PERFECT POSTERIOR SIMULATION FOR MIXTURE AND HIDDEN MARKOV MODELS

KASPER K. BERTHELSEN, LAIRD A. BREYER AND GARETH O. ROBERTS

Abstract

In this paper we present an application of read-once coupling from the past to problems in Bayesian inference for latent statistical models. We describe a method of simulating perfectly from the posterior distribution of the unknown mixture weights in a mixture model. Our method is extended to a more general mixture problem where unknown parameters exist for the mixture components, and to a hidden Markov model.

1. *Introduction*

Following the seminal work in [**9**] on perfect simulation, considerable effort has gone into developing the general methodology, and extending the range of distributions for which these methods apply. In this paper we describe the implementation of a perfect simulation algorithm for generating samples from the posterior distribution in three different Bayesian latent variable models. Our methods are firstly applied to the posterior distribution of the unknown mixture weights in a Bayesian analysis of a mixture of known distributions. Secondly, we extend the method to the case where, in addition to unknown weights, the mixture component distributions are parameterised by unknown parameters, and thirdly, we consider the Bayesian analysis of a hidden Markov model.

Our approach will be to use and extend a collection of recently developed techniques for perfect simulation. Central to all of our methodology will be the use of read-once coupling from the past (roCFTP) algorithm for perfect simulation, see [**11**]. At the core of this algorithm is the construction of uniformly ergodic blocks of Markov chain update rules. The main challenge is to construct these blocks of updates so that they have a significant coalescence probability, ie. the probability that a block maps the entire state space into just one state is non-negligible.

The main example in this paper is the mixture problem where the mixture components are known but the mixture weights are unknown. For this example the primary updating construction makes use of the duality principle, see [**7**], by augmenting allocation variables. This allows the simultaneous update of all potential allocations within a rectangular regions in the allocation space and, crucially, to determine a rectangular region containing the updated allocations. These update functions will be augmented with *catalytic* updates.

The catalytic update was first considered by [**3**] and [**2**]. As an example [**3**] considers how catalytic updates may be used for perfect simulation of the posterior

Received January 9, 2007.

mixture weights. This is essentially the perfect simulation algorithm we consider and extend in Section 5.1. Although thus extension is in principle rather general, the methodology becomes computationally infeasible for even moderately sized problems. The advantage of the catalytic updates in this context is that they significantly help in detecting if a block of updates is coalescent. On the downside the first catalytic update in each block of updates is computationally expensive. Using the idea of one-shot coupling introduced by [**10**], we extend the approach by [**1**] of tracking the rectangular bounding set until it is sufficiently small to make the first catalytic update "cheap enough".

In the case where the mixture consists of r densities $p(\cdot|\theta_1), \ldots, p(\cdot|\theta_r)$ where $\theta_1, \ldots, \theta_r$ are unknown parameters there is an identifiability problem. This problems stems from the fact that the mixture in invariant under permutation of the unknown parameters. As a result the posterior distribution has $k!$ identical modes. This typically leads to problems when attempting to sample the posterior by conventional MCMC methods as only one of these modes is likely to be explored. Attempts to eliminate the unidentifiability by e.g. imposing restrictions on the parameter space may introduce new problems, for more details see [**5**]. Some of these problems could be solved if perfect simulation was available.

Our extension of perfect simulation to the case of unknown parameters in the specification of the component densities is capable of producing exact sample — although this method is currently only computationally feasible for small data sets. This is not surprising given the mixing problems experienced using conventional MCMC for this problem. the same.

As a further extension of our methodology, we consider the case where the allocation variables are identically distributed we assume they are distributed as a Markov chain obtaining a hidden Markov model. Posterior simulation is done using the same basic approach as for the mixture problem. The main difference being that applying catalytic update appears infeasible.

There exist several works on perfect simulation in connection with mixture models. In [**8**] the basic update function is used in a CFTP type algorithm for perfect simulation. The detection of coalescence is done by tracking a bounding set similar to what we do. Unlike our method this method seems limited to mixtures of at most three components. [**4**] present a perfect slice sampler which turns out to be inefficient for moderate data sizes. In addition a perfect simulation algorithm is considered where each update consists in doing a "simple" catalytic update. For this algorithm coalescence is achieved when a single catalytic update couples the entire state space. The waiting time for this to happen may be very long.

The remainder of the paper is organised as follows. In section 2 we specify the basic mixture problem and review the roCFTP algorithm. In section 3 we construct an update function that has the posterior of interest as it equilibrium distribution. Section 4 concerns how to construct bounds for the image of this update function. These bounds are used in Section 5 to produce perfect samples of the posterior weight distribution. In Section 5 we also review the catalytic update and use it for an improved perfect simulation algorithm. The section is concluded with a simulation study comparing different schemes for constructing blocks of updates. In Section 6 we extend our method to the situation where the component densities are Gaussian with unknown means. In Section 7 we describe an algorithm for perfect simulation from the posterior transition probabilities in a two state hidden Markov model.

2. *Problem specification*

2.1. *The posterior weight distribution*

Assume that the data points $\eta_1, \ldots, \eta_n$ are an independent sample from an r component the mixture

$$\sum_{k=1}^{r} m_k p_k(\cdot),$$

where the component densities $p_1, \ldots, p_r$ are assumed to be known and the mixture weights $m_1, \ldots, m_r$ are restricted to $m_k \geqslant 0$ and $\sum_{k=1}^{r} m_k = 1$.

Introducing a uniform Bayesian prior distribution (Dirichlet $\mathcal{D}(1, 1, \ldots, 1)$) for the unknown weights $m = (m_1, \ldots, m_r)$ on the simplex

$$S = \Big\{(m_1, \ldots, m_r) : m_k \geqslant 0 \text{ for all } k \text{ and } \sum_{k=1}^{r} m_k = 1\Big\},$$

we obtain the posterior distribution

$$\pi(m_1, \ldots, m_r|\eta) = \prod_{i=1}^{n}\Big[\sum_{k=1}^{r} m_k p_k(\eta_i)\Big]. \tag{1}$$

As this is typically intractible, it is usual to resort to Monte Carlo methods for exploring this distribution. Mixing of MCMC algorithms for this problem is often problematical. Our goal in perfect simulation is to avoid the burn-in problem associated with ordinary MCMC.

2.2. *Wilson's read-once CFTP algorithm*

The perfect simulation algorithms considered in this paper are all examples of the roCFTP algorithm introduced by [**11**]. For completeness we briefly review the roCFTP algorithm below and set some of the related notation used throughout this paper.

Assume that we want to sample from a distribution Π on a state space Ω and that we know how to generate a sequence of independent realisations of a random update function $C : \Omega \to \Omega$ with the following properties: 1) It preserves stationarity, i.e. $\int_\Omega \mathbb{P}[C(x) \in A]\Pi(dx) = \Pi(A)$ for all $A \subseteq \Omega$ and 2) has a positive probability of being coalescent, ie. C maps the entire state space into a single state. A realisation of a random update function is denoted an update function. In the following $\#W$ denotes the cardinality of the set W.

Under these assumptions we can generate a perfect sample as follows. Generate a sequence $C_1, C_2, \ldots$ of independent realisations of the random update function C. Furthermore, let T_i denote the index of the ith coalescent update function, ie. $C_{T_i}(x)$ is coalescent and hence its image does not depend on x. Then $x_1 = C_{T_2-1} \circ \cdots \circ C_{T_1}(x)$ does not depend on $x \in \Omega$, and more importantly x_1 is a sample from the target distribution. In general, if $x_i = C_{T_{i+1}-1} \circ \cdots \circ C_{T_i}(x)$ then $x_1, x_2, \ldots$ are independent and identically distributed according to the target distribution. For a proof based on the ideas of Propp and Wilson's coupling from the past algorithm, see [**11**] and for a more algebraic proof see [**3**].

In most cases of interest the random update function C is a compound made up of several "basic" random update functions each fulfilling 1) but not necessarily

2). So $C_i = F_{i,K} \circ \cdots \circ F_{i,2} \circ F_{i,1}$ where $F_{i,1}, \ldots, F_{i,K}$ are random update function preserving stationarity. For example, F could represent a Gibbs update of the type described in Section 3.

A further complication is that typically there is no feasible way of telling if a given realisation C_i is coalescent or not. In some situations it may be possible to find a criteria which implies that C_i is coalescent but not the other way around. If the criteria is fulfilled we can declare C_i coalescent. Given such a criteria we redefine T_i to be the index of the ith update function for which the criteria holds. Letting x_i be defined as before we obtain a subsample of the original perfect sample which, crucially, is still a perfect sample from the target distribution, see [**3**]. So it would seem that the better the criteria is in detecting coalescent update functions the more effective our perfect sampler is. The downside, as we shall see later, an effective criteria can come at a high computational cost.

If Ω is of sufficiently low cardinality it may be feasible to check if each $x \in \Omega$ result in the same value of $C_i(x)$. In this case all coalescent update functions will be detected. Instead assume that given update function F_t and $W \subseteq \Omega$ we we can construct a bounding set W' so that $F_t(W) \subseteq W'$, where $F_t(W) = \{F_t(x) : x \in W\}$. For each C_i we initially set $W = \Omega$ and if at any point $\#W' = 1$ we declare C_i coalescent and non-coalescent otherwise. The roCFTP perfect simulation algorithm using bounding sets is illustrated in Figure 1 and can be summarised in pseudo code as follows.

1. Choose arbitrary x in Ω and set $s := 0$
2. For $i = 1, 2, 3, \ldots$
3. Set $W := \Omega$ and $x_{\text{old}} := x$
4. For $t = 1, \ldots, K$
5. Generate F_t and set $x := F_t(x)$
6. Determine $W' \subseteq \Omega$ so that $F_t(W) \subseteq W'$
7. Set $W := W'$
8. If $\#W = 1$ set $x_s := x_{\text{old}}$ and $s := s + 1$

Notice that the x_0 generated by the algorithm is not part of the sample from Π, only $x_1, x_2, \ldots$ are. Notice further that each update function F_t is only used once unlike conventional CFTP algorithms, hence the name read once CFTP.

In practise constructing F_t so that is preserves stationarity is in general easy but not quite enough. If Ω is an unbounded state space we typically want F_1 to map Ω into a bounded region with positive probability. In this paper this is only an issue when the component densities involve unknown parameters. A further challenge is to construct F_t in such a way that detecting coalescence becomes feasible.

In Section 3 we consider how to construct a random map F that has (1) as its equilibrium distribution. In Section 4 we show how to construct bounding sets making perfect simulation possible. In Section 5 we consider three different choices of the random update function C. Two of these involve using catalytic updates. Using catalytic updates proves to be computationally expensive but if successfully applied they make it much easier to tell if a block is coalescent. The third choice makes use of the fact that the distribution of F_t is allowed to depend on W as long as F_t conditional on W still preserves stationarity, i.e. $\int_\Omega \mathbb{P}[F_t(x) \in A|W]\Pi(dx) = \Pi(A)$ for all $A, W \subseteq \Omega$.

