Metropolis–Hastings acceptance step for the whole vector $\mathbf{x}^{(t)}$ converges to $\pi^{\otimes n}$; the advantage in producing an iid sample at each step is balanced by the drawback that the acceptance probability decreases approximately as a power of n. If, instead, we pick at each iteration the points in the sample according to their importance weight $\varrho_i^{(t)}$, we improve the selection mechanism by removing the points that are the most incompatible with the target distribution π. This device also automatically corrects for poor choices of proposal distributions, to some extent. (However, if the proposal distribution is completely inappropriate and none of the simulated values is acceptable for the target distribution, the Monte Carlo scheme will not manage to recover acceptable values and, besides, it will not necessarily detect the poor fit.) In the example of the ion channel in Section 4.6, it actually occurs that a Metropolis–Hastings scheme based on the same proposal does not work well, while population Monte Carlo produces correct answers. Given that the SIR scheme works towards the selection of the points with the highest target density, this suggests choosing proposal distributions with heavy tails, in order to reach the tails of the target distribution with a reasonable probability at each proposal. The mixture example of Section 3 illustrates the natural adaptability of the algorithm, which allows us to implement simultaneously several proposals and select on-line the most performant of them.

This new adaptive scheme produces in the end a sample in the parameter space that is close to an iid sample from the true posterior distribution, although the degree of approximation involved there is yet unclear. At worst, it can serve as a starting distribution for an MCMC sampler based on the importance function as proposal distribution (see Section 4.6), although it is sufficient for integral approximations, just like any static importance sampling estimator.

Note that adaptive importance sampling strategies were already considered in the pre-

MCMC area. See, e.g., Oh and Berger (1992,1993). In the MCMC setup, adaptive algorithms are less common because the adaptativity, that is, the ability to use the past behaviour to correct the proposal distribution, cancels the Markovian nature of the sequence and thus calls for more elaborate convergence studies. See, e.g., Andrieu and Robert (2001) and Haario *et al.* (1999,2001) for recent developments in this area.

3. Mixture model

Our first example is a Bayesian modelling of a mixture model, which is a problem simple enough to introduce but complex enough to lead to poor performances for badly designed algorithms (Robert and Casella, 1999, Chap. 9; Cappé *et al.*, 2002). The mixture problem we consider is based on an iid sample $\mathbf{x} = (x_1, \ldots, x_n)$ from the distribution

$$p\mathcal{N}(\mu_1, \sigma^2) + (1-p)\mathcal{N}(\mu_2, \sigma^2),$$

where $p \neq 1/2$ and $\sigma > 0$ are known. The prior associated with this model, π, is a normal $\mathcal{N}(\theta, \sigma^2/\lambda)$ prior on both μ_1 and μ_2. We thus aim at simulating from the posterior distribution

$$\pi(\mu_1, \mu_2|\mathbf{x}) \propto f(\mathbf{x}|\mu_1, \mu_2)\, \pi(\mu_1, \mu_2)\,.$$

Although the "standard" MCMC resolution of the mixture problem is to use a Gibbs sampler based on a data augmentation step via indicator variables, recent developments (Celeux *et al.*, 2000; Chopin, 2002; Cappé *et al.*, 2002) have shown that the data augmentation step is not necessary to run an MCMC sampler. We will now demonstrate that a population Monte Carlo sampler can be efficiently implemented without this augmentation step either.

Our adaptative importance sampling scheme will be as follows: The initialization step consists first in choosing a set of initial values for μ_1 and μ_2 (e.g., a grid of points around the empirical mean of the x_i's). The importance functions are then random walks, that is, random isotropic perturbations of the points of the current particle system. As noted above, a very appealling feature of the Population Monte Carlo method is that the importance function may vary from one particle to another without jeopardizing the validity of the method and, in particular, its unbiasedness. At a first level, the importance functions are all different, since they are normal distributions centerered in every particle. At a second level, we can also choose different variances for these normal distributions, for instance in a predetermined set of p scales v_i $(1 \leq i \leq p)$ ranging from 10^3 down to 10^{-3}, and select these variances at each step of the Population Monte Carlo algorithm according to the performances of the scales on the previous iterations. For instance, we decided to select a scale proportionnaly to its non-degeneracy on the previous iterations. (Note the formal similarity of this scheme with Stavropoulos and Titterington's (1999) *smooth bootstrap*, or *adaptive importance sampling*, when the kernel used in their mixture approximation of π is normal. The main difference is that we do not aim at a good approximation of π using standard kernel results like bandwith selection, but rather keep the different scales v_i over the iterations.) Our Population Monte Carlo algorithm thus looks as follows:

Mixture PMC

Step 0 Initialization

(a) for $j \in \{1, \ldots, pm\}$, choice of $(\mu_1)_j^{(0)}$ and $(\mu_2)_j^{(0)}$

(b) for $k \in \{0, \ldots, p-1\}$ and for $j \in \{km+1, \ldots, (k+1)m\}$ generate

$$(\mu_1)_j^{(1)} \sim \mathcal{N}\left((\mu_1)_j^{(0)}, v_{k+1}\right) \quad \text{and} \quad (\mu_2)_j^{(1)} \sim \mathcal{N}\left((\mu_2)_j^{(0)}, v_{k+1}\right)$$

(c) for $k \in \{0, \ldots, p-1\}$ and for $j \in \{km+1, \ldots, (k+1)m\}$ compute the weights

$$\varrho_j \propto \frac{f\left(\mathbf{x} \middle| (\mu_1)_j^{(1)}, (\mu_2)_j^{(1)}\right) \pi\left((\mu_1)_j^{(1)}, (\mu_2)_j^{(1)}\right)}{\tilde{f}\left((\mu_1)_j^{(1)} \middle| (\mu_1)_j^{(0)}, v_{k+1}\right) \tilde{f}\left((\mu_2)_j^{(1)} \middle| (\mu_2)_j^{(0)}, v_{k+1}\right)}$$

(d) resample the $\left((\mu_1)_j^{(1)}, (\mu_2)_j^{(1)}\right)_j$ using the weights ϱ_j,

(e) for $k \in \{1, \ldots, p\}$, calculate $(r_k)^{(1)}$, number of elements generated with variance v_k which have been resampled.

(f) set $(s_1)^{(0)} = 1$, $(s_p)^{(1)} = mp$ and, for $k \in \{1, \ldots, p-1\}$, compute

$$(s_k)^{(1)} = \sum_{w=1}^{k} (r_w)^{(1)}$$

Step i. $(i = 1, \ldots)$

(a) for $k \in \{0, \ldots, p-1\}$ and for $j \in \{(s_k)^{(i-1)}, \ldots, (s_{k+1})^{i-1}\}$ generate

$$(\mu_1)_j^{(i)} \sim \mathcal{N}\left((\mu_1)_j^{(i-1)}, v_{k+1}\right) \quad \text{and} \quad (\mu_2)_j^{(i)} \sim \mathcal{N}\left((\mu_2)_j^{(i-1)}, v_{k+1}\right)$$

(b) compute the weights ($k \in \{0, \ldots, p-1\}$ and $j \in \{(s_k)^{(i-1)}, \ldots, (s_{k+1})^{i-1}\}$)

$$\varrho_j \propto \frac{f\left(\mathbf{x} \middle| (\mu_1)_j^{(i)}, (\mu_2)_j^{(i)}\right) \pi\left((\mu_1)_j^{(i)}, (\mu_2)_j^{(i)}\right)}{\tilde{f}\left((\mu_1)_j^{(i)} \middle| (\mu_1)_j^{(i-1)}, v_{k+1}\right) \tilde{f}\left((\mu_2)_j^{(i)} \middle| (\mu_2)_j^{(i-1)}, v_{k+1}\right)}$$

(c) resample the $\left((\mu_1)_j^{(i)}, (\mu_2)_j^{(i)}\right)_j$ using the weights ϱ_j,

(d) for $k \in \{1, \ldots, p\}$, calculate $(r_k)^{(i)}$ the number of elements generated with variance v_k which have been resampled.

