A new rejection sampling method without using hat function and exact simulation for the posterior of finite mixture models

Hongsheng Dai†

University of Essex, UK

Summary.
It is nontrivial to draw exact realisations from the posterior of finite mixture models, for which all existing exact Monte Carlo simulation methods are not practical or just work theoretically. Motivated by this problem, this paper proposes a new exact simulation method, which simulates a realisation from a proposal density and then uses exact simulation of a Langevin diffusion to check whether the proposal should be accepted or rejected. Comparing to the existing rejection sampling method, the new method does not require the proposal density function to bound the target density function. The new method is much more efficient than existing methods for simulation from the posterior of finite mixture models.

Keywords: Conditioned Brownian motion; Coupling from the past; Diffusion bridges; Exact Monte Carlo simulation; Langevin diffusion; Mixture models; Rejection sampling.

1. Introduction

In Bayesian analysis of complex statistical models, the calculation of posterior normalising constants and the evaluation of posterior estimates, are typically infeasible either analytically or by numerical quadrature. Monte Carlo simulation provides an alternative. Markov chain Monte Carlo (MCMC) methods have been the most popular methods in more than 20 years for statistical analysis of complex data sets. MCMC methods generate statistically dependent and approximate realisations from the target distribution. A potential weakness of these methods is that the simulated trajectory of a Markov chain will depend on its initial state. Concerns about the *quality* of the sampled realisations of the simulated Markov chains have motivated the search for Monte Carlo methods that can be guaranteed to provide samples from the target distribution.

A breakthrough in the search for perfect Monte Carlo simulation methods was made by Propp and Wilson (1996). Their method, named as *coupling from the past* (CFTP), is an MCMC-based algorithm that produces realisations exactly from the target distribution. CFTP transfers the difficulty of running the Markov chain for extensive periods (to ensure convergence) to the difficulty of establishing whether a large number of coupled Markov chains have coalesced. The CFTP algorithm is only practical for small discrete sample spaces or for a target distribution having a probability space equipped a partial order preserved by an appropriate Markov chain construction. Although in recent decades, there have been many theoretical developments and

†*Address for correspondence:* Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK
E-mail: hdaia@essex.ac.uk

applications in this area such as Mira et. al. (2001), Huber (2004), Wilson (2000), Dai (2008) and Dai (2011), CFTP algorithm is still not practical for complex statistical models.

Perfect sampling can also be achieved via rejection sampling method. This involves sampling from a density that bounds a suitable multiple of the target density, followed by acceptance or rejection of the sampled value. In general it is a very challenging task to find a bounding density, although efficient rejection sampling methods have been developed for the special class of one-dimensional log-concave densities (Gilks and Wild, 1992).

In the next subsection, we briefly introduce the finite mixture models and discuss why it is non-trivial to draw exact realisations from the posterior of finite mixture models using existing methods.

1.1. Motivation: exact simulation from the posterior of finite mixture models

This research is motivated by the question: how to draw exact realisations from the posterior of finite mixture models, which is very challenging and has been an open problem for more than a decade. Consider the following mixture model, where the data $\{z_i, i = 1, \cdots, n\}$ are from the density

$$h(z_i; \mathbf{\Theta}) = \sum_{k=1}^{K} p_k h_k(z_i; \mathbf{\Theta}_k), \tag{1}$$

where h_k is the component density and p_k is the component proportion with $0 \leq p_k \leq 1$ and $\sum_k p_k = 1$. Given the prior $\pi_0(\mathbf{\Theta})$ for $\mathbf{\Theta} = (\mathbf{\Theta}_1, \cdots, \mathbf{\Theta}_K)$, we can write the posterior distribution as

$$f(\mathbf{\Theta}) \propto \prod_{i=1}^{n} \left[\sum_{k=1}^{K} p_k h_k(z_i; \mathbf{\Theta}_k) \right] \pi_0(\mathbf{\Theta}). \tag{2}$$