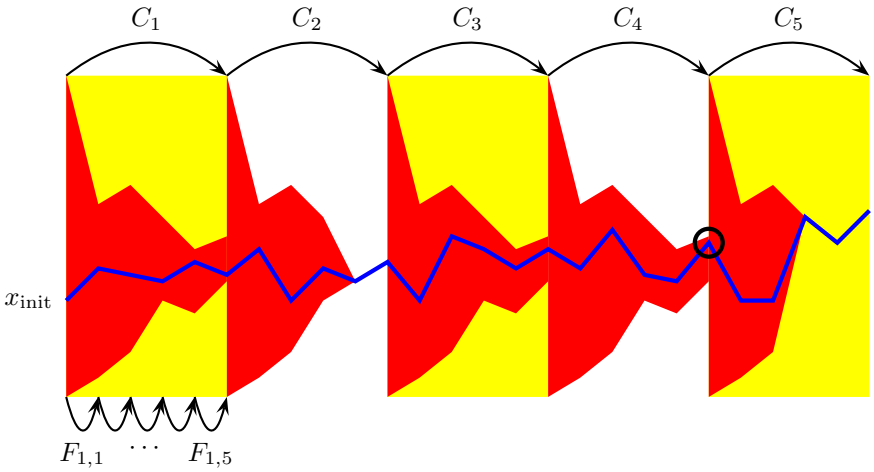

Figure 1: Illustration of the roCFTP algorithm. The red region corresponds to the bounding sets. The blue line corresponds to the target process started in the arbitrarily chosen state x_{init}. In this example the compound update functions C_2 and C_5 are declared coalescent and accordingly $T_1 = 2$, $T_2 = 5$ and $x_1 = C_4 \circ C_3 \circ C_2(x)$ (black circle) is a sample from Π.

3. *Update function: Gibbs sampler updates*

It is well known that the Gibbs sampler can be used to approximately simulate from the posterior distribution (1) by means of the data augmentation methodology, see [**7**]. More precisely, to simulate from π, we define auxiliary variables $Z_1, \ldots, Z_n \in \{1, \ldots, r\}$ which represent the component allocations of each data point, and use a Gibbs sampler, whose updates are formally:

$$\begin{aligned}(M_1', \ldots, M_r') &\sim \pi_0(\cdot \,|\, Z_1, \ldots, Z_n),\\ Z_i' &\sim \pi_i(\cdot \,|\, M_1', \ldots, M_r'), \quad i = 1, \ldots, n,\end{aligned}$$

where $\pi_0(\cdot \,|\, z_1, \ldots, z_n) = \mathcal{D}(N_1(z) + 1, \ldots, N_r(z) + 1)$ with $N_k(z) = \#\{s : z_s = k\}$, and

$$\pi_i(k \,|\, m_1, \ldots, m_r) = m_k p_k(\eta_i) \Big/ \sum_{j=1}^{r} m_j p_j(\eta_i), \quad k = 1, \ldots, r. \tag{2}$$

Recall that the Dirichlet distribution $\mathcal{D}(\alpha_1 + 1, \ldots, \alpha_r + 1)$ has density on S given by

$$h_\alpha(m_1, \ldots, m_r) = \frac{\Gamma(r + \alpha_1 + \cdots + \alpha_r)}{\Gamma(1 + \alpha_1) \cdots \Gamma(1 + \alpha_r)} m_1^{\alpha_1} \ldots m_r^{\alpha_r}.$$

Implementation of the above Gibbs sampler involves a recursively defined sequence $X_t = (Z_{1t}, \ldots, Z_{nt}, M_{1t}, \ldots, M_{rt})$ such that $X_{t+1} = F_{t+1}(X_t)$, where $F_1, F_2, \ldots$ are independent and identically distributed random functions of the form

$$F(x) = (Z_1'(x), \ldots, Z_n'(x), M_1'(x), \ldots, M_r'(x)). \tag{3}$$

We now give the detail for the construction of F.

To generate a probability vector $(M'_1(x), \ldots, M'_r(x))$ with the Dirichlet $\mathcal{D}(N_1(x)+1, \ldots, N_r(x)+1)$ distribution, it suffices to generate independent random variables $G_1 \sim \Gamma(N_1(x)+1)$, $\ldots$, $G_r \sim \Gamma(N_r(x)+1)$, setting

$$M'_k(x) = G_k(x) / \sum_{k=1}^{r} G_k(x),$$

where $\Gamma(N+1)$ denotes a Gamma distribution with shape parameter $N+1$ and scale parameter 1.

To generate each $Z'_s(x)$, we suggest using the following sequential rejection method: Generate r independent uniform random variables $\xi_{s,1}, \ldots, \xi_{s,r}$. For each component $k = 1, \ldots, r$ in turn, we accept $Z'_s(x) = k$ if

- $p_k(\eta_s) M'_k(x) / \sum_{j=k}^{r} p_j(\eta_s) M'_j(x) > \xi_{s,k}$, and
- no other component $j < k$ has yet been accepted.

This method of generating $Z'_s(x)$ has the advantage of requiring a bounded number of iterations to return an answer, independently of the weights $M'_j(x)$ or the factors $p_k(\eta_s)$.

3.1. *Generating gamma random variables*

We choose to generate $G_k \sim \Gamma(N_k+1)$ by rejection sampling based on a t density with 2 degrees of freedom (Best's algorithm, [**6**, p. 410]). More precisely, let $Y_k/\sqrt{2} \sim g(y) = \frac{1}{2\sqrt{2}}(1+y^2/2)^{-3/2}$, which we generate as $Y_k = (U-1/2)/\sqrt{U(1-U)}$ with $U \sim U[0,1]$, and propose the value

$$G_k = N_k + Y_k \sqrt{3N_k + 2.25}. \tag{4}$$

This is accepted if and only if

$$\log[64U^3(1-U)^3V^2] \leqslant 2(N_k \log[G_k/N_k] - Y\sqrt{3N_k+2.25}), \tag{5}$$

where $V \sim U[0,1]$. It is known that the average number of proposals needed is less than $e\sqrt{6/\pi} \leqslant 4$. Observe that the returned value G_k above is a convex (resp. concave) function of N_k whenever $Y \geqslant 0$ (resp. $Y \leqslant 0$).

We remark for future reference that the full Gibbs update requires a bounded amount of computation.

4. *Determining the bounding sets*

We now consider how to construct bounding sets for the update function F described in Section 3. From the construction of F it may appear that we need rather complex bounding sets, e.g. specifying a set of possible allocations for each z_s combined with a subset of the simplex S. However, the following observation reduces the problem. The construction of each $Z'_s(x)$ depends only on x through $M'_1(x), \ldots, M'_r(x)$, which in turn only depend on x through $N_1(x), \ldots, N_r(x)$. Hence the update function, F, only depends on the numbers $N_k(x)$. This leads us to consider bounding sets of the form $W = \{x : a_k \leqslant N_k(x) \leqslant b_k\}$. Below we describe a practical way of computing constants a'_k and b'_k such that

$$a_k \leqslant N_k(x) \leqslant b_k \quad \text{implies that} \quad a'_k \leqslant N_k(F_t(x)) \leqslant b'_k, \tag{6}$$

or in other words, $P(W) \subseteq W = \{x : a_k \leqslant N_k(x) \leqslant b_k\}$. In particular, by setting $a_k = 0$ and $b_k = n$, we obtain a bounding set that equals Ω.

Suppose that, for each data point η_s, we compute upper and lower bounds for the component ratios, i.e. numbers $\mathrm{LO}_a^b[s,k]$ and $\mathrm{HI}_a^b[s,k]$ such that

$$\mathrm{LO}_a^b[s,k] \leqslant p_k(\eta_s)M_k'(x)/\sum_{j=k}^{r} p_j(\eta_s)M_j'(x) \leqslant \mathrm{HI}_a^b[s,k] \tag{7}$$

for all $x \in \{x : a_k \leqslant N_k(x) \leqslant b_k\}$. We shall give details of this calculation in Sections 4.1 and 4.2. Armed with these bounds, consider the variety of possible values the sequential rejection method assigns to $Z_s'(x)$ as x varies over the allowed configurations.

- If $\xi_{s,k} < \mathrm{LO}_a^b[s,k]$ then every single allowed configuration accepts the proposal $Z_s(x) = k$, provided it hasn't already accepted $Z_s(x) = i$ for $i < k$.
- If $\xi_{s,k} < \mathrm{HI}_a^b[s,k]$ then some (but not necessarily all) configurations accept the proposal $Z_s(x) = k$, so some may still accept a later proposal $Z_s(x) = j$ for $j > k$.
- If $\xi_{s,k} > \mathrm{HI}_a^b[s,k]$ then none of the configurations accept the proposal $Z_s(x) = k$.

We keep track of the possible outcomes for each k by means of two sets

- $\mathrm{LOW}(s) = \{k : \xi_{s,k} < \mathrm{LO}_a^b[s,k] \text{ and } \xi_{s,i} > \mathrm{HI}_a^b[s,i] \text{ for all } i < k\}$,
- $\mathrm{HIGH}(s) = \{k : \xi_{s,k} < \mathrm{HI}_a^b[s,k] \text{ and } \xi_{s,i} > \mathrm{LO}_a^b[s,i] \text{ for all } i < k\}$.

We obviously have $\mathrm{LOW}(s) \subseteq \mathrm{HIGH}(s)$, the lower set representing the (necessarily unique) component accepted by all allowable configurations (this set is empty if the configurations are split) while the upper set represents all those components which are potentially accepted by at least one allowable configuration. It is now clear that we obtain (6) if we choose

$$a_k' = \#\{s : \mathrm{LOW}(s) = \{k\}\} \quad \text{and} \quad b_k' = \#\{s : \mathrm{HIGH}(s) \supseteq \{k\}\}.$$

We also remark that the calculation of the sets $\mathrm{LOW}(s)$ and $\mathrm{HIGH}(s)$ is somewhat biased by the ordering chosen for the components, i.e. since the proposal $Z_s(x) = 1$ is always investigated first, this same component is necessarily overrepresented in the collection of sets $\mathrm{HIGH}(s)$. Similarly, the last component r is necessarily underrepresented. It is easy to avoid this problem (and thereby improve the sandwiching bounds) by permuting randomly the labels for the components for each $s = 1, \ldots, n$. In some cases, the improvement is dramatic, and we shall always assume this is done, although it is suppressed in the notation.

4.1. *Calculation of the lower bound $LO_a^b[s,k]$*

To begin the discussion, consider the generation of the Gamma distributions (4) as a function of x. For each component we generate $Y_k^{(1)}, Y_k^{(2)}, \ldots, Y_k^{(\alpha_k)}$ as in Section 3.1 until (5) is fulfilled for all $a_k \leqslant N_k \leqslant b_k$ for some $Y_k^{(t)}$. Consequently we have

$$G_k(x) \in N_k(x) + \{Y_k^{(1)}, \ldots, Y_k^{(\alpha_k)}\} \cdot \sqrt{3N_k(x) + 2.25}. \tag{8}$$

This way we obtain a superset of the set of values of $(M_1'(x), \ldots, M_r'(x))$ obtained for the configurations in the set $\{x : a_k \leqslant N_k(x) \leqslant b_k\}$.