(e) denote $(s_1)^{(i)} = 1$, $(s_p)^{(i)} = mp$ and, for $k \in \{1, \ldots, p-1\}$, compute

$$(s_k)^{(i)} = \sum_{w=1}^{k} (r_w)^{(i)}$$

As mentioned above, the weight associated with each variance v_k is thus proportional to the regeneration (or survival) rate of the corresponding sample. If most μ_j's associated with a given v_k are not resampled, the next step will see less generations using this variance v_k. However, to avoid the complete removal of a given variance v_k, we chose to maintain a minimum number of particles simulated from each variance level, namely 1% of the whole importance sample.

The performances of the above algorithm are illustrated on a simulated dataset of 1000 observations from the distribution $x_i \sim 0.2\mathcal{N}(0,1) + 0.8\mathcal{N}(2,1)$. We also took $\theta = 1.5$ and $\lambda = 0.1$ as hyperparameters of the prior

In the simulation experiment corresponding to this sample, we illustrate that the mixture PMC algorithm produce a non-degenerate particle system, that is, such that their weights are not all equal to either 0 or 1. We also show how the adaptive feature for choosing amongst the v_k's enable us to explore the state space of the unknown means. In this case, $p = 5$ and the five variances are equal to $5, 2, .1, .05$ and $.01$. Moreover, at each step i of the PMC algorithm, we generated 1050 particles.

The two upper graphs of Figure 1 illustrate the degeneracy phenomenon associated with the population Monte Carlo algorithm. It represents the sizes of the samples issued from the different proposals, that is the number of points resulting from the resampling step: the upper left graph exhibits a nearly cyclic behavior for the largest variances v_k, alternating from no particle issued from these proposals to a large number of particles. This behaviour agrees with intuition: proposals that have too large a variance mostly produce particles that are irrelevant for the distribution of interest, but once in a while they happen to generate particles that are close to one of the modes of the distribution of interest. In the later situation, the corresponding particles are associated with large weights ϱ_j and are thus heavily resampled. The upper right graph shows that the other proposals are rather evenly considered along iterations. This is not surprising for the smaller variances, since they modify very little the current particle system, but the cyclic predominance of the three possible variances is quite reassuring about the mixing abilities of the algorithm and thus about its exploration performances.

We can also study the influence of the variation in the proposals on the estimation of the means μ_1 and μ_2, as illustrated by the middle and lower panels of Figure 1. First, when considering the cumulative means of these estimations over iterations, the quantities quickly stabilise. The corresponding variances are not so stable over iterations, but this is to be expected, given the periodic reappearance of subsamples with large variances.

Figure 2 provides an additional insight into the performances of the population Monte Carlo algorithm, by representing a weighted sample of means with dots proportional to the weights. As should be obvious from this graph, there is no overwhelming particle that concentrates most of the weight. On the opposite, the 1050 particles are rather evenly weighted, especially for those close to the posterior modes of the means.

4. Ion channels

4.1. The stylised model

As a realistic example of implementation of the population Monte Carlo scheme, we consider here a formalised version of the ion channel model considered in Hodgson (1999). We refer the reader to this paper, as well as to Ball *et al.* (1999) and Carpenter *et al.* (1999), for a biological motivation of this model, alternative formulations, and additional references.

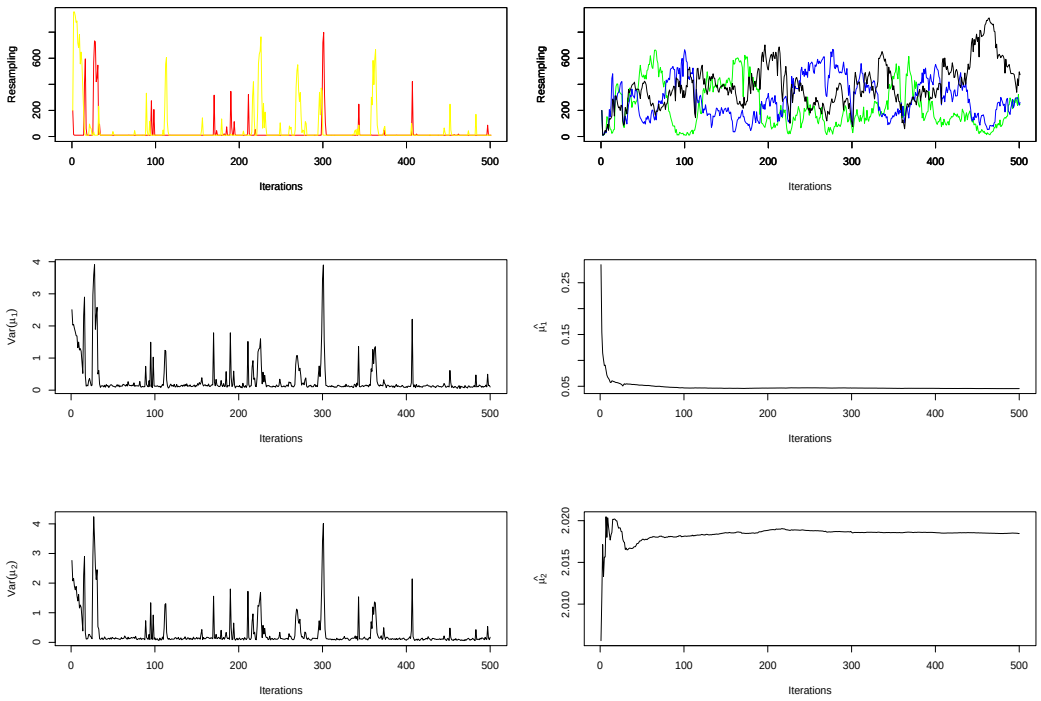

Fig. 1. Performances of the mixture population Monte Carlo algorithm: *(upper left)* Number of resampled particles for the variances $v_1 = 5$ and $v_2 = 2$; *(upper right)* Number of resampled particles for the other variances *(middle left)* Variance of the simulated μ_1's along iterations; *(middle right)* Complete average of the simulated μ_1's over iterations; *(lower left)* Variance of the simulated μ_2's along iterations; *(lower right)* Complete average of the simulated μ_2's over iterations.

Let us insist at this point on the *formalised* aspect on our model, which predominantly serves as a realistic support for the comparison of a population Monte Carlo approach with a more standard MCMC approach in a semi-Markov setting. The finer points of model choice and model comparison for the modelling of ion channel kinetics, while of importance as shown by Ball *et al.* (1999) and Hodgson and Green (1998), are not considered in the present paper. Note also that, while a Bayesian analysis of this model provides a complete inferential perspective, the focus of attention is generally set on the restoration of the true channel current, rather than on the estimation of the parameters of the model.