Gibbs sampler for (2) is readily available (Diebolt and Robert, 1994). However, it is very difficult to diagnose the convergence of the MCMC algorithms for the above posterior of mixture models. Some illustrations of this are given by Fearnhead and Meligkotsidou (2007). Celeux et al. (2000) even argue that 'almost the entirety of MCMC samplers implemented for mixture models has failed to converge'. Therefore, it is important to find an efficient method to draw exact realisations from (2). This is extremely challenging when $(\boldsymbol{p}, \mathbf{\Theta})$ is unknown, because by expanding (2) the posterior is a sum of K^n terms, which is a huge number even for small values of K and n and makes direct simulation from (2) to be non-trivial.

Note that for certain simpler versions of (1), exact simulation from its posterior is readily available. For example for the simple mixture models with known $\mathbf{\Theta}$, methods in Hobert et. al. (1999) and Fearnhead (2005) and the method of adaptive rejection sampling for log-concave densities base on the algorithm in Leydold J. (1998) can draw exact realisations from its posterior. The application of all these methods is limited to small sample sizes and small number of components. Dai (2007) proposed a rejection sampling method, called Geometric-Arithmetic Mean (GAM) method, to draw from the posterior of the simple mixture model. This method can deal with large sample sizes and large number of components and is much more efficient than all the other existing methods. Although practical methods are available for simple mixture models, it is extremely difficult to apply them to the mixture models with unknown component parameters $\mathbf{\Theta}$.

If z_i is a discrete random variable, say from a mixture of Poisson distributions, then many terms in the expanded posterior can be merged which makes it possible to draw from (2) directly (Fearnhead, 2005). For continuous random variable z_i, however, this is not the case. For instance z_i is from the mixture of normal densities with

$$h_k(z_i;\theta_k,\nu) = |\nu| e^{-\frac{\nu^2}{2}(z_i-\theta_k)^2}. \tag{3}$$

By discretising continuous data, Fearnhead and Meligkotsidou (2007) extend the direct simulation method to continuous case and suggested to using an importance sampling approach. Another method (Casella et al., 2002) investigates the use of a CFTP algorithm, called the perfect slice sampler, to simulate from (2), but this method only works theoretically.

Note that the rejection sampling method does not work either since the posterior (2) is not log-concave and it is very challenging to find a good hat function. In summary, existing methods are based on either direct simulation, simple rejection sampler or CFTP algorithms, and they do not work practically.

For the above reasons, concerns about the *quality* of the MCMC realisations and lack of efficient and exact simulation methods for the mixture of normal (or other continuous) densities motivate us to develop a new rejection sampling method to solve this problem.

1.2. The new idea and the structure of the paper

To introduce the idea of the new method in this paper, we first consider the decomposition of a density f, as $f(\cdot) = g_1(\cdot)g_2(\cdot)$, where g_1 and g_2 are also proper density functions and it is easy to simulate from them. Note that here f, g_1 and g_2 are density functions up to a normalising constant. Such a decomposition can be find easily, especially for Bayesian posterior distributions. For example, we can decompose the posterior in (2) as

$$\left\{\prod_{i=1}^{n_1}\left[\sum_k p_k h_k(z_i;\boldsymbol{\Theta})\right]\sqrt{\pi_0}\right\}\cdot\left\{\prod_{i=n_1+1}^{n}\left[\sum_k p_k h_k(z_i;\boldsymbol{\Theta})\right]\sqrt{\pi_0}\right\}.$$

If we can find M such that $g_2(\cdot) \leq M$, then traditional rejection sampling can be used to draw samples from f with the hat function $M \cdot g_1$. In practice, we may not be able to find M or M is too large to make the rejection sampling efficient.