We use the notation $Y_k^{(t)} = \max\{Y_k^{(i)} : i = 1, \ldots, \alpha_k\}$. Obviously, we have

$$p_k(\eta_s)M_k'(x)/\sum_{j=k}^{r} p_j(\eta_s)M_j'(x) \geqslant \min_{\substack{a_k \leqslant c \leqslant b_k \\ t=0,\ldots,\alpha_k}} \left\{ \frac{p_k(\eta_s)[c + Y_k^{(t)}\sqrt{3c+2.25}]}{p_k(\eta_s)[c + Y_k^{(t)}\sqrt{3c+2.25}] + \max_l \sum_{j=k+1}^{r} p_j(\eta_s)[l_j + \overline{Y}_j\sqrt{3l_j+2.25}]} \right\},$$

where the minimum is over those t such that $c + Y_k^{(t)}\sqrt{3c+2.25} \geqslant 0$, and the maximum is over those vectors l belonging to the set

$$S(\mathbf{a}, \mathbf{b}) := \Big\{ l : a_j \leqslant l_j \leqslant b_j \text{ for all } j > k \text{ and } \sum_{j=k+1}^{r} l_j \leqslant n - c - \sum_{i<k} a_i \Big\}. \qquad (9)$$

Note that this set is convex. The computational cost of this lower bound is obviously at most $O(n)$ times the cost of the maximisation, which as it stands is very expensive. Indeed, unless we have $a_j = b_j$ for all but one component $j > k$, an exhaustive maximisation is out of the question here. However, we can simplify the problem by first finding an upper bound valid for all vectors l of interest, and then maximising this. Indeed, suppose that we have $a_j < b_j$ for at least two different components $j > k$. Then we have the bound

$$\max_l \sum_{j=k+1}^{r} p_j(\eta_s)[l_j + \overline{Y}_j\sqrt{3l_j+2.25}] \leqslant \max_l \sum_{j=k+1}^{r} p_j(\eta_s)[l_j + 0 \vee \overline{Y}_j\sqrt{3l_j+2.25}],$$

where $0 \vee z = \max(0, z)$. The upper bound is now an increasing function of l in each variable, and can therefore be maximised by a greedy hill climbing algorithm: set $l_j = a_j$ initially for all $j > k$, and iteratively allocate the remaining available amount $n - c - \sum_{i<k} a_i$ to those components which at each stage give the highest score increase. The worst case cost of this calculation is $O(n)$, but in practise it appears to run in $O(1)$ steps on average, since the likelihoods $p_k(\eta_s)$ dominate the score calculation (i.e. when a single $p_j(\eta_s)$ is much larger than all the others).

In summary, if $S(\mathbf{a}, \mathbf{b})$ is one dimensional we set

$$\mathrm{MAX}(c) = \max_{l \in S(\mathbf{a},\mathbf{b})} \sum_{j=k+1}^{r} p_j(\eta_s)[l_j + \overline{Y}_j\sqrt{3l_j+2.25}]$$

where the maximisation is exhaustive. Otherwise we set

$$\mathrm{MAX}(c) = \max_{l \in S(\mathbf{a},\mathbf{b})} \sum_{j=k+1}^{r} p_j(\eta_s)[l_j + \overline{Y}_j 1_{(Y_j>0 \text{ or } a_j=b_j)}\sqrt{3l_j+2.25}]$$

where maximisation is obtained using the greedy hill climbing algorithm. Then given $\mathrm{MAX}(c)$ we set

$$\mathrm{LO}_a^b[s,k] = \min_{\substack{a_k \leqslant c \leqslant b_k \\ t=0,\ldots,\alpha_k}} \left\{ \frac{p_k(\eta_s)[c + Y_k^{(t)}\sqrt{3c+2.25}]}{p_k(\eta_s)[c + Y_k^{(t)}\sqrt{3c+2.25}] + \mathrm{MAX}(c)} \right\}.$$

4.2. *Calculation of the upper bound* $\mathrm{HI}_a[s,k]$

The upper bound is calculated similarly. We begin by writing

$$p_k(\eta_s)M'_k(x)/\sum_{j=1}^{r} p_j(\eta_s)M'_j(x) \leqslant \max_{a_k \leqslant c \leqslant b_k} \left\{ \frac{p_k(\eta_s)[c+\overline{Y}_k\sqrt{3c+2.25}]}{p_k(\eta_s)[c+\overline{Y}_k\sqrt{3c+2.25}] + \min_{l,t}\sum_{j=k+1}^{r} p_j(\eta_s)[l_j + Y_j^{(t)}\sqrt{3l_j+2.25}]} \right\},$$

where the minimisation holds over $S(\mathbf{a},\mathbf{b})$ and those values of $Y_j^{(t)}$ satisfying $l_j + Y_j^{(t)}\sqrt{3l_j+2.25} \geqslant 0$. Now unlike the previous case, it seems hard to find a lower bound on this quantity because of the extra positivity requirement for the bracketed quantity (exhaustive minimisation being out of the question). We opt therefore for the cruder estimate

$$\mathrm{MIN}(c) = \sum_{j=k+1}^{r} p_j(\eta_s) \min_{\substack{a_j \leqslant l \leqslant b_j \\ t=1,\ldots,\alpha_j}} [l + Y_j^{(t)}\sqrt{3l+2.25}],$$

with the proviso $l + Y_j^{(t)}\sqrt{3l+2.25} \geqslant 0$, after which we set

$$\mathrm{HI}_a^b[s,k] = \max_{\substack{a_k \leqslant c \leqslant b_k \\ t=0,\ldots,\alpha_k}} \left\{ \frac{p_k(\eta_s)[c+Y_k^{(t)}\sqrt{3c+2.25}]}{p_k(\eta_s)[c+Y_k^{(t)}\sqrt{3c+2.25}] + \mathrm{MIN}(c)} \right\}.$$

5. *Specifying the compound update function*

In this section we consider how to generate perfect samples from (1) using the roCFTP algorithm using three different choices of the compound random update function C. In the following $\Omega = \{1,\ldots,r\}^n \times S$ and $F : \Omega \to \Omega$ denotes the random update function (3) described in Section 3.

Initially we assume C to be a compound of K replica of F, so $C_i = F_{i,K} \circ \cdots \circ F_{i,2} \circ F_{i,1}$ where $F_{i,1},\ldots,F_{i,K}$ are independent realisations of F. To determine if C_i is coalescent, we make use of the bounding sets found in Section 4 as follows. Initially set $a_j = 0$ and $b_j = n$, $j = 1,\ldots,r$. Then, sequentially for each $F_{i,k}$, $k = 1,\ldots,K$, update a and b according to the scheme in Section 4. If, after the $(K-1)$st update, we have $a_j = b_j$, $j = 1,\ldots,r$, we declare C_i coalescent, and non-coalescent otherwise. The reason we need $a_j = b_j$ after $K-1$ updates is that $a_j = b_j$ implies coalescence in the allocation space, but not necessarily coalescence in the weight space. The latter is obtained after the Kth update. In practise the bounding sets derived in Section 4 can be quite "sloppy" in the sense that they are much larger than the exact bounding $\{F(x) : x \in W\}$. As mentioned in Section 2.2 this will make the perfect sampler less efficient. Below we will explore one way of alleviating this problem.

5.1. *Catalytic updates*

An alternative to the above approach is to use catalytic perfect simulation introduced by [**3**]. Catalytic perfect simulation is a special case of the roCFTP algorithm where the update functions have been replaced be so-called catalytic updates. A

catalytic update is obtain by modifying an existing update function. The basic idea is that the first catalytic update in each block is constructed in such a way that with high probability it maps the entire state space into a small number of points. When this happens coalescence can be detected by tracking these points under subsequent updates. In practise this eliminates the problem of sloppy bounding sets. The main drawback is that the implementation of the first catalytic update in each block comes at a computationally high price. [**3**] also consider catalytic perfect simulation for the mixture problem. They use an approximative check for coalescence to reduce the computational cost resulting in an imperfect sampler.

First we consider the following simple catalytic update which modifies a realisation of F:

$$\tilde{F}_{x^*}(F)(x) = \begin{cases} \tilde{x} & \text{if } \xi \leqslant \frac{P(x,\tilde{x})}{P(x,F(x))} \frac{P(x^*,F(x))}{P(x^*,\tilde{x})} \\ F(x) & \text{otherwise.} \end{cases} \tag{10}$$

Here $P(x,x') = \pi(m'|z)\pi(z'|m')$ is the transition kernel of F. Further, $x^* \in \Omega$, $\xi \sim \mathrm{U}[0,1]$ and $\tilde{x}$ is a sample from $P(x^*,\cdot)$. In the notation we have suppressed the dependence on $\tilde{x}$ and ξ. It may be helpful to think of the catalytic update as a Metropolis-Hastings update with stationary distribution $P(x,\cdot)$. By invariance and the fact that $F(x)$ is a sample form $P(x,\cdot)$ it follows that $F_{x^*}(F)(x)$ is also a sample from $P(x,\cdot)$. In the following we will denote F a basic update to distinguish it from the catalytic update.

Inspecting (10) it is clear that $F_{x^*}(F)(x^*) = \tilde{x}$ and in general there will be a "basin of attraction" around x^* in which points are mapped to $\tilde{x}$. More precisely the basin of attraction is given by $\mathrm{Basin}(\tilde{x},F,\xi,x^*) = \{x \in \Omega : \xi P(x,F(x))P(x^*,\tilde{x}) \leqslant P(x,\tilde{x})P(x^*,x')\}$. The catalytic update is illustrated in Figure 2. Notice that if $\mathrm{Basin}(\tilde{x},F,\xi,x^*) = \Omega$ then $\tilde{F}_{x^*}(F)$ mapes the entire state space into $\tilde{x}$. Unfortunately, unless Ω is relatively small this is unlikely to happen, see e.g. the centre plot of Figure 2. This motivates using the following variant of the catalytic update, see [**2**] for more details.

Assume that $\mathbf{x}^* = (x_1^*,\ldots,x_\nu^*)$ is a vector of ν states and each x_i^* is associated with a state $\tilde{x}_i \sim P(x_i^*,\cdot)$ and $\xi_i \sim U[0,1]$ both generated independently of everything else. We can then define the general catalytic updates as

$$\tilde{F}_{\mathbf{x}^*}(F) = \tilde{F}_{x_\nu^*} \circ \cdots \circ \tilde{F}_{x_2^*} \circ \tilde{F}_{x_1^*}(F).$$

In words, we first modify F using $F_{x_1^*}$, then modify the resulting modified function using $F_{x_2^*}$ and so on. The general catalytic update can be thought of as the result of running a ν step Metropolis-Hastings chain with stationary distribution given by $P(x,\cdot)$ and initial state $x' = F(x) \sim P(x,\cdot)$. By invariance $F_{\mathbf{x}^*}(F)(x)$ is a sample from $P(x,\cdot)$. Further, we define the basin of attraction for $\tilde{F}_{\mathbf{x}^*}(F)$ as

$$\mathrm{Basin}(\mathbf{x}^*,F) = \cup_{i=1}^n \mathrm{Basin}(\tilde{x}_i,F,\xi_i,x_i^*). \tag{11}$$

Perfect samples can now be obtained using the roCFTP algorithm with

$$C = \tilde{F}_{\mathbf{x}_K^*}(F_K) \circ \cdots \circ \tilde{F}_{\mathbf{x}_2^*}(F_2) \circ \tilde{F}_{\mathbf{x}_1^*}(F_1)(x),$$

where $F_1,\ldots,F_K$ are independent realisations of F and $\mathbf{x}_1^*,\ldots,\mathbf{x}_K^*$ are vectors of states which may be of different length and where the configuration of $\mathbf{x}_t^*$ may depend on the bounding set obtained after the first $t-1$ updates.