Consider, thus, observables $\mathbf{y} = (y_t)_{1 \leq t \leq T}$ directed by a hidden Gamma (indicator) process $\mathbf{x} = (x_t)_{1 \leq t \leq T}$ in the following way:

$$y_t | x_t \sim \mathcal{N}(\mu_{x_t}, \sigma^2) \,,$$

while $x_t \in \{0, 1\}$, with durations $d_j \sim \mathcal{G}a(s_i, \lambda_i)$ $(i = 0, 1)$. More exactly, the hidden process $(x_t)_t$ is a (continuous time) Gamma jump process with jump times t_j $(j = 1, 2, \ldots)$ such that

$$d_{j+1} = t_{j+1} - t_j \sim \mathcal{G}a(s_i, \lambda_i)$$

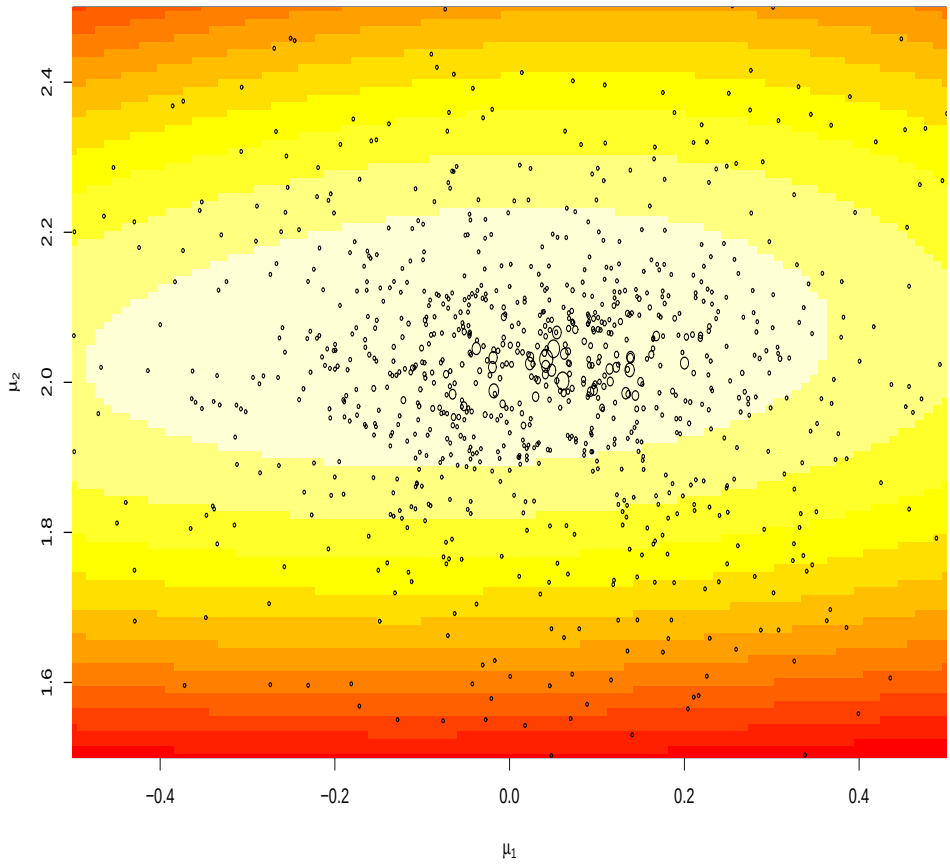

Fig. 2. Representation of the log-posterior distribution via colour levels (red stands for lowest and white for highest) and of a weighted sample of means. (The weights are proportional to the surface of the disc.)

if $x_t = i$ for $t_j \leq t < t_{j+1}$, that is, $\mathbb{E}[d_{j+1}] = s_i/\lambda_i$. Figure 3 provides a simulated sample of size 4000 from this model.

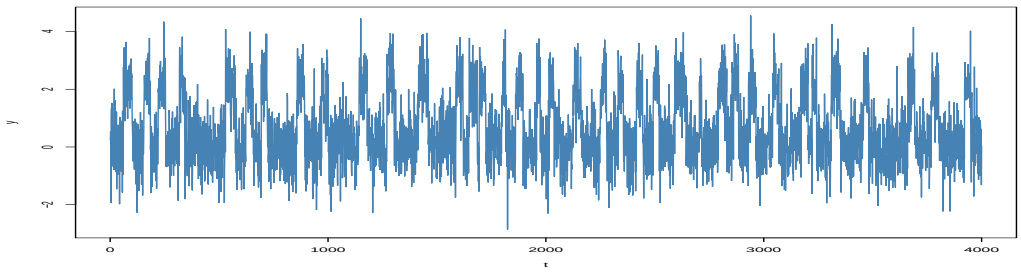

Fig. 3. Simulated sample of size 4000 from the ion channel model

A first modification of the ion channel model is introduced at this level: we assume that the durations d_j, that is, the time intervals in which the process $(x_t)_{1\leq t\leq T}$ remains in a given state, are integer valued, rather than real valued, as in Hodgson (1999). The reasons for this change are that

(a) the true durations of the Gamma process are not identifiable;
(b) this model is a straightforward generalisation of the hidden Markov model where the jumps do occur at integer times (see Ball *et al.*, 1999, or Carpenter *et al.*, 1999). A natural generalisation of the geometric duration of the hidden Markov model is a negative binomial distribution, $\mathcal{N}eg(s,\omega)$, which is very close to a Gamma density $\mathcal{G}a(s+1,-\log(1-\omega))$ (up to a constant) for s small. Indeed, the former is approximately
$$\frac{d^s}{s!}(1-\omega)^d\left(\frac{\omega}{1-\omega}\right)^s$$
while the later is
$$\frac{d^s}{s!}(1-\omega)^d\{-\log(1-\omega)\}^{s+1}$$
(The simulations detailed below were also implemented using a negative binomial modelling, leading to very similar results in the restoration process.)
(c) inference on the d_j's given $(x_t)_{1\leq t\leq T}$ involves an extra level of simulations, even if it can be easily implemented via a slice sampler, as long as we do not consider the possibility of several jumps between two integer observational times. (This later possibility is actually negligible for the datasets we consider.); *and*
(d) the replacement of d_j by its integral part does not strongly modify the likelihood.

In a similar vein, we omit the truncation effect of both the first and the last intervals, given that the influence of this truncation on a long series is bound to be small.

A second modification in our model, when compared with Hodgson (1999), is that we choose a uniform prior for the shape parameters s_0 and s_1 on the finite set $\{1,\ldots,S\}$, rather than an exponential $\mathcal{E}xp(\xi)$ prior on $\mathbb{R}^+$. The reasons for this modification is that

(a) the hidden Markov process has geometric switching times, which correspond to exponential durations. A natural extension is to consider that the durations of the stays

within each state (or *régime*) can be represented as the cumulated duration of s_i exponential stays, with s_i an unknown integer, which exactly corresponds to gamma durations. This representation thus removes the need to call for a level of variable dimension modelling. Carpenter et al. (1999) and Hodgson and Green (1999) use a different approach, based on the replication of the "open" and "closed" sets into several states, to approximate the semi-Markov model.

(b) the following simulations show that the parameters s_0 and s_1 are strongly identified by the observables $(y_t)_{1\le t\le T}$;

(c) the prior information on the parameters s_0 and s_1 is most likely to be sparse and thus a uniform prior is less informative than a Gamma prior when S is large; *and*

(d) the use of a finite support prior allows for the computation of the normalising constant in the posterior conditional distribution of the parameters s_0 and s_1, a feature that is paramount for the implementation of population Monte Carlo.

A third modification, when compared with both Hodgson (1999) and Carpenter *et al.* (1999), is that the observables are assumed to be independent, given the x_t's, rather than distributed from either an AR(15) (Hodgson, 1999) or an ARMA(1,1) (Carpenter *et al.*, 1999) model. This modification somehow weakens the identifiability of both régimes as the data becomes potentially more volatile.

The other parameters of the model are distributed as in Hodgson (1999), using conjugate priors,

$$\begin{aligned} \mu_0, \mu_1 &\sim \mathcal{N}(\theta_0, \tau\sigma^2) \\ \sigma^{-2} &\sim \mathcal{G}(\zeta, \eta) \\ \lambda_0, \lambda_1 &\sim \mathcal{G}(\alpha, \beta) \end{aligned}$$

Figure 4 illustrates the dependences induced by this modelling on a DAG.