Our idea is not to find the hat function for f, but to independently simulate $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from g_1 and g_2, respectively. If the two independent samples $\boldsymbol{x}_1 = \boldsymbol{x}_2 = \boldsymbol{y}$ then it is easy to show (at least for discrete variables and heuristically for continuous variables) that the value $\boldsymbol{y}$ must be from $f(\cdot) \propto g_1(\cdot)g_2(\cdot)$. Note that when f, g_1 and g_2 are densities for discrete random variables, it is possible to simulate $\boldsymbol{y}$ from f using the above idea since $\mathrm{P}(\boldsymbol{x}_1 = \boldsymbol{x}_2) > 0$, but for continuous densities this is impossible since $\mathrm{P}(\boldsymbol{x}_1 = \boldsymbol{x}_2) = 0$.

Although it is impossible to achieve $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ with distance 0 for continuous case, the simulated $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, if they are very *close* (defined in later sections), can be viewed as approximately from the target f. Our idea is to use $\boldsymbol{x}_1$ (or $\boldsymbol{x}_2$) as a proposal and then accept $\boldsymbol{x}_1$ (or $\boldsymbol{x}_2$) as a perfect sample from f based on perfect Monte Carlo simulation of diffusion bridges (Beskos et al., 2006, 2008). We will show that the new method is an exact simulation algorithm theoretically and via simulation studies. The new method is more efficient than all existing exact simulation methods when applying to simulations from the posterior of Bayesian mixture models.

This paper is organized as follows. In Section 2 we present the new methodology and show it is an exact simulation algorithm theoretically and via simulations for a toy example. We also demonstrate that the new algorithm is related to CFTP algorithm. In Section 3, we present an improved version of the exact simulation algorithm. The improvement is achieved via two aspects. One aspect is to find a better decomposition of $f = g_1 \cdot g_2$ and the other aspect is improve the perfect Monte Carlo simulation algorithms for diffusion bridges in (Beskos et al., 2006) and Beskos et al. (2008). In Section 4 we provide a generalised version of the new method. Then we apply the the new method to the mixture of normal densities and demonstrate that the new method is more practical than all existing algorithms in Section 5. Section 6 provides a discussion.

2. Methodology

2.1. Preliminaries

Consider the target density $f(\boldsymbol{x})$ with support $\mathbf{R}^q$ for a q-dimensional random variable $\boldsymbol{X}$. Suppose that it is non-trivial to simulate from f and that f can be decomposed as a product of two proper density functions, $f(\boldsymbol{x}) \propto g_1(\boldsymbol{x})g_2(\boldsymbol{x})$. Assume that we can easily simulate from g_1 and g_2. We can further write $f(\boldsymbol{x}) \propto f_1(\boldsymbol{x})f_2(\boldsymbol{x})$ with $f_1 = g_1^2$ and $f_2 = g_2/g_1$. Clearly f_1 is also a proper density. Note that here f, f_1, g_1 and g_2 are densities up to a constant.

Let

$$\begin{aligned} A(\boldsymbol{x}) &= \frac{1}{2}\log f_1(\boldsymbol{x}) = \log g_1(\boldsymbol{x}) \\ \boldsymbol{\alpha}(\boldsymbol{x}) &= (\alpha^{(1)}, \cdots, \alpha^{(q)})^{tr}(\boldsymbol{x}) = \nabla A(\boldsymbol{x}) \end{aligned} \tag{4}$$

where ∇ is the partial derivative operator for each component of $\boldsymbol{x}$.

Then we consider a q-dimensional diffusion process $\boldsymbol{X}_t(\boldsymbol{\omega}), t \in [0,T]$ $(T < \infty)$, defined on the space $\boldsymbol{\Omega} = (C[0,T]^q, \mathcal{B}(C[0,T]^q))$, given by

$$d\boldsymbol{X}_t = \boldsymbol{\alpha}(\boldsymbol{X}_t)dt + d\boldsymbol{B}_t, \tag{5}$$