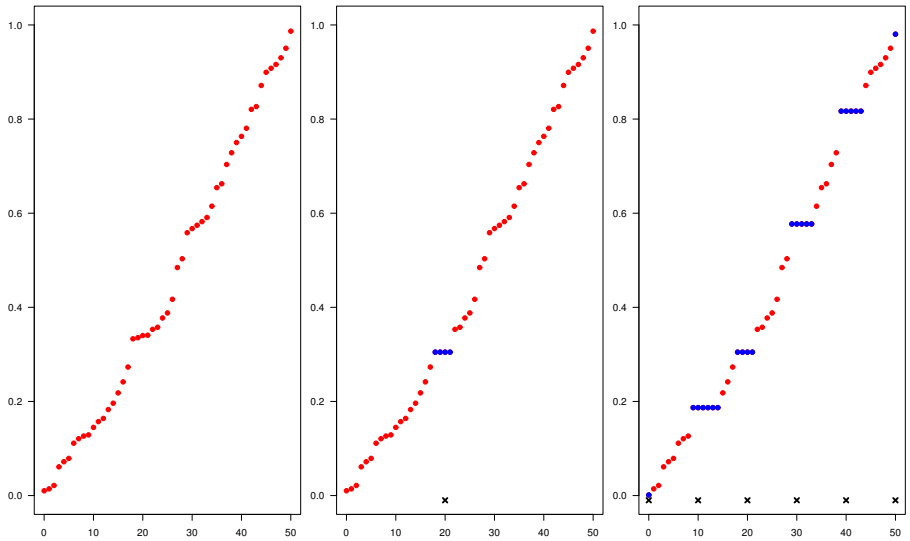

Figure 2: The left plot shows how $M_1(x)$ depends on $N_1(x)$ for a realisation of F when $n = 50$ and $r = 2$. Centre and right plots shows the same but for a catalytic update $\tilde{F}(F)$ and $\tilde{F}_{\mathbf{x}^*}(F)$, respectively. The blue dots are coordinates affected by the modification, i.e. the corresponding values of $N_1(x)$ belong to a basin of attraction. Crosses at the bottom of the centre and right plots indicate the location of the x^*s. Notice how the cardinality of the image decreases from F over $\tilde{F}(F)$ to $\tilde{F}_{\mathbf{x}^*}(F)$.

The detection of coalescence depends crucially on whether $\text{Basin}(\mathbf{x}_1^*, F_1) = \Omega$ or not. If $\text{Basin}(\mathbf{x}^*, F) = \Omega$ then $\tilde{F}_{\mathbf{x}^*}(F)$ maps the state space into a subset of the ν points $\tilde{x}_1, \ldots, \tilde{x}_\nu$. Accordingly we let the bounding set be $W = \{\tilde{x}_1, \ldots, \tilde{x}_\nu\}$. After this the bounding set is updated as follows: if W is the current bounding set then the next bounding set is $W' := F_{\mathbf{x}^*}(F)(W) = \{F_{\mathbf{x}^*}(F)(x) : x \in W\}$. Hence the bounding sets are all exact with the exception of the first as $\tilde{F}_{x_1^*}(F_1)(\Omega)$ may be a strict subset of $\{\tilde{x}_1, \ldots, \tilde{x}_\nu\}$. Coalescence is declared the moment $\#W = 1$. In practise this is checked by tracking each $\tilde{x}_i$ associated with x_1^* as $F_{\mathbf{x}_2^*}(F_2), \ldots, F_{\mathbf{x}_K^*}(F_K)$ are applied. As we have (alomost) eliminated the sloppy bounding sets defined in Section 4 this alogrithm is more effecient in detecting coalescence and inducing coalescence as a small W is likely to be mapped into just one point. This is confirmed by simulation studies showing that coalescence is obtained much faster in terms of the size of K when using catalytic perfect simulation. A further advantage is that this algorithm much simpler to implement.

If $\text{Basin}(\mathbf{x}_1^*, F_1) \neq \Omega$ we do not know if $\tilde{F}_{\mathbf{x}_1^*}(F_1)$ maps Ω into $\{\tilde{x}_1, \ldots, \tilde{x}_\nu\}$ or not. One option would be to check if $\text{Basin}(\mathbf{x}_2^*, F_2) = \Omega$. Instead we choose to declare C non-coalescent and for the remainder of the block we only apply basic updates which is equivalent to setting $\mathbf{x}_2^* = \cdots = \mathbf{x}_K^* = \emptyset$.

In practise it seems difficult to check if $\text{Basin}(\mathbf{x}^*, F) = \Omega$ by other means than brute force, i.e. for each $x \in \Omega$ check that $x \in \text{Basin}(\tilde{x}_i, F, \xi_i, x_i^*)$ for some $i = 1, \ldots, \nu$. This is potentially computationally infeasible but the following observations help reduce the problem. Notice that $P(x, x') = \pi(m'|z)\pi(z'|m')$ so letting $x' = F(x)$ the Hastings ratio in (10) can be rewritten as

$$\begin{aligned}
\frac{P(x,\tilde{x})}{P(x,x')}\frac{P(x^*,x')}{P(x^*,\tilde{x})} &= \frac{\pi(\tilde{m}|z)\pi(\tilde{z}|\tilde{m})}{\pi(m'|z)\pi(z'|m')}\frac{\pi(m'|z^*)\pi(z'|m')}{\pi(\tilde{m}|z^*)\pi(\tilde{z}|\tilde{m})} = \frac{\pi(\tilde{m}|z)}{\pi(m'|z)}\frac{\pi(m'|z^*)}{\pi(\tilde{m}|z^*)} \\
&= \frac{\tilde{m}_1^{N_1(x)}\cdots\tilde{m}_r^{N_r(x)}}{{m'_1}^{N_1(x)}\cdots{m'_r}^{N_r(x)}}\frac{{m'_1}^{N_1(x^*)}\cdots{m'_r}^{N_r(x^*)}}{\tilde{m}_1^{N_1(x^*)}\cdots\tilde{m}_r^{N_r(x^*)}} \\
&= \prod_{k=1}^{r}\left(\frac{\tilde{m}_k}{{m'}_k}\right)^{N_k(x)-N_k(x^*)}. \qquad (12)
\end{aligned}$$

As both m_k and $\tilde{m}$ are generated as described in Section 3 they are only functions of $N_1(x), \ldots, N_r(x)$ and $N_1(x^*), \ldots, N_r(x^*)$, respectively. Hence, the Hastings ratio (12) only depends on x and x^* though $N_1(x), \ldots, N_r(x)$ and $N_1(x^*), \ldots, N_r(x^*)$. So verifying $\text{Basin}(\mathbf{x}^*, F) = \Omega$ amounts to checking that each valid configuration of $N_1(x), \ldots, N_k(x)$ belongs to one of the ν basins of attraction. As the number of valid configurations is

$$\sum_{k=1}^{r}\binom{n-1}{r-k}\binom{r}{k-1}$$

the complexity of checking if $\text{Basin}(\mathbf{x}^*, F) = \Omega$ is of order $O(\nu n^{r-1})$. Even though this reduces the computational cost of verifying $\text{Basin}(\mathbf{x}^*, F) = \Omega$ it may still be infeasible in practise. This problem can be reduced by combining basic and catalytic updates.

5.2. *Delayed catalyst*

So far we have considered two ways of constructing C. Either it consist in applying K basic updates or K catalytic updates. Assume instead that C is constructed by applying basic updates until the cardinality of the bounding set $W = \{x : a_j \leqslant N_j(x) \leqslant b_j\}$ is below some given threshold. When this happens we start applying catalytic updates until a total number of K updates have been applied. Note that if the cardinality of W does not get below the threshold no catalytic updates are applied. In the following let $\tilde{K}$ denote the number of basic updates applied to get below the threshold. With this setup perfect samples are obtain using the roCFTP algorithm with

$$C(x) = \tilde{F}_{\mathbf{x}^*_{K-\tilde{K}}}(F_K) \circ \cdots \circ \tilde{F}_{\mathbf{x}^*_1}(F_{\tilde{K}+1}) \circ F_{\tilde{K}} \circ \cdots \circ F_1(x),$$

where $F_1, \ldots, F_K$ are independent realisations of F and each vector of states $\mathbf{x}^*_i$ may depend on the bounding set obtained after the initial $\tilde{K} + i - 1$ updates.

Coalescence is detected as before by updating the bounding set for each update. For the initial $\tilde{K}$ basic updates the bounding set is updated using the scheme in Section 4. Let W be the bounding set obtained after the initial $\tilde{K}$ updates. Then when the first catalytic update $\tilde{F}_{\mathbf{x}^*_1}(F_{\tilde{K}+1})$ is applied we check if $\mathrm{Basin}(\mathbf{x}^*_1, F_{\tilde{K}+1}) = W$. If this is the case we track the states $\tilde{x}_i$ associated with $\mathbf{x}^*_1$ under the subsequent catalytic updates. If they have coalesced after the $K-1$st updates we declare C coalescent otherwise not. Further, if $\mathrm{Basin}(\mathbf{x}^*_1, F_{\tilde{K}+1}) \neq W$ we declare C non-coalescent and apply basic updates for the remainder of the block.

By waiting until $\#W$ is below the threshold before we apply catalytic updates the computational cost of checking $\mathrm{Basin}(\mathbf{x}^*_1, F_{\tilde{K}+1}) = W$ has hopefully been reduced considerably. This idea of waiting until a bounding set is sufficiently small before a more effective update functions is applied is closely related to the concept of one-shot coupling, see [**10**] and the idea in [**1**] of tracking a bounding set and then apply a more efficient update at the end of the block.

Notice that a threshold of zero corresponds to the case where C consists only of realisation of F. Similarly, a threshold larger than $(n+1)^r$ corresponds to the case where C consists of only modified updates.

5.3. *Specifying* $\mathbf{x}^*$

It remains how to specify $\mathbf{x}^*_i$. Recall that (10) only depends on x^* through $N_1(x^*), \ldots, N_r(x^*)$. So specifying $\mathbf{x}^*_j$ reduces to choosing points $l = (l_1, \ldots, l_n)$ in $\{(l_1, \ldots, l_n) : l_j \in \{1, \ldots, n\}, \sum_{j=1}^r l_j = n\}$. Assume that W is the current bounding set. Then we let $\mathbf{x}^*_i$ be an equally spaced subgrid $\{(l_1, \ldots, l_n) : y_j = a_j + \Delta d, d = 0, \ldots, \lfloor (b_j - a_j)/\Delta \rfloor, \sum_{j=1}^r l_j = n\}$, where Δ is the spacing and $a_k = \min_{x \in W} N_k(x)$ and $b_k = \max_{x \in W} N_k(x)$. With this choice checking $\mathrm{Basin}(\mathbf{x}^*_1, F_{\tilde{K}+1}) = \Omega$ has complexity $O(n^{2(r-1)})$.

It seems appealing that as the range of W increases, so does the number of points in $\mathbf{x}^*$ and that these points are evenly distributed. In practise the main advantage of this choice is when checking if $\mathrm{Basin}(\mathbf{x}^*_1, F_{\tilde{K}+1}) = W$. Recall that checking if $\mathrm{Basin}(\mathbf{x}^*_1, F_{\tilde{K}+1}) = W$ consist in checking for each $x \in W$ if $x \in \mathrm{Basin}(\tilde{x}_i, F, \xi_i, x^*_i)$ for some $i = 1, \ldots, \nu$. Instead of systematically going through all basins we start with the basin associated to $x^* \in \mathbf{x}^*$ chosen so that it minimises $\| N(x) - N(x^*_i) \|$. Due to the grid structure the minimisation is cheap and this trick is expected to

speedup the check of $\text{Basin}(\mathbf{x}_1, F_{\tilde{K}+1}) = W$.