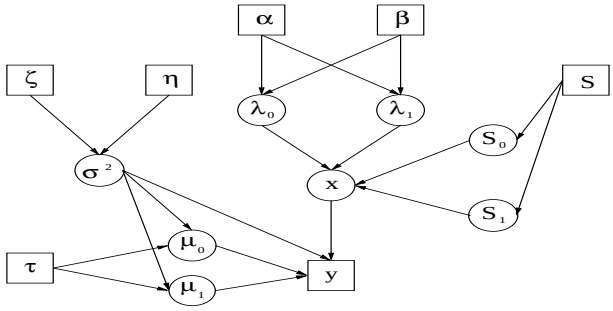

Fig. 4. DAG representation of the probabilistic dependences in the Bayesian ion channel model.

This formalised ion channel model is thus a special case of discrete time hidden semi-Markov model for which there exists no explicit polynomial time formula for the posterior distribution of the hidden process $(x_t)_{1\le t\le T}$, as opposed to the hidden Markov model with the forward–backward formula of Baum and Petrie (1966). From a computational (MCMC) point of view, there is therefore no way of eliminating this hidden process to simulate directly the parameters conditional on the observables $(y_t)_{1\le t\le T}$, as was done in the hidden Markov model by, e.g., Cappé *et al.* (2001). Note also that, as opposed to Hodgson (1999), we use the *saturated* missing data representation of the model via $\mathbf{x}$ to avoid the complication of

using reversible jump techniques for which population Monte Carlo are more difficult to implement.

4.2. Population Monte Carlo for ion channel models

Our choice of proposal function is based on the availability of closed form formulae for the hidden Markov model. We thus create a pseudo hidden Markov model based on the current values of the parameters for the ion channel model, simply by building the Markov transition matrix from the average durations in each state,

$$\mathbb{P} = \begin{pmatrix} 1 - \frac{\lambda_0}{s_0} & \frac{\lambda_0}{s_0} \\ \frac{\lambda_1}{s_1} & 1 - \frac{\lambda_1}{s_1} \end{pmatrix},$$

since, for a hidden Markov model, the average sojourn within one state is exactly the inverse of the transition probability to the other state. We denote by $\pi_H(\mathbf{x}|\mathbf{y}, \omega)$ the full conditional distribution of the hidden Markov chain $\mathbf{x}$ given the observables $\mathbf{y}$ and the parameters

$$\omega = (\mu_0, \mu_1, \sigma, \lambda_0, \lambda_1, s_0, s_1)$$

constructed via the forward–backward formula: see, e.g., Cappé *et al.* (2001) for details. The simulation of the parameters ω proceeds in a natural way by using the full conditional distribution $\pi(\omega|\mathbf{y}, \mathbf{x})$ since it is available.

Note that Carpenter *et al.* (1999) also consider the ion channel model in their particle filter paper, with the differences that they replace the semi-Markov structure with an approximative hidden Markov model with more than 2 states, and that they work in a dynamic setting based on this approximation. The observables $\mathbf{y}$ are also different in that they come from an ARMA(1,1) model with only the location parameter depending on the unknown state. Hodgson and Green (1998) similarly compared several hidden Markov model with duplicated "open" and "closed" states. Ball *et al.* (1999) also rely on a hidden Markov modelling with missing observations.

The subsequent use of importance sampling bypasses the exact simulation of the hidden process $(x_t)_{1 \le t \le T}$ and thus avoids the recourse to variable dimension models and to more sophisticated tools like reversible jump MCMC. This *saturation* of the parameter space by the addition of the whole indicator process $(x_t)_{1 \le t \le T}$ is obviously more costly in terms of storage, but it provides unrestricted moves between configurations of the process $(x_t)_{1 \le t \le T}$. Since we do not need to define the corresponding jump moves, we are thus less likely to encounter the slow convergence problems of Hodgson (1999).

We therefore run population Monte Carlo as follows:

Population Monte Carlo Algorithm

Step 0. Generate ($j = 1, \ldots, J$)

(a) $\omega^{(j)} \sim \pi(\omega)$
(b) $\mathbf{x}_-^{(j)} = (x_t^{(j)})_{1 \le t \le T} \sim \pi_H(\mathbf{x}|\mathbf{y}, \omega^{(j)})$

compute the weights ($j = 1, \ldots, J$)

$$\varrho_j \propto \frac{\pi(\omega^{(j)}, \mathbf{x}_-^{(j)}|\mathbf{y})}{\pi(\omega^{(j)})\, \pi_H(\mathbf{x}_-^{(j)}|\mathbf{y}, \omega^{(j)})}$$

resample the $(\omega^{(j)}, \mathbf{x}_-^{(j)})_j$ using the weights ϱ_j

Step i. ($i = 1, \ldots$) Generate ($j = 1, \ldots, J$)

(a) $\omega^{(j)} \sim \pi(\omega|\mathbf{y}, \mathbf{x}_-^{(j)})$
(b) $\mathbf{x}_+^{(j)} = (x_t^{(j)})_{1 \le t \le T} \sim \pi_H(\mathbf{x}|\mathbf{y}, \omega^{(j)})$

compute the weights ($j = 1, \ldots, J$)

$$\varrho_j \propto \frac{\pi(\omega^{(j)}, \mathbf{x}_+^{(j)}|\mathbf{y})}{\pi(\omega^{(j)}|\mathbf{y}, \mathbf{x}_-^{(j)})\pi_H(\mathbf{x}_+^{(j)}|\mathbf{y}, \omega^{(j)})}$$

resample the $(\omega^{(j)}, \mathbf{x}_+^{(j)})_j$ using the weights ϱ_j, and take $\mathbf{x}_-^{(j)} = \mathbf{x}_+^{(j)}$ ($j = 1, \ldots, J$).

The justification for the weights ϱ_j used in the above algorithm is that conditional on the $\mathbf{x}_-^{(j)}$'s, $\omega^{(j)}$ is simulated from $\pi(\omega|\mathbf{y}, \mathbf{x}_-^{(j)})$ and, conditional on $\omega^{(j)}$, $\mathbf{x}_+^{(j)}$ is simulated from $\pi_H(\mathbf{x}|\mathbf{y}, \omega^{(j)})$. The normalising factor of the ϱ_j's converges to the correct constant by the law of large numbers.

4.3. Normalising constants

A major point in this development covers the normalising constants in the various terms: $\pi(\omega|\mathbf{y}, \mathbf{x})$ is available in closed form (see below in Section 4.4), including its normalising constant, due to the conjugacy of the distributions on $\mu_0, \mu_1, \sigma, \lambda_0, \lambda_1$ and the finiteness of the support of s_0, s_1. The conditional distribution $\pi_H(\mathbf{x}|\mathbf{y}, \omega)$ is also available with its normalising constant, by virtue of the forward–backward formula. The only difficulty in the ratio

$$\frac{\pi(\omega, \mathbf{x}|\mathbf{y})}{\pi(\omega|\mathbf{y}, \mathbf{x})\pi_H(\mathbf{x}|\mathbf{y}, \omega)}$$

lies within the numerator $\pi(\omega, \mathbf{x}|\mathbf{y})$ whose normalised version is unknown. We therefore use instead the proportional term

$$\pi(\omega, \mathbf{x}|\mathbf{y}) \propto \pi(\omega)\, f(\mathbf{y}|\mathbf{x}, \omega)\, f(\mathbf{x}|\omega)\,. \tag{3}$$

and normalise the ϱ_j's by their sum. The foremost feature of this reweighting is that the normalising constant missing in (3) only depends on the observables $\mathbf{y}$ and is therefore truly a *constant*, that is, does not depend on the previous value of the particle $\mathbf{x}_-^{(j)}$. This scheme crucially relies on (i) the particles encompassing both the parameters ω and the latent data $\mathbf{x}$, and (ii) the distribution $\pi(\omega, \mathbf{x}|\mathbf{y})$ being available in closed form.

More generally, attention must be paid to the selection of the proposal densities $q(x|y)$ so that the normalising constants in these densities that depend on y must be available in closed form.