where $\vec{\boldsymbol{\omega}} = \{\boldsymbol{\omega}_t, t \in [0,T]\}$ a typical element of $\boldsymbol{\Omega}$. Under the probability measure $\mathbb{W}_{\boldsymbol{\omega}_0}$, the coordinate mapping process $\boldsymbol{B}_t(\vec{\boldsymbol{\omega}}) = \boldsymbol{\omega}_t$ is a Brownian motion starting at $\boldsymbol{B}_0 = \boldsymbol{\omega}_0$. Let $\mathbb{W}$ be the probability measure for a Brownian motion with the initial probability distribution $\boldsymbol{B}_0 = \boldsymbol{\omega}_0 \sim f_1(\cdot)$.

From the equations in (4) we know that the above $\boldsymbol{X}(t)$ is a Langevin diffusion (Hansen, 2003) with the invariant distribution $f_1(\boldsymbol{x})$, which means $\boldsymbol{X}_t \sim f_1(\boldsymbol{x})$ for any $t \in [0,T]$ if $\boldsymbol{X}_0 \sim f_1(\boldsymbol{x})$. Let $\mathbb{Q}_{\boldsymbol{\omega}_0}$ be the probability law induced by $\boldsymbol{X}_t, t \in [0,T]$, given $\boldsymbol{X}_0 = \boldsymbol{\omega}_0$. Let $\mathbb{Q}$ be the probability law induced by $\boldsymbol{X}_t, t \in [0,T]$, with $\boldsymbol{X}_0 = \boldsymbol{\omega}_0 \sim f_1(\cdot)$, i.e. under $\mathbb{Q}$ we have $\boldsymbol{X}_t \sim f_1(\boldsymbol{x})$ for any $t \in [0,T]$.

We shall assume that $\boldsymbol{\alpha}$ satisfies the following standard conditions. Note that under careful variable transformations it is usually possible to guarantee that $\boldsymbol{\alpha}$ satisfies these conditions. We will demonstrate this by the toy example in Section 2.5 and via the posterior of mixture models in Section 5.

Condition 2.1. *$\boldsymbol{\alpha}$ is continuously differentiable in all its arguments.*

Condition 2.2. *There exists $l > -\infty$ such that*

$$\phi(\boldsymbol{x}) = \frac{1}{2}(||\boldsymbol{\alpha}||^2 + \mathbf{div}\ \boldsymbol{\alpha})(\boldsymbol{x}) - l \geq 0, \tag{6}$$

where **div** *is the divergence of $\boldsymbol{\alpha}$, defined in (9).*

Condition 2.3. *The following*

$$\exp\left(\int_0^T \boldsymbol{\alpha}(\boldsymbol{\omega}_s)d\boldsymbol{\omega}_s - \frac{1}{2}\int_0^T ||\boldsymbol{\alpha}(\boldsymbol{\omega}_s)||^2 ds\right)$$

is a martingale with respect to each measure $\mathbb{W}_{\boldsymbol{\omega}_0}$.

Consider a *biased* diffusion process $\bar{\mathbf{X}} = \{\bar{\boldsymbol{X}}_t; 0 \le t \le T\}$ defined as follows. The starting and ending points are $\bar{\boldsymbol{X}}_0 \sim f_1(\boldsymbol{x})$, $\bar{\boldsymbol{X}}_T \sim f(\boldsymbol{x})$ and given $(\bar{\boldsymbol{X}}_0, \bar{\boldsymbol{X}}_T)$ the process $\{\bar{\boldsymbol{X}}_t, 0 < t < T\}$ is given by the diffusion bridge driven by (5). Note that $\bar{\mathbf{X}}$ is actually a biased version of $\mathbf{X}$. Conditional on the right ending point the two processes $\bar{\mathbf{X}}$ and $\mathbf{X}$ have the same distribution.