5.4. *Simulation study*

We now consider a small simulation study mainly comparing different choices of the threshold. We assume that each component density p_i is that of a normal distributed random variable with mean μ_i and common variance σ^2.

Figure 3 shows the bounding set for $N_1(x)$ when generating perfect posterior samples using two different choices of the threshold and block size K. The same simulated data set with $n = 100$ points has been used for both simulations. The top plot of figure 3 shows the zero threshold case. In this case the size of the bounding set decreases rapidly to a small size but still struggle to coalesce. In the lower plot the threshold is 45^3. After just a few updates the first catalytic update is applied. With one exception the catalytic update is successful, in the sense that $\text{Basin}(\mathbf{x}_1^*, F_{\tilde{K}+1}) = W$. Notice that for this particular simulation a block is declared coalescent whenever the first catalyst is successful even if the even if a much smaller block size is used.

Intuitively, the more well-separated the true means are, the easier it should be to estimate the unknown weights. Figure 4 shows the bounding set in a situation where the true means are less well separated compared to the true means used in Figure 3. As a result the bounding set tends to remain close to Ω making the zero threshold case impracticable. In fact a threshold close to n^r is required for the perfect sampler to work in practise.

Table 1 compares the computational costs of producing perfect posterior samples for different configurations of the true means, block size, threshold and data size. Whenever a catalytic update is applied, we let $\mathbf{x}^*$ be a grid with a spacing $\Delta = 5$. The results confirm the intuition that as the component densities become more separated, it becomes easier to determine the individual weights and as consequence the Markov chains should converge faster. Further, it can be seen that for increased separation of the true means the advantage of using catalytic updates decreases.

Comparing the results in row 3, 4 and 6 in Table 1 we see the effect of the threshold. Choosing the threshold too large makes the first catalytic update too computationally expensive. On the other hand, choosing the threshold too small may result in the catalytic updates being applied too late to introduce coalescence, or maybe not applied at all.

Comparing rows 4 and 5 of Table 1 show that the block size, K, has little influence on the computing time. This is because when coalescence has been declared, the algorithm is only updating the target process for the remainder of the block which is computationally inexpensive.

6. *Generalisation: Gaussian mixtures with unknown means*

So far we have described how to generate perfect samples from the posterior weight distribution when the mixture components are known. In this section we extend our approach to the case where the components are specified by unknown parameters. Specifically we assume that the r components are Gaussian with common, known variance σ^2 and individual, unknown means $\mu_1, \ldots, \mu_r$. As for the Bayesian setup we assume a uniform prior on the weights and a Gaussian $\mathcal{N}(\mu_*, \tau^2)$ prior for each μ_j.

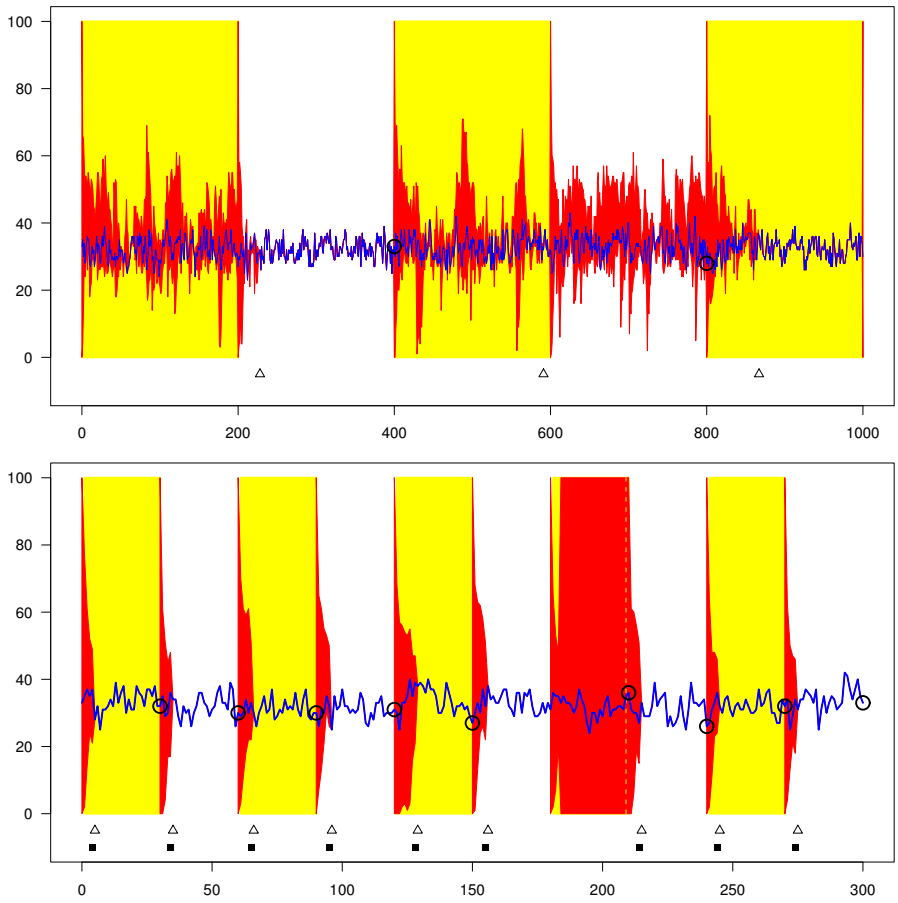

Figure 3: Plots illustrates the roCFTP algorithm with different choices of threshold and block size. For both plot we have used the same simulated data set specified by $n = 100$, $r = 3$, $\mu = (0, 1.1, 2.2)$, $m = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and $\sigma^2 = 0.25$. The red region illustrates the bounding set for $N_1(x)$ given by the interval $[a_1, b_1]$. As the bounding set is not updated when $\text{Basin}(\mathbf{x}_1^*, F_{\tilde{K}+1}) \neq W$ we plot the full interval $[0, n]$ when this happens. The blue line is the target process. Solid squares indicates when the event $\text{Basin}(\mathbf{x}_1^*, F_{\tilde{K}+1}) = W$ occurs, i.e. when the basin of first catalytic update successfully cover the bounding set. Triangles indicate the time a given C_i achieves $\#W = 1$, i.e. when coalescence is declared. Circles indicate a perfect sample from the posterior. The top plot combines a zero threshold with a block size $K = 200$. In the lower plot the block size is $K = 30$ and the threshold is 45^3.

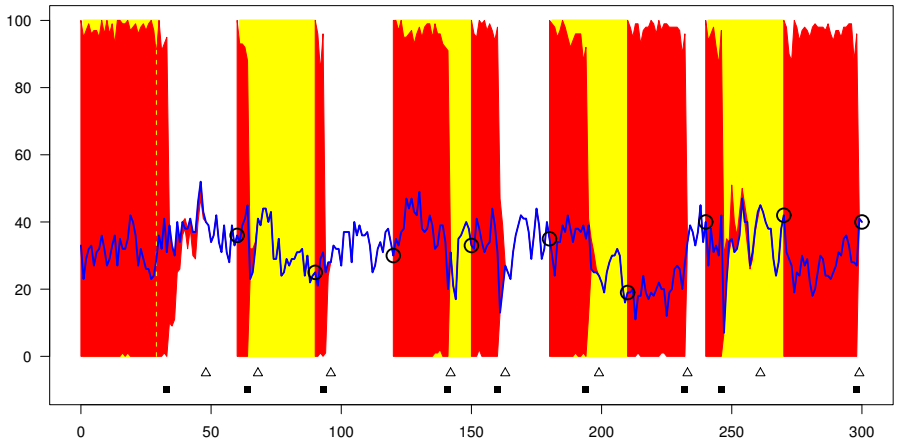

Figure 4: This plot illustrates perfect posterior simulation in a situation where the data analysed was simulated for the setup specified by $n = 100$, $r = 3$, $\mu = (0, 0.5, 1)$, $m = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and $\sigma^2 = 0.25$. For this example $K = 30$ and the threshold is 95^3.

Table 1: Computational cost of perfect simulation. For each row we have the table shows the time for generating 100 independent realisations of the posterior weight distribution. The data analysed was generated using the parameter n, μ, m and σ specified in the first eight columns. For rows with identical specifications of n, μ, m and σ the same data set has been analysed. the delayed catalyst algorithm. The last two rows specify the threshold (Thl) and block size (K) used in the perfect simulation algorithm. When the threshold is zero no catalytic updates are ever used.

n	μ_1	μ_2	μ_3	m_1	m_2	m_3	σ	CPU	Thl	K
100	0	0.5	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.5	3653	95^3	30
100	0	1	2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.5	11767	0	200
100	0	1	2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.5	475	20^3	30
100	0	1	2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.5	181	45^3	30
100	0	1	2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.5	217	45^3	100
100	0	1	2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.5	918	95^3	30
100	0	2	4	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.5	73	20^3	30
100	0	2	4	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.5	68	0	50
200	0	2	4	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.5	391	20^3	30
200	0	2	4	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.5	413	0	50

As in Section 3 we start by considering how to simulate approximatively from the posterior using a Gibbs sampler. Given an augmented state $x = (z_1, \ldots, z_n, m_1, \ldots, m_r, \mu_1, \ldots, \mu_r)$, we first update the means followed by updates of the weights and allocations. This leads us to specify an augmented Gibbs evolution of the form

$$F(x) = (Z_1'(x), \ldots, Z_n'(x), m_1'(x), \ldots, m_r'(x), \mu_1'(x), \ldots, \mu_r'(x)) \tag{13}$$

where

$$\mu_k'(x) \sim \mathcal{N}\left(\frac{\sigma^{-2} \sum_{s \in A_k(x)} \eta_s + \tau^{-2}\mu_*}{\sigma^{-2}\,|A_k(x)| + \tau^{-2}}, (\sigma^{-2}\,|A_k(x)| + \tau^{-2})^{-1} \right), \tag{14}$$

and $A_k(x) = \{s : z_s = k\}$. Note that $N_k(x) = |A_k(x)|$ in our previous notation. The updates $M_k'(x)$ and $Z_s'(x)$ are as in Section 3 but with $p_j(v)$ replaced by

$$p_k(v, x) = \exp -[v - \mu_k'(x)]^2/2\sigma^2.$$

By analogy with the previous case, we want to track a bounding set which in the present setup is of the general form

$$\Big\{ x : \underline{A}_k \subseteq A_k(x) \subseteq \overline{A}_k,\ k = 1, \ldots, r \Big\}. \tag{15}$$

The set $\overline{A}_k$ corresponds to the set of data points that could be allocated to the kth component, whereas $\underline{A}_k$ corresponds to set of data points allocated to the kth component and no other component. The full set of possible configurations is initially of the above type, with $\underline{A}_k = \emptyset$ and $\overline{A}_k = \{1, \ldots, n\}$ for each k.