4.4. Simulation details

Figure 5 illustrates the performances of population Monte Carlo by representing the graph of the dataset against the fitted average

$$\sum_{j=1}^{J} \varrho_j \mu_{x_t^{(j)}}$$

for each observation y_t. As obvious from the picture, the fit is quite good.

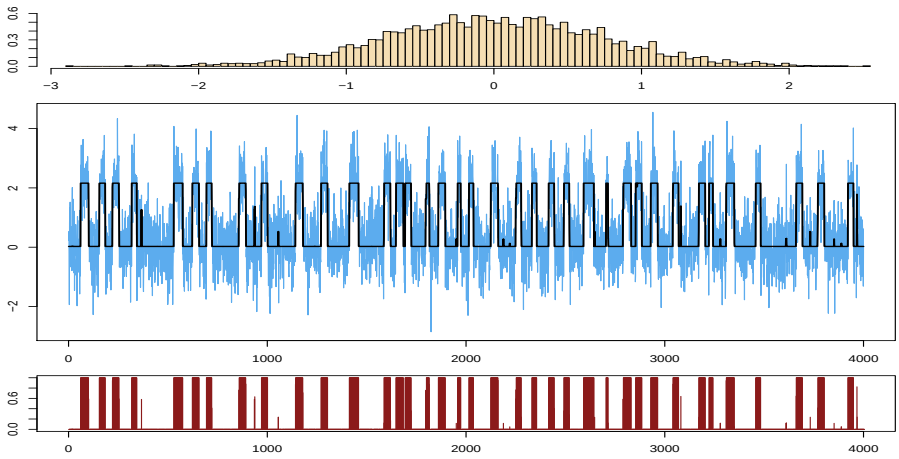

Fig. 5. (top) Histograms of residuals after fit by averaged μ_{x_t}; (middle) Simulated sample of size 4000 against fitted averaged μ_{x_t}; (bottom) Probability of allocation to first state for each observation

The unobserved Gamma process is distributed as

$$\prod_{m=1}^{M} \frac{(t_{m+1}-t_m)^{s_m-1}\,\lambda_m^{s_m}\,e^{-\lambda_m(t_{m+1}-t_m)}}{\Gamma(s_m)} = \frac{\lambda_0^{n_0 s_0}\,e^{-\lambda_0 v_0}\,\Delta_0^{s_0-1}}{\Gamma(s_0)^{n_0}} \frac{\lambda_1^{n_1 s_1}\,e^{-\lambda_1 v_1}\,\Delta_1^{s_1-1}}{\Gamma(s_1)^{n_1}},$$

with obvious notations: M is the number of changes, the t_m's are the successive times when the gamma process changes state, the s_m's, λ_m's are the corresponding sequences of s_0, s_1 and λ_0, λ_1, n_i is the number of visits to state i, Δ_i is the product of the sojourn durations in state i [corresponding to the geometric mean], v_i the total sojourn duration in state i [corresponding to the arithmetic mean]. (This is based on the assumption of no censoring, made in Section 4, namely that $t_1 = 1$ and $t_{M+1} = T + 1$.)

The posterior distributions on the μ_i's and σ^{-2} [conditional on the hidden process] are thus the standard Normal-Gamma conjugate priors while

$$\begin{aligned} \lambda_i|s_i,\mathbf{x} &\sim \mathcal{G}a(\alpha + n_i s_i, \beta + v_i) \\ s_i|\mathbf{x} &\sim \pi(s_i|\mathbf{x}) \propto \left[\frac{\Delta_i}{(\beta+v_i)^{n_i}}\right]^{s_i} \frac{\Gamma(n_i s_i + \alpha)}{\Gamma(s_i)^{n_i}}\, \mathbb{I}_{\{1,2,\ldots,S\}}(s_i) \end{aligned}$$

Therefore, except for the s_i's, the posterior distributions on the parameters of the model are the same as in Hodgson (1999).

The distribution on the s_i's is highly variable, in that the product

$$\left[\frac{\Delta_i}{(\beta+v_i)^{n_i}}\right]^{s_i} \frac{\Gamma(n_i s_i+\alpha)}{\Gamma(s_i)^{n_i}} \tag{4}$$

often leads to a highly asymmetric distribution, which puts most of the weight on the minimum value of s. Indeed, when the geometric and arithmetic means, $\Delta_i^{1/n}$ and v_i/n, are similar, a simple Stirling approximation to the Gamma function leads to (4) being equivalent to $\sqrt{n}/\sqrt{s}^n$.

Figure 6 gives the histograms of the posterior distributions of the various parameters of ω without reweighting by the importance sampling weights ϱ_j. As seen from this graph, the histograms in μ_i and σ are well concentrated, while the histogram in λ_1 exhibits two modes which correspond to the two modes of the histogram of s_1 and indicate that the parameter (λ_i, s_i) is not well identified. This is to be expected, given that we only observe a few realisations of the underlying gamma distribution, and this with added noise since the durations are not directly observed. However, the histograms of the average durations s_i/λ_i do not exhibit such multimodality and are well-concentrated around the values of interest. Note also that the concentration of the distributions of s_0 and s_1 around the smaller integers should be connected with Hodgson and Green's (1999) findings that, within their model, the number of duplicated copies of the "open" and "closed" states is precisely 2.

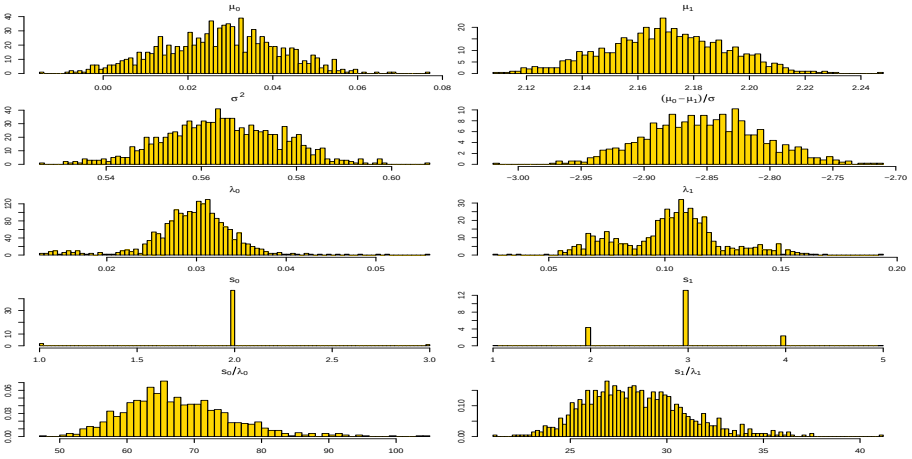

Fig. 6. Histograms of the samples produced by population Monte Carlo, before resampling

4.5. Degeneracy

As noted above, population Monte Carlo is simply an importance sampling algorithm when implemented once, that is, for a single collection of J particles $(\omega^{(j)}, \mathbf{x}^{(j)})$. As such, it provides an approximation device for the target distribution but it is also well-known that a poor choice of the importance sampling distribution can jeopardise the interest of the approximation, as for instance when the weights ϱ_j have infinite variance.

An incentive of using population Monte Carlo in a static setting is thus to overcome a poor choice of the importance function by recycling the best particles and discarding the

worst ones. This point of view makes population Monte Carlo appear as a primitive kind of *adaptive algorithm*, in that the support of the importance function is adapted to the performance of the previous importance sampler.

The difficulty with this approach is in determining the long-term behaviour of the algorithm and, correlatively, the *stopping rule* that decides that nothing is gained in running the algorithm any longer. For instance, it often happens that only a few particles are kept after the resampling step of the algorithm, because only a few weights ϱ_j are different from 0. Figure 7 gives for instance the sequence of the number of particles that matter at each iteration, out of 1000 original particles: the percentage of relevant particles is thus less than 10% on average and in fact much closer to 5%. In addition, there is no clearcut stabilisation in either the number of relevant particles or the variance of the corresponding weights, the later being far from exhibiting a stabilisation as the number of iterations increases. Some more rudimentary signals can be considered though, like the stabilisation of the fit in Figure 8. While the averages for 1 and 2 iterations are quite unstable for most observations, the two states are much more clearly identified for 5 and 10 iterations, and hardly change over subsequent iterations.

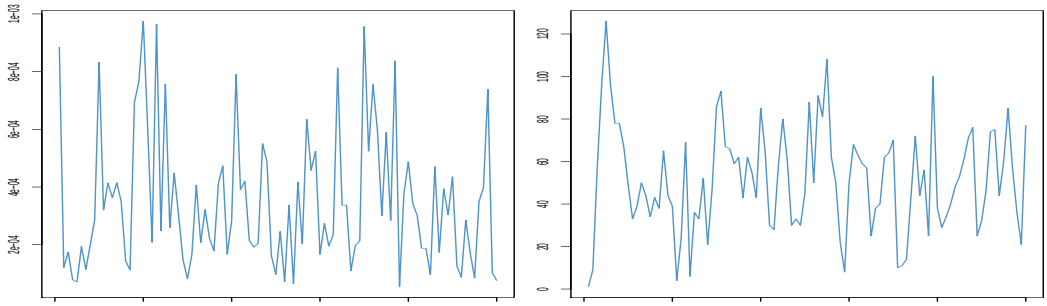

Fig. 7. (left) Variance of the weights ϱ_j along 100 iterations, and (right) Number of particles with descendants along 100 iterations, for a sample of 4000 observations and 1000 particles.

A related phenomenon pertains to the degeneracy of ancestors observed in the iterations of our algorithm: as the number of steps increases, the number of particles from the first generation used to generate particles from the last generation diminishes and, after a few dozen iterations, reduces to a single ancestor. This is for instance what occurs in Figure 9 where, after only two backward iterations, there is a single ancestor to the whole system of particles. (Note also the iterations where all particles originate from a single particle.) This phenomenon appears in every setting and, while it cannot be avoided, since some particles are bound to vanish at each iteration even using the systematic sampling of Carpenter *et al.* (1999) the surprising factor is the speed with which the number of ancestors decreases.

4.6. A comparison with Hastings–Metropolis algorithms

As mentioned above, the proposal distribution associated with the pseudo hidden Markov model could be as well used as a proposal distribution in a Metropolis–Hastings algorithm of the following form:

MCMC Algorithm

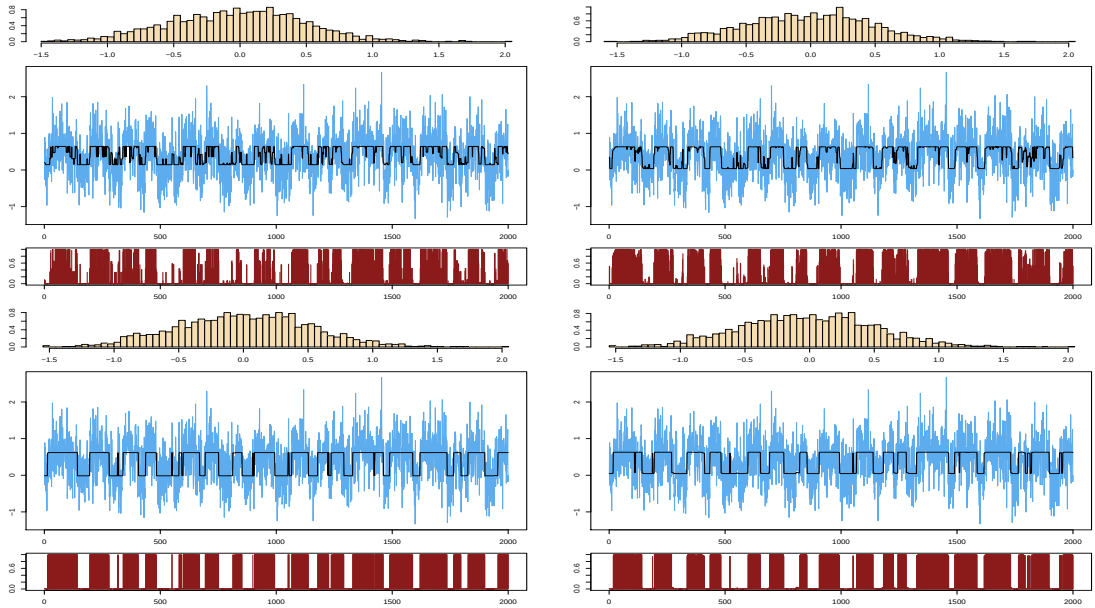

Fig. 8. Successive fits of population Monte Carlo iterated by the weight-resample algorithm for 2000 observations and 2000 particles, for (clockwise starting from top left) 1, 2, 5 and 10 iterations. (See the caption of Fig. 5 for a description of the displayed quantities.)

Step i ($i = 1, \ldots, J$)

(a) Generate $\omega^{(i)} \sim \pi(\omega|\mathbf{y}, \mathbf{x}^{(i-1)})$

(b) Generate $\mathbf{x}^\star \sim \pi_H(\mathbf{x}|\mathbf{y}, \omega^{(i)})$, $u \sim \mathcal{U}([0,1])$ and take

$$\mathbf{x}^{(i)} = \begin{cases} \mathbf{x}^\star & \text{if } u \leq \dfrac{\pi(\mathbf{x}^\star|\omega^{(i)}\mathbf{y})}{\pi_H(\mathbf{x}^\star|\mathbf{y},\omega^{(i)})} \Big/ \dfrac{\pi(\mathbf{x}^{(i-1)}|\omega^{(i)}\mathbf{y})}{\pi_H(\mathbf{x}^{(i-1)}|\mathbf{y},\omega^{(i)})}, \\ \mathbf{x}^{(i-1)} & \text{otherwise} \end{cases}$$

The performances of this alternative algorithm are, however, quite poor. Even with a well-separated dataset like the simulated dataset represented in Figure 3, the algorithm requires a very careful preliminary tuning not to degenerate into a single state output. More precisely, the following occurs: when started at random, the algorithm converges very quickly to a configuration where both means μ_0 and μ_1 of the ion channel model are very close to one another (and to the overall mean of the sample), with, correlatively, a large variance σ^2 and very short durations within each state. To overcome this degenerescence of the sample, we had paradoxically to ressort to a *sequential* implementation as follows: noticing that the degenerescence was only occurring with large sample sizes, we start the MCMC algorithm on the first 100 observations $y_{1:100}$ and, once a stable configuration has been achieved, we gradually increase the number of observations taken into account [by a factor of $\min(s_0/\lambda_0, s_1/\lambda_1)$] till the whole sample is included. The results provided in Figures 10–12 were obtained following this scheme.

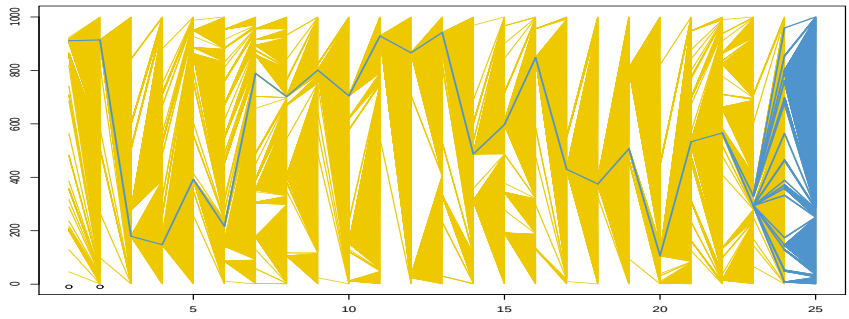

Fig. 9. Representation of the sequence of descendents (yellow) and ancestors (blue) along iterations through bars linking a given ancestor and all its descendents (yellow) or a given particle and its ancestor (blue). In the simulation corresponding to this graph, there were 4000 observations and 1000 particles.