As before, the bounds we shall derive are used to obtain expressions of the type

$$\mathrm{LO}[\underline{A}_k, \overline{A}_k] \leqslant M_k'(x) / \sum_{j=k}^{r} [p_j(\eta_s, x)/p_k(\eta_s, x)] M_j'(x) \leqslant \mathrm{HI}[\underline{A}_k, \overline{A}_k]. \tag{16}$$

These limits are in turn used exactly as before, to generate the sets $\mathrm{LOW}(s)$ and $\mathrm{HIGH}(s)$ for each data point η_s. It remains to define

$$\underline{A}_k' = \{s : k \in \mathrm{LOW}(s)\} \text{ and } \overline{A}_k' = \{s : k \in \mathrm{HIGH}(s)\},$$

which completes the sandwiching calculation, giving

$$F\Big(\{x : \underline{A}_k \subseteq A_k(x) \subseteq \overline{A}_k\}\Big) \subseteq \{x : \underline{A}_k' \subseteq A_k(x) \subseteq \overline{A}_k'\}.$$

We now explain how to define the random mapping $F(x)$ in (13). The explicit implementation details for $M_k'(x)$ and $Z_s'(x)$ remain unchanged, once the $\mu_k'(x)$ have been computed. We therefore concentrate on describing this case.

If we know that $|A_k(x)| = c$ say, then (14) becomes simply

$$\mu_k'(x) \sim \frac{\eta(A_k(x)) + (\sigma^{-2}c + \tau^{-2})^{1/2}\mathcal{N}(0,1)}{\sigma^{-2}c + \tau^{-2}}, \tag{17}$$

where $\eta(A) = \sigma^{-2} \sum_{s \in A} \eta_s + \tau^{-2}\mu_*$ satisfies the constraints

$$\underline{\eta}(c) := \min_{\substack{\underline{A}_k \subseteq A \subseteq \overline{A}_k, \\ |A| = c}} \eta(A) \leqslant \eta(A_k(x)) \leqslant \max_{\substack{\underline{A}_k \subseteq A \subseteq \overline{A}_k, \\ |A| = c}} \eta(A) =: \overline{\eta}(c).$$

It is important to note that these upper and lower bounds can be quickly calculated, for each c separately, if we assume (as we shall from now on) that the data points $\eta_1, \ldots, \eta_n$ are sorted according to size.

For each c, we may generate a mapping $\mu'_k(x) = F(x)$ satisfying (17) for all x with $|A_k(x)| = c$, and containing only a small number of image points (hopefully much smaller than $\binom{n}{c}$, the number of different subsets of size c sandwiched between $\underline{A}_k$ and $\overline{A}_k$.). This may be done in generality using Field Couplers ([**2**]), but here we prefer to use the technique of Layered Multishift Coupling introduced by [**12**], which requires fewer computations but is restricted to the Gaussian and other unimodal distributions with easily invertible densities. The formula is given explicitly as (writing $\lfloor z \rfloor$ for the integer part of z)

$$\mu'_k(x) = \frac{1}{\sqrt{\sigma^{-2}c + \tau^{-2}}}\left(W_k + \left\lfloor \frac{\eta(A_k(x))/\sqrt{\sigma^{-2}c+\tau^{-2}} + R_k - W_k}{R_k - L_k} \right\rfloor (R_k - L_k)\right), \tag{18}$$

where $W_k \sim \mathcal{N}(0,1)$, and we compute $L_k = -\sqrt{-2\log(1-Y_k)}$ and $R_k = \sqrt{-2\log Y_k}$ by setting $U_k \sim U[0,1]$, and $Y_k = U_k \exp[-W_k^2/2]$ if $W_k > 0$, but $Y_k = 1 - U_k \exp[-W_k^2/2]$ if $W_k \leqslant 0$. An important consequence is that

$$\mu'_k(x) \in \left\{ (\sigma^{-2}c + \tau^{-2})^{-1/2}\Big[W_k + \ell(R_k - L_k)\Big] \text{ if } |A_k(x)| = c,\ \ell \in \mathbb{N} \right\}$$

where ℓ is an integer satisfying

$$\left\lfloor \frac{\underline{\eta}(c)/\sqrt{\sigma^{-2}c+\tau^{-2}} + R_k - W_k}{R_k - L_k} \right\rfloor \leqslant \ell \leqslant \left\lfloor \frac{\overline{\eta}(c)/\sqrt{\sigma^{-2}c+\tau^{-2}} + R_k - W_k}{R_k - L_k} \right\rfloor. \tag{19}$$

By allowing c to vary from $c = \left|\underline{A}_k\right|$ to $c = \left|\overline{A}_k\right|$ (but keeping R_k, L_k, and W_k fixed for simplicity), we therefore arrive at a complete cover of all the possible values which could be taken by $\mu'_k(x)$, for $x \in \{x : \underline{A}_k \subseteq A_k(x) \subseteq \overline{A}_k\}$. The average computational cost for this estimate is $O(N \log N)$ for the initial sort (worst case $O(N^2)$), and at most $O(N)$ for each computation of $\underline{\eta}(c)$ and $\overline{\eta}(c)$, hence giving $O(N^2)$ altogether. This final computation is repeated for each separate component.

Bounds on the likelihood ratios. For each c, let $\mathrm{MEAN}_k(c)$ denote the set of possible values taken by $\mu'_k(x)$ as x varies over the set $\{x : \underline{A}_k \subseteq A_k(x) \subseteq \overline{A}_k\}$ with $|A_k(x)| = c$, in accordance with the formula (18). If η_s is any datapoint, and $|A_j(x)| = l$, $|A_k(x)| = c$, we therefore have

$$\min_{\substack{\mu_j \in \mathrm{MEAN}_j(l) \\ \mu_j \in \mathrm{MEAN}_k(c)}} \exp -\frac{1}{2\sigma^2}\Big[(\eta_s - \mu_k)^2 - (\eta_s - \mu_j)^2\Big] \leqslant p_j(x,\eta_s)/p_k(x,\eta_s)$$
$$\leqslant \max_{\substack{\mu_j \in \mathrm{MEAN}_j(l) \\ \mu_j \in \mathrm{MEAN}_k(c)}} \exp -\frac{1}{2\sigma^2}\Big[(\eta_s - \mu_k)^2 - (\eta_s - \mu_j)^2\Big],$$

Using the formula (18), both the minimum and maximum are straightforward to calculate. We formalise this by writing

$$\underline{p}_{sjk}(l,c) \leqslant p_j(x,\eta_s)/p_k(x,\eta_s) \leqslant \overline{p}_{sjk}(l,c) \text{ on } \{x : |A_j(x)| = l,\ |A_k(x)| = c\}. \tag{20}$$

Define

$$\overline{u}_k(c,\eta_s) = \frac{1}{2\sigma^2}\left[\max_{\underline{z}(c) \leqslant i \leqslant \overline{z}(c)} \Big(\eta_s - (\sigma^{-2}c + \tau^{-2})^{-1/2}(W_k + i(R_k - L_k))\Big)^2\right] \tag{21}$$

and

$$\underline{u}_k(c,\eta_s) = \frac{1}{2\sigma^2}\left[\min_{\underline{z}(c)\leqslant i\leqslant \overline{z}(c)}\Big(\eta_s - (\sigma^{-2}c+\tau^{-2})^{-1/2}(W_k + i(R_k-L_k))\Big)^2\right] \quad (22)$$

where $\underline{z}(c)$ and $\overline{z}(c)$ are respectively the lower and upper bounds given in (19). The left bound in (20) is then given explicitly by

$$\underline{p}_{sjk}(l,c) = \exp\big\{\underline{u}_k(c,\eta_s) - \overline{u}_j(l,\eta_s)\big\}.$$

Note that the minimum in (22) is achieved when i equals one of the four values

$$\underline{z}(c),\ \overline{z}(c),\ \Big\lfloor \frac{\sqrt{\sigma^{-2}c+\tau^{-2}}\eta_s - W_k}{R_k-L_k}\Big\rfloor,\ \Big\lceil \frac{\sqrt{\sigma^{-2}c+\tau^{-2}}\eta_s - W_k}{R_k-L_k}\Big\rceil.$$

The maximum in (21) is achieved when i equals one of the two values $\underline{z}(k),\ \overline{z}(k)$. These observations speed up the calculation considerably.

The right bound in (20) satisfies

$$\overline{p}_{sjk}(l,c) = \exp\big\{\overline{u}_k(c,\eta_s) - \min_{a_j\leqslant \ell\leqslant l}\underline{u}_j(\ell,\eta_s)\big\}$$

Note the asymmetry in the definitions above. This is needed to cope with the asymmetry in the calculations of the upper and lower bounds below.

Upper bound in (16)**.** Mimicking the upper bound calculation in Section 4, we have from (16) and (4) the bound

$$M_k'(x)/\sum_{j=1}^r [p_j(x,\eta_s)/p_k(x,\eta_s)]M_j'(x) \leqslant \max_{a_k\leqslant c\leqslant b_k}$$
$$\left\{\frac{[c+\overline{Y}_k\sqrt{3c+2.25}]}{[c+\overline{Y}_k\sqrt{3c+2.25}] + \sum_{j=k+1}^r \min_{\substack{a_j\leqslant l\leqslant b_j\\ t=0,\dots,\alpha_j}} \underline{p}_{sjk}(l,c)[l+Y_j^{(t)}\sqrt{3l+2.25}]}\right\},$$

where the minimisation is over those values of $Y_j^{(t)}$ satisfying $l+Y_j^{(t)}\sqrt{3l+2.25}\geqslant 0$.

Lower bound in (16)**.** This also is a slight generalisation of the computation described in Section 4. The starting point is given by the formula

$$M_k'(x)/\sum_{j=1}^r [p_j(\eta_s)/p_k(\eta_s)]M_j'(x) \geqslant \min_{\substack{a_k\leqslant c\leqslant b_k\\ t=0,\dots,\alpha_k}}$$
$$\left\{\frac{[c+Y_k^{(t)}\sqrt{3c+2.25}]}{[c+Y_k^{(t)}\sqrt{3c+2.25}] + \max_l \sum_{j=k+1}^r \overline{p}_{skj}(c)[l_j+\overline{Y}_j\sqrt{3l_j+2.25}]}\right\},$$

where the maximisation is over $S(\mathbf{a},\mathbf{b})$ (as defined in (9)) as in Section 4, and we have set $\overline{p}_{skj}(c) = \max_{a_j\leqslant l\leqslant b_j} p_{skj}(l,c)$ (for otherwise the maximisation is not so simple). If all but a single component $j>k$ satisfies $a_j=b_j$, the maximisation is straightforward and can be done exactly. In case this assumption doesn't hold, we look for an easily maximised upper bound on the maximum. Define

$$w_j(l,\eta_s) = \min_{a_j\leqslant c\leqslant l}\underline{u}_j(c,\eta_s),$$

then we obviously have

$$\max_l \sum_{j=k+1}^{r} e^{\overline{u}_k(c,\eta_s)-\underline{u}_j(l_j,\eta_s)}[l_j+\overline{Y}_j\sqrt{3l_j+2.25}] \leqslant$$

$$\max_l \sum_{j=k+1}^{r} e^{\overline{u}_k(c,\eta_s)-w_j(l_j,\eta_s)}[l_j+0\vee\overline{Y}_j\sqrt{3l_j+2.25}],$$

where the latter sum is increasing as a function of each l_j, thus amenable to the same greedy hill climbing algorithm as before.

6.1. *Simulation experiments*

In practise this algorithm only seems to be feasible for quite small data sets ($n < 5$) where data consist of one cluster. In other words, the algorithm works best if the likelihood that the data came from just one Gaussian distribution is high.

To understand why this algorithm does not work satisfactory consider the following. The algorithm relies on generating a random map that maps all possible configuration of allocations, weights and means into just one configuration. Assume $r = 2$ and that data consists of two distinct clusters points. Consider a configuration where the left most cluster corresponds to component one (and right most cluster corresponds to component two). A coalescent map should then "flip" the component means and component allocations. Given the way allocations are updated here this would happen with an infeasibly small probability.