For conciseness' sake, we did not reproduce the history of the allocations $\mathbf{x}$ over the iterations. The corresponding graph shows a very stable history with hardly any change, except on a few boundaries. Note the connected strong stability in the number of switches in Figure 11 *(right)*. [The cumulated means on the rhs of Figure 11 indicate that more iterations of the MCMC sampler were necessary but our purpose here was simply to illustrate the proper behaviour of this sampler, provided the initialisation was adequate.]

Attempts with very mixed datasets as the one used in Figure 8 were much less successful since, even with a careful tuning of the starting values (we even tried starting with the known values of the parameters), we could not avoid the degenerescence to a single state. The problem with the Metropolis–Hastings algorithm in this case is clearly a strong dependence on the starting value, i.e., a poor mixing ability. This is further demonstrated by the following experiment: when starting the above sampler from 1000 particles obtained by running population Monte Carlo 20 times, the sampler always produced a satisfactory solution with two clearcut states and no degeneracy. Figure 12 compares the distributions of the particle and the MCMC samples via a qq-plot and shows there is very little difference between both. The same behaviour is shown by a comparison of the allocations (not represented here). This indicates that the MCMC algorithm does not lead to a better exploration of the parameter space.

For a fairly mixed dataset of 2000 observations corresponding to Figure 8, while the MCMC algorithm initialised at random could not avoid degeneracy, a preliminary run of population Monte Carlo produced stable allocations to two states, as shown in Figure 13 by the fit for both population Monte Carlo and MCMC samples: they are indistinguishable, even though the qq-plots in Figure 14 indicate different tail behaviours.

This is not to say that an MCMC algorithm cannot work in this setting, since Hodgson (1999) demonstrated the contrary, but this shows that *global* updating schemes, that is, proposals that update the whole missing data $\mathbf{x}$ at once, are difficult to come with, and that one has to instead rely on more *local* moves as those proposed by Hodgson (1999). A similar conclusion was drawn by Billio *et al.* (1999) in the setup of switching ARMA models. (See also Kim, Shephard and Chib, 1998.)

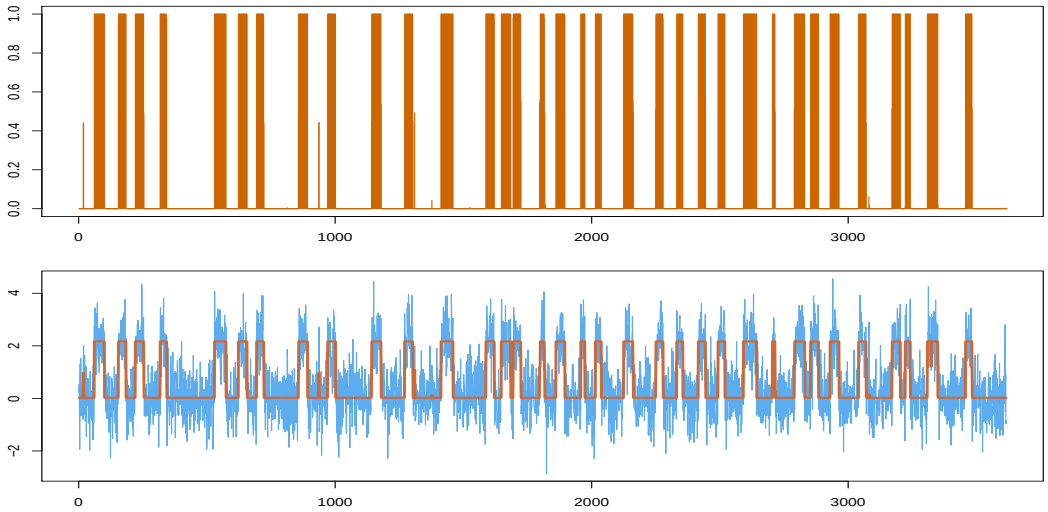

Fig. 10. Representation of a dataset of 3610 simulated values, along with the average fit *(bottom)*, and average probabilities of allocation to the upper state *(top)*. This fit was obtained using a sequential tuning scheme and 5000 MCMC iterations in the final run.

5. Conclusion

The above developments have confirmed Chopin's (2002) realisation that population Monte Carlo is also useful tools in static—as opposed to sequential, rather than dynamic—setups. Quite obviously, the specific Monte Carlo scheme we built for the ion channel model can be used in a sequential setting in a very similar way. The comparison with the equivalent MCMC algorithm is also very instructive in that it shows the superior robustness of population Monte Carlo to a possibly poor choice of the proposal distribution.

There still are issues to explore about population Monte Carlo scheme. In particular, a more detailed assessment of the dynamic feature is in order, to decide whether or not it is a real asset. It is possible that there is an equivalent to the "cutoff phenomenon": after a given t_0, the distribution of $\mathbf{x}^{(t)}$ may be the same for all $t \geq t_0$. Further comparisons with full Metropolis–Hastings moves based on similar proposals would also be of interest, to study which scheme brings the most information about the distribution of interest.

An extension not explored in this paper is that the particle system can be started with a few particles that explore the parameter space and, once the mixing is well-established, the particles can be duplicated further to increase the precision of the approximation to the distribution of interest. This is a straightforward extension in terms of programming, but the selection of the duplication rate and schedule is more delicate.

Acknowledgements

The authors are grateful to the Friday lunch discussion group at Institut Henri Poincaré for their input. Comments from Nicolas Chopin and Gareth Roberts were also particularly helpful.

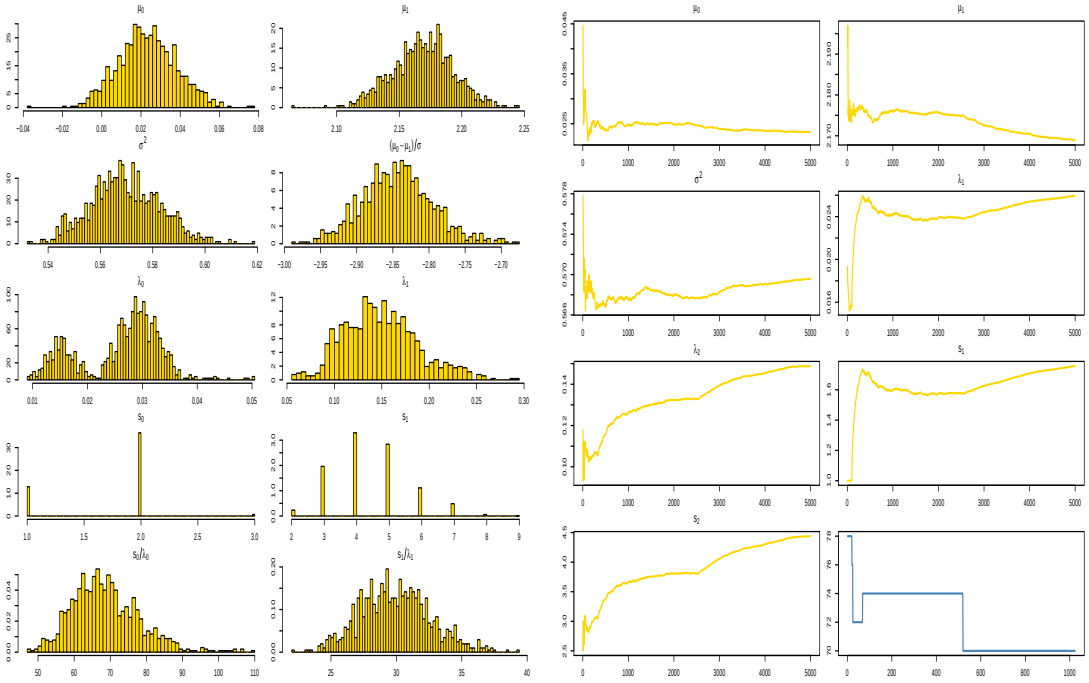

Fig. 11. Details of the MCMC sample for the dataset of Figure 10: *(left)* histograms of the components of the MCMC sample and *(right)* cumulative averages for the parameters of the models and evolution of the number of switches.

References

Andrieu, C. and Robert C.P. (2001) Controlled Markov chain Monte Carlo methods for optimal sampling. *Cahiers du Ceremade* 0125, Université Paris Dauphine.

Ball, F.G., Cai, Y., Kadane, J.B. and O'Hagan, A. (1999) Bayesian inference for ion channel gating mechanisms directly from single channel recordings, using Markov chain Monte Carlo. *Proc. Royal Society London* A **455**, 2879–2932.

Baum, L.E. and Petrie, T. (1966) Statistical inference for probabilistic functions of finite state Markov chains. *Annals Math. Statist.* **37**, 1554–1563.

Berzuini, C., Best, N., Gilks, N.G., and Larizza, C. (1997) Dynamic conditional independence models and Markov Chain Monte Carlo methods. *JASA* **92**, 590–599.

Billio, M., Monfort, A. and Robert, C.P. (1999) Bayesian estimation of switching ARMA Models. *J. Econometrics* **93**, 229–255.

Cappé, O., Robert, C.P., and Rydén, T. (2001) Discrete time and continuous time jump processes with applications to hidden Markov models. Tech. report, CREST, INSEE, Paris.

Carpenter, J., Clifford, P., and Fernhead, P. (1999) Building robust simulation based filters for evolving data sets. Technical report, Department of Statistics, Oxford University.

Celeux, G., Hurn, M. and Robert, C.P. (2000) Computational and inferential difficulties with mixture posterior distributions. *J. Amer. Statist. Assoc.* **95**, 957–979.

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

Doucet, A., deFreitas, N., and Gordon, N. (2001) *Sequential MCMC in Practice*. Springer-Verlag, New York.

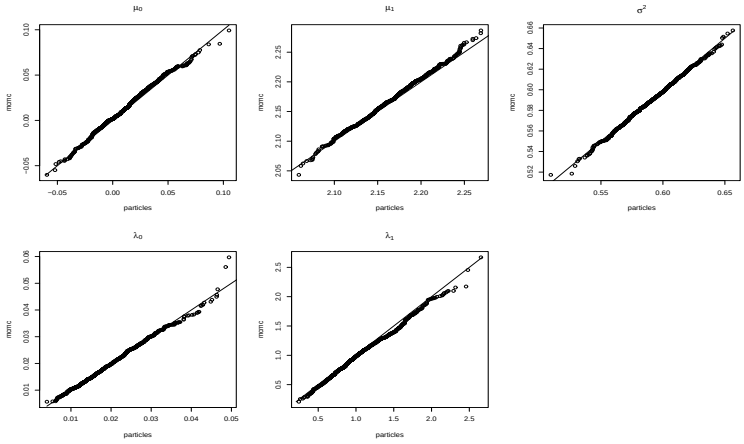

Fig. 12. QQ-plot comparing the distribution of the particle system with the distribution of the MCMC sample obtained after 5000 iterations started at the particles.

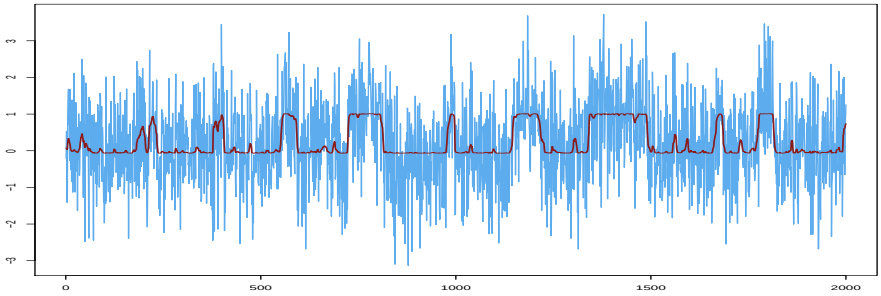

Fig. 13. Simulated dataset of 2000 points with superimposed fits by population Monte Carlo and the MCMC samples, the later being initialised by population Monte Carlo. Both fits are indistinguishable.

Gordon, N., Salmond, D., and Smith, A.F.M. (1993) Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proceedings F*, **140**, 107–113.

Green, P.J. (1995) Reversible jump MCMC computation and Bayesian model determination. *Biometrika* **82**(4), 711–732.

Haario, H., Saksman, E. and Tamminen, J. (1999) Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics* **14** 375–395.

Haario, H., Saksman, E. and Tamminen, J. (2001) An adaptive Metropolis algorithm. *Bernoulli* **7**.

Hodgson, M.E.A. (1999) A Bayesian restoration of a ion channel signal. *JRSS B* **61**(1), 95–114.

Hodgson, M.E.A. and Green, P.G. (1999) Bayesian choice among Markov models of ion channels using Markov chain Monte Carlo. *Proc. Royal Society London* A **455**, 3425–3448.

Kim, S., Shephard, N. and Chib, S. (1998) Stochastic volatility: Likelihood inference and comparison with ARCH models. *Rev. Econom. Stud.* **65**, 361–393.

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

Liu, J.S., Liang, F. and Wong, W.H. (2001) A theory for dynamic weighting in Monte Carlo computation. *JASA* **96**, 561–573.

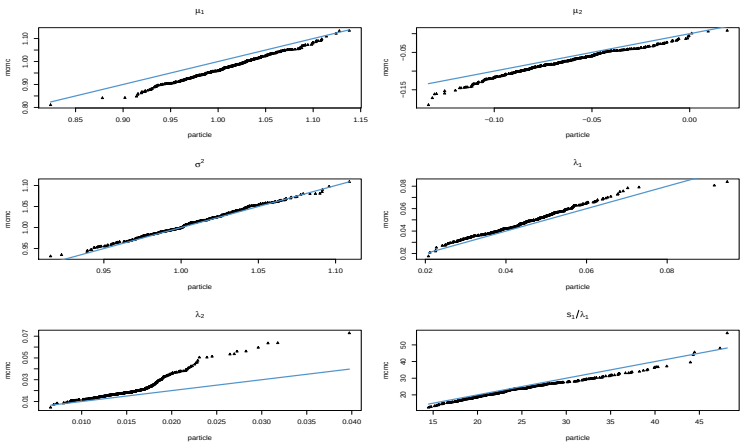

Fig. 14. QQ-plot comparing the distribution of the particle system with the distribution of the MCMC sample obtained after 5000 iterations started at one particle.

MacEachern, S.N. and Peruggia, M. (2000) Importance link function estimation for Markov Chain Monte Carlo methods. *J. Comput. Graph. Statist.* **9**, 99–121.

Oh, M.S. and Berger, J.O. (1992) Adaptive importance sampling in Monte Carlo integration. *J. Statist. Comput. Simul.* **41**, 143–168.

Oh, M.S. and Berger, J.O. (1993) Integration of multimodal functions by Monte Carlo importance sampling. *JASA* **88**, 450–456.

Pitt, M.K. and Shephard, N. (1999) Filtering via simulation: auxiliary particle filters. *JASA* **94**, 1403–1412, 1403–1412

Robert, C.P. and Casella, G. (1999) *Monte Carlo Statistical Methods.* Springer-Verlag, New York.

Rubin, D.B. (1987) *Multiple Imputation for Nonresponse in Surveys.* John Wiley, New York