6.2. *Integrating out the mean*

In an attempt to alleviate the problems indicated above we integrating out $\mu_1,\ldots,\mu_r$ in the posterior density. The resulting joint density for m and z is proportional to

$$\prod_{k=1}^{r} m_k^{N_k(x)}(N_k(x)\sigma^{-2}+\tau^{-2})^{-\frac{1}{2}}\exp\left[\frac{1}{2}\frac{(\eta(A_k(x))\sigma^{-2}+\mu_*\tau^{-2})^2}{N_k(x)\sigma^{-2}+\tau^{-2}}\right].$$

Defining

$$\zeta_j(i,x) = \left(\sigma^{-2}(N_j(x_{-s})+i)+\tau^{-2}\right)^{-\frac{1}{2}} \times \exp\left(\frac{1}{2}\frac{\left(\sigma^{-2}(\eta(A_j(x_{-s}))+i\eta_s)+\tau^{-2}\mu_*\right)^2}{\sigma^{-2}(N_j(x_{-s})+i)+\tau^{-2}}\right)$$

we obtain

$$\mathbb{P}(Z_s=k|m,Z_s\geqslant k,Z_{-s}) = m_k\big(\zeta_k(1,x)/\zeta_k(0,x)\big)\Big/ \left[m_k\big(\zeta_k(1,x)/\zeta_k(0,x)\big)+\sum_{j=k+1}^{r} m_j\big(\zeta_j(1,x)/\zeta_j(0,x)\big),\right]$$

where $Z_{-s}=(Z_1,\ldots,Z_{s-1},Z_{s+1},\ldots,Z_n)$. Notice that m_k is a function of $N_k(x)$. Even though component means have been eliminated problems with identifiability persists. Complicating things further is the fact that the allocation variables are no

longer conditional independent given the weights. In fact the conditional probability $\mathbb{P}[Z_s = k|Z_{-s}, m]$ depends on the exact configuration of Z.

One solution to the identifiability issue is to restrict the state space to those allocations z where $\frac{\eta(A_k(x))}{|A_k(x)|} \leqslant \frac{\eta(A_j(x))}{|A_k(x)|}$ when $k < j$. One way of sampling under this restriction is using a Metropolis-within-Gibbs sampler: For each Z_i we propose a new allocation. Given the proposal we find a lower bound on the acceptance probability. This implies, in principle, minimising the Hastings ratio over all allowed allocation fulfilling the restriction imposed above. The Hastings ratio is

$$\begin{aligned}\frac{\mathbb{P}(Z_s = k|Z_{-s})}{\mathbb{P}(Z_s = j|Z_{-s})} = &\left\{ m_k \sqrt{\frac{\sigma^{-2}N_k(Z_{-s}) + \tau^{-2}}{\sigma^{-2}(N_k(Z_{-s}) + 1) + \tau^{-2}}} \times \right. \\ &\exp\left(\frac{1}{2} \frac{(\sigma^{-2}(\eta(A_k(Z_{-s})) + \eta_s) + \tau^{-2}\mu_*)^2}{\sigma^{-2}(N_k(Z_{-s}) + 1) + \tau^{-2}} \right. \\ &\left. \left. - \frac{1}{2} \frac{(\sigma^{-2}\eta(A_k(Z_{-s})) + \tau^{-2}\mu_*)^2}{\sigma^{-2}N_k(Z_{-s}) + \tau^{-2}} \right) \right\} \times \\ &\left\{ \frac{1}{m_j} \sqrt{\frac{\sigma^{-2}(N_j(Z_{-s}) + 1) + \tau^{-2}}{\sigma^{-2}N_j(Z_{-s}) + \tau^{-2}}} \times \right. \\ &\exp\left(- \frac{1}{2} \frac{(\sigma^{-2}(\eta(A_j(Z_{-s})) + \eta_s) + \tau^{-2}\mu_*)^2}{\sigma^{-2}(N_j(Z_{-s}) + 1) + \tau^{-2}} \right. \\ &\left. \left. + \frac{1}{2} \frac{(\sigma^{-2}\eta(A_j(Z_{-s})) + \tau^{-2}\mu_*)^2}{\sigma^{-2}N_j(Z_{-s}) + \tau^{-2}} \right) \right\}\end{aligned}$$

We refer to the expression in the two sets of curly brackets as the k term and j term, respectively. Given m_k, $N_k(Z_{-s})$, m_j and $N_j(Z_{-s})$ it is straight forward to find upper and lower bounds for $\eta(A_k(Z_{-s}))$ and $\eta(A_j(Z_{-s}))$. Further, the exponents in the k and j terms are second order polynomials in $\eta(A_k(Z_{-s}))$ and $\eta(A_j(Z_{-s}))$, respectively. So given m_k, $N_k(Z_{-s})$, m_j and $N_j(Z_{-s})$ minimising the Hastings ratio reduces to minimising a simple function of two variables under the constraint that $\eta(A_k(Z_{-s})) + \eta_s \leqslant \eta(A_j(Z_{-s})) - \eta_s$. The final minimisation is obtain by going through $j = 1, \ldots, k$ and run through the (super)set of possible values of $N_k(Z_{-s})$ and $N_j(Z_{-s})$. Notice that m_k and m_j depend on $N_k(Z)$ and $N_j(Z)$, respectively. In practise this method does not seem to give any significant improvement compared to the situation where the means are not integrated out.

7. *Hidden Markov models*

In this section we extend our approach to perfect simulation of the posterior transition probabilities in a discrete state hidden Markov model. Essentially this means that instead of assuming that the allocation variables $Z_0, \ldots, Z_n$ are independent and identically distributed we assume that $Z = (Z_0, \ldots, Z_n)$ is a Markov chain. For simplicity we assume that Z is a two state Markov chain with transition matrix $Q = \{q_{ij} : i, j \in \{1, 2\}\}$, i.e. $\mathbb{P}(Z_{s+1} = j|Z_s = i) = q_{ij}$. This implies a state vector $x = (z_0, \ldots, z_n, q_{11}, q_{12}, q_{21}, q_{22})$.

It is easily seen that this Markov chain has stationary distribution with density proportional to $(1-q_{22}, 1-q_{11}) = (q_{21}, q_{12})$. As before, we a priori assume that $\eta_1, \ldots, \eta_n$ are independent given Z and that $\eta_s | Z_s = i \sim p_i$. Assuming that the Markov chain Z is started in equilibrium the joint likelihood for $\eta = (\eta_0, \ldots, \eta_n)$ and Z given Q is

$$\pi(\eta, Z|Q) = \prod_{s=1}^{n} q_{Z_{s-1}Z_s} \prod_{s=0}^{n} p_{Z_s}(\eta_s)(q_{21} + \mathbb{1}[Z_0 = 2](q_{12} - q_{11}))/(q_{21} + q_{12}).$$

To obtain a simple posterior we assume that the joint priori density for q_{11} and q_{22} is proportional to $q_{12} + q_{21}$. The posterior density is then proportional to

$$\prod_{s=0}^{n} p_{z_s}(\eta_s) \prod_{s=1}^{n} q_{z_{s-1}z_s}(q_{21} + \mathbb{1}[z_0 = 2](q_{12} - q_{21})).$$

7.1. *Specifying the update function*

The posterior can be sampled using a Gibbs sampler noticing that conditional on Z and η the posterior distributions of q_{11} and q_{22} are independent and beta:

$$q_{11}|Z = z, \eta \sim \text{beta}(N_{11}(x) + 1, N_{12}(x) + \mathbb{1}[z_0 = 2] + 1) \tag{23}$$

and

$$q_{22}|Z = z, \eta \sim \text{beta}(N_{22}(x) + 1, N_{21}(x) + \mathbb{1}[z_0 = 1] + 1),$$

where $N_{ij}(x) = \#\{s : 1 \leqslant s \leqslant n, z_{s-1} = i, z_s = j\}$ is the number of transitions from state i to state j in z. The conditional posterior probabilities for the allocation variables are

$$\begin{aligned}
\mathbb{P}(Z_0 = 1|Q, \eta, Z_{-0}) &= p_1(\eta_0)q_{21}q_{1z_1} / \left[p_1(\eta_0)q_{21}q_{1z_1} + p_2(\eta_0)q_{12}q_{2z_1}\right] \\
\mathbb{P}(Z_s = 1|Q, \eta, Z_{-s}) &= p_1(\eta_s)q_{z_{s-1}1}q_{1z_{s+1}} / \\
&\qquad \left[p_1(\eta_s)q_{z_{s-1}1}q_{1z_{s+1}} + p_2(\eta_s)q_{z_{s-1}2}q_{2z_{s+1}}\right] \\
&\qquad \text{for} \quad 1 \leqslant s \leqslant n-1, \\
\mathbb{P}(Z_n = 1|Q, \eta, Z_{-n}) &= p_1(\eta_n)q_{z_{n-1}1} / [p_1(\eta_n)q_{z_{n-1}1} + p_2(\eta_n)q_{z_{n-1}2}]
\end{aligned}$$

where $Z_{-s} = (Z_0, \ldots, Z_{s-1}, Z_{s+1}, \ldots, Z_n)$.

As in Section 3 the Gibbs sampler can be expressed as a random update function

$$F(x) = (Z_0'(x), \ldots, Z_n'(x), q_{11}'(x), q_{22}'(x)).$$

To generate $q_{11}'(x)$ from the beta distribution (23) generate gamma distributed variables $G_{11} \sim \Gamma(N_{11}(x) + 1)$ and $G_{12}(x) \sim \Gamma(N_{12}(x) + \mathbb{1}[Z_0 = 2] + 1)$ and set $q_{11}'(x) = G_{11}(x)/(G_{11}(x) + G_{12}(x))$. The gamma distributed random variables are generated as in Section 3, i.e.

$$G_{ij}(x) \in N_{ij}(x) + \{Y_{ij}^{(1)}, \ldots, Y_{ij}^{(\alpha_{ij})}\} \cdot \sqrt{3N_{ij}(x) + 2.25}.$$

The variable $q_{22}'(x)$ is generated in a way similar to that of $q_{11}'(x)$.

To generate $Z_0'(x)$, generate $\xi_0 \sim U[0,1]$ and if

$$\xi_0 \leqslant p_1(\eta_0)q_{21}'q_{1z_1}' / \left[p_1(\eta_0)q_{21}'q_{1z_1}' + p_2(\eta_0)q_{12}'q_{2z_1}'\right]$$

set $Z'_0 = 1$ otherwise set $Z'_0(x) = 2$. To generate $Z'_s(x)$, $s = 1, \ldots, n-1$ generate $\xi_s \sim U[0,1]$ and if

$$\xi_s \leqslant p_1(\eta_s) q'_{z'_{s-1}} q'_{1 z_{t+1}} / [p_1(\eta_s) q'_{z'_{s-1}1} q'_{1 z_{s+1}} + p_2(\eta_s) q'_{z'_{s-1}2} q'_{2 z_{s+1}}]$$

set $Z'_s(x) = 1$ otherwise set $Z'_s(x) = 2$. To generate $Z'_n(x)$ generate $\xi_n \sim U[0,1]$ and if

$$\xi_n \leqslant p_1(\eta_n) q'_{z'_{n-1}1} / [p_1(\eta_n) q'_{z'_{n-1}1} + p_2(\eta_n) q'_{z'_{n-1}2}]$$

set $Z'_n(x) = 1$ otherwise set $Z'_n(x) = 2$.

7.2. *Determining the bounding sets*

Unlike in Section 4 here $F(x)$ depends on the exact configuration of x. Consequently we consider bounding sets of the form $W[A] = \{x : z_i \in A_i\}$, where $A = (A_1, \ldots, A_n)$ and $A_1, \ldots, A_n$ are non-empty subsets of $\{1, 2\}$. Notice that whereas in Section 6 we have sets $\overline{A}_k$ and $\underline{A}_k$ for each component here we have a set A_s for each data point. Given A and a realisation of F we give a practical way of determining $A' = (A'_1, \ldots, A'_n)$ so that $x \in W[A]$ implies $F(x) \in W[A']$.

Assume that for each t we can find bounds $\mathrm{LO}_s[A]$ and $\mathrm{HI}_s[A]$ so that

$$\mathrm{LO}_s[A] \leqslant \mathbb{P}(Z_s(x) = 1 | Q(x), Z_{-s}(x), \eta) \leqslant \mathrm{HI}_s[A], \quad \forall x \in W[A],$$

where $Z_{-s}(x) = (Z'_0(x), \ldots, Z'_{s-1}(x), z_{s+1}, \ldots, z_n)$. Then if $\xi_s \leqslant \mathrm{LO}[A]_s$ we know that for any $x \in W[A]$ we have $Z'_s(x) = 1$, hence we set $A'_s = \{1\}$. If $\mathrm{LO}[A]_s < \xi_s \leqslant \mathrm{HI}[A]_s$ then the state of $Z'_s(x)$ may be either 1 or 2 depending on x, hence $A'_s = \{1, 2\}$. Finally, if $\xi_s > \mathrm{HI}[A]_s$ then $Z'_s(x) = 2$ for all $x \in W[A]$, hence $A'_s = \{2\}$.

When finding the bounds, we consider three separate cases: $s = 0$, $1 \leqslant s \leqslant n-1$ and $s = n$. In all three cases this involves a maximisation/minimisation over a (super)set of possible configurations of $N = (N_{11}, N_{12}, N_{21}, N_{22})$. To determine this set we notice that for the two state Markov chain we consider here it is clear that the difference between N_{12} and N_{21} is at most one. Further, as $\sum_{i=1}^{2} \sum_{j=1}^{2} N_{ij} = n$ this implies $N_{12} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[z_0 = 1]((n - N_{11} - N_{22}) \mod 2)$ and $N_{21} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[z_0 = 2]((n - N_{11} - N_{22}) \mod 2)$.

Assume we have bounds $\underline{N}_{ij}$ and $\overline{N}_{ij}$ so that $\underline{N}_{ij} \leqslant N_{ij}(x) \leqslant \overline{N}_{ij}$ for all $x \in W[A]$ and $i, j = 1, 2$. Then $\underline{N}_{11} \leqslant N_{11} \leqslant \overline{N}_{11}$ and given N_{11}, $\underline{N}_{22} \leqslant N_{22} \leqslant \min(\overline{N}_{22}, n - N_{11})$. Given N_{11} and N_{22} and for each value of $\tilde{z} \in A_0$ we have $N_{12} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[\tilde{z} = 1]((n - N_{11} - N_{22}) \mod 2)$ and $N_{21} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[\tilde{z} = 2]((n - N_{11} - N_{22}) \mod 2)$. This leads us to define the set

$$\begin{aligned} N[A] = \{(N_{11}, N_{12}, N_{21}, N_{22}) : \underline{N}_{11} \leqslant N_{11} \leqslant \overline{N}_{11}, \\ \underline{N}_{22} \leqslant N_{22} \leqslant \min(n - N_{11}, \overline{N}_{22}), \\ N_{12} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[\tilde{z} = 1]((n - N_{11} - N_{22}) \mod 2), \\ N_{21} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[\tilde{z} = 2]((n - N_{11} - N_{22}) \mod 2), \tilde{z} \in A_0\}. \end{aligned}$$

We are now ready to define the bounds. In the case $s = 0$

$$\mathrm{LO}_0[A] = \min_{\substack{N \in N[A] \\ \tilde{z} \in A_1}} \frac{p_1(\eta_0)(1 - q_{22}(N)) q_{1\tilde{z}_1}(N)}{p_1(\eta_0)(1 - q_{22}(N)) q_{1\tilde{z}_1} + p_2(\eta_0)(1 - q_{11}(N)) q_{2\tilde{z}_1}(N)},$$

Table 2: Each row in the table corresponds to a single data set. Each data set is a realisation of the hidden Markov model specified by the parameters given in the first six columns in the corresponding row. The perfect simulation algorithm was run until it had produced 100 perfect samples. The time taken to produce the 100 perfect samples is shown in the last column.

n	μ_1	μ_2	q_{11}	q_{22}	σ	CPU time (sec.)
25	-1	1	0.3	0.6	0.5	61
100	-1	1	0.3	0.6	0.5	10512

where

$$q_{11}(N) = \frac{G_{11}}{G_{11}+G_{12}} = \frac{N_{11}Y_{11}^{(t_{11})}\sqrt{3N_{11}+2.25}}{N_{11}Y_{11}^{(t_{11})}\sqrt{3N_{11}+2.25}+N_{12}Y_{12}^{(t_{12})}\sqrt{3N_{12}+2.25}}$$

and t_{11} and t_{12} are minimal w.r.t. fulfilling (5). The variable $q_{22}(N)$ given in a similar way. More over, $q_{12}(N) = 1 - q_{11}(N)$ and $q_{21}(N) = 1 - q_{22}(N)$.

For $1 \leqslant s \leqslant n-1$

$$\mathrm{LO}_s[A] = \min_{\substack{N\in N[A] \\ \tilde{z}\in A'_{s-1},\tilde{\tilde{z}}\in A_{t+1}}} \frac{p_1(\eta_s)q_{\tilde{z}1}(N)q_{1\tilde{\tilde{z}}}(N)}{p_1(\eta_s)q_{\tilde{z}1}(N)q_{1\tilde{\tilde{z}}}(N)+p_2(\eta_s)q_{\tilde{z}2}(N)q_{2\tilde{\tilde{z}}}(N)}, \tag{24}$$

and for $s = n$

$$\mathrm{LO}_n[A] = \min_{\substack{N\in N[A] \\ \tilde{z}\in A'_{n-1}}} \frac{p_1(\eta_n)q_{\tilde{z}1}(N)}{p_1(\eta_n)q_{\tilde{z}1}(N)+p_2(\eta_n)q_{\tilde{z}2}(N)}, \tag{25}$$

Each HI_s is defined as LO_s except minimisation is replaced by maximisation. Notice that that (24) and (25) involves the set A'_{s-1}. This implies that LO_{s-1} and HI_{s-1} have to be determined before LO_s and HI_s can be determined.

The minimisation and maximisation over $N[A]$ has a worst case complexity of $O(n^2)$. With n data points the worst case complexity for determining A' is $O(n^3)$.

7.3. *Generating perfect samples*

As in Section 5 we use Wilson's read-once algorithm for perfect simulation. The main difference is that we do not make use of catalytic updates as they are computationally infeasible. Catalytic updates were feasible in Section 5 because determining $\mathrm{Basin}(Y_1, F_{\tilde{K}+1})$ could be reduced from checking all valid configurations of z to all valid configurations of N. It is not clear how one can construct a catalytic update in the hidden Markov chain context with a similar computationally convenient property.

Table 2 shows the computational cost of two different data sizes. In both cases p_i is the density of a normal distributed random variable with mean μ_i and variance σ^2. Furthermore, each C_i is a compound of 10 update functions. It seems that a quadrupling of the size of the data result in a dramatic increase in the computational cost as could be expected given the complexity of the algorithm.

8. *Discussion*

We have introduced a general methodology for perfect simulation for mixture models. Although the applications of our techniques do not extend as far as the class of problems addressed by MCMC, the methods are exact, and more applicable than existing method for perfect simulation.

The use of catalytic updates, though computationally expensive, seem particular useful for harder problems. The advantage is two-fold, first of all it helps us declare coalescence much sooner than would be possible without it. This is mainly due to the fact that the catalytic move, if successful, maps (a subset of) the state space into a "small" countable number of states. Secondly the catalytic update increases the probability of coupling when upper and lower bounds are close.

It is somewhat unsatisfactory that checking the basin of the first catalytic update can only be determined by brute force. Further research should go into improving this. Another open question considers whether there exists a optimal way of choosing Y. A simulation study, not considered here, suggests that an unevenly spaced grid reduce the number of catalysts needed. Another open questions is if it possible to develop theoretical guidelines for how to choose the threshold.

Acknowledgement

This research was supported by the European Union TMR network ERB-FMRX-CT96-0095 on "Spatial and Computational Statistics" and by an Engineering and Physical Sciences Research Council grant.

References

1. Alexandros Beskos and Gareth O. Roberts, 'One-shot CFTP; application to a class of truncated Gaussian densities.' *Methodology and Computing in Applied Probability* 31 (2005) 407–437.

2. L. A. Breyer and G. O. Roberts, 'A new coupling construction for random fields.' *LMS Journal of Computational Mathematics* 3 (2002) 161–177.

3. Laird. A. Breyer and Gareth O. Roberts, 'Catalytic perfect simulation.' *Methodology and Computing in Applied Probability* 3 (2001) 161–177.

4. G. Casella, K. L. Mengersen, C. P. Robert and D. M. Titterington, 'Perfect samplers for mixtures of distributions.' *Journal of the Royal Statistical Society, Ser. B* 64 (2002) 777–790.

5. Gilles Celeux, Merrilee Hurn and Christirn P. Robert, 'Computational and inferential difficulties with mixture posetrior distributions.' *Journal of the American Statistical Association* 95 (2000) 957–970.

6. Luc Devroye, *Non-Uniform random variate generation* (Springer-Verlag, New York, 1986).

7. Jean Diebolt and Christian P. Robert, 'Estimation of finite mixture distributions through bayesian sampling.' *Journal of the Royal Statistical Society, Ser. B* 56 (1994) 363–375.

8. James P. Hobert, Christian P. Robert and D. M. Titterington, 'On perfect simulation for some mixtures of distributions.' *Statistics and Computing* 9 (1999) 287–298.

9. James G. Propp and David B. Wilson, 'Exact sampling with coupled Markov chains and applications to statistical mechanics.' *Random Structures and Algorithms* 9 (1996) 223–252.

10. Gareth O. Roberts and Jeffrey S. Rosenthal, 'One-shot coupling for certain stochastic recursive sequences.' *Stochastic Processes and their Applications* 99 (2002) 195–208.

11. David B. Wilson, 'How to couple from the past using a read-once source of randomness.' *Random Structures and Algorithms* 16 (2000) 85–113.

12. David B. Wilson, 'Layered multishift coupling for use in perfect sampling algorithms (with a primer on CFTP).' 'Monte Carlo Methods,' (ed. Neil Madras). Amer. Stat. Soc., (2000), vol. 26 of *Fields Institute Communications* pp. 141–176, pp. 141–176.

Kasper K. Berthelsen `k.berthelsen@lancaster.ac.uk`
Laird A. Breyer
Gareth O. Roberts `g.o.roberts@lancaster.ac.uk`

Department for Mathematics and Statistics
Lancaster University
Lancaster
LA1 4YF
UK