Lemma 2.1. *Let $\bar{\mathbb{Q}}$ be the probability law induced by $\bar{\mathbf{X}}$. Then we have the Radon-Nikodym derivative:*

$$\frac{d\bar{\mathbb{Q}}}{d\mathbb{Q}}(\vec{\boldsymbol{\omega}}) \propto f_2(\boldsymbol{\omega}_T). \tag{7}$$

Proof. The proof of the lemma follows easily from the proof of Proposition 3 in Beskos and Roberts (2005). □

To draw a sample from the target distribution $f(\boldsymbol{x})$ we need to simulate a process $\bar{\boldsymbol{X}}_t, t \in [0, T]$ from $\bar{\mathbb{Q}}$ and then $\bar{X}_T \sim f(\boldsymbol{x})$. The above lemma gives us an implication of how to simulate the process $\bar{\mathbf{X}}$, which will be introduced in the following subsection.

2.2. Simulating the process $\bar{\mathbf{X}}$

We here use similar rejection sampling ideas as that in Beskos et al. (2006) and Beskos et al. (2008). Under conditions 2.1 to 2.3 and following Beskos et al. (2006) we have

$$\frac{d\mathbb{Q}}{d\mathbb{W}}(\omega) = \exp\left[A(\boldsymbol{\omega}_T) - A(\boldsymbol{\omega}_0) - \frac{1}{2}\int_0^T (||\boldsymbol{\alpha}||^2 + \mathbf{div}\ \boldsymbol{\alpha})(\boldsymbol{\omega}_t)dt\right] \tag{8}$$

where

$$\mathbf{div}\ \boldsymbol{\alpha}(\boldsymbol{x}) = \sum_{j=1}^{q} \frac{\partial \boldsymbol{\alpha}^{(j)}(\boldsymbol{x})}{\partial x^{(j)}} \tag{9}$$

and $x^{(j)}$ is the jth component of $\boldsymbol{x}$.

Then we consider a *biased Brownian motion* $\{\bar{\boldsymbol{B}}_t; 0 \le t \le T\}$ defined as $(\bar{\boldsymbol{B}}_0, \bar{\boldsymbol{B}}_T)$ following a distribution with a density $h(\boldsymbol{x}, \boldsymbol{y})$ and $\{\bar{\boldsymbol{B}}_t; 0 < t < T\}$ to be a Brownian bridge given $(\bar{\boldsymbol{B}}_0, \bar{\boldsymbol{B}}_T)$.

Lemma 2.2. *Let $\mathbb{Z}$ be the probability law induced by $\{\bar{\boldsymbol{B}}_t; 0 \le t \le T\}$. We have that the Radon-Nikodym derivative of $\mathbb{Z}$ with respect to $\mathbb{W}$ is given by*

$$\frac{d\mathbb{Z}}{d\mathbb{W}}(\omega) = \frac{h(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)}{f_1(\boldsymbol{\omega}_0)\frac{1}{\sqrt{2\pi T}}e^{-\frac{||\boldsymbol{\omega}_T - \boldsymbol{\omega}_0||^2}{2T}}}. \tag{10}$$

PROOF. Let $\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}$ be the probability measure, under which $\boldsymbol{B}_t(\vec{\boldsymbol{\omega}}) = \boldsymbol{\omega}_t$ (given $\boldsymbol{B}_0 = \boldsymbol{\omega}_0, \boldsymbol{B}_T = \boldsymbol{\omega}_T$) is a Brownian bridge. Let $\mathbb{Z}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}$ be the probability law induced by $\bar{\boldsymbol{B}}_t$ (given $\bar{\boldsymbol{B}}_0 = \boldsymbol{\omega}_0, \bar{\boldsymbol{B}}_T = \boldsymbol{\omega}_T$). From the definition of $\bar{\boldsymbol{B}}_t$ we know that $\bar{\boldsymbol{B}}_t$ and $\boldsymbol{B}_t$ have the same distribution law given $\bar{\boldsymbol{B}}_0 = \boldsymbol{B}_0$ and $\bar{\boldsymbol{B}}_T = \boldsymbol{B}_T$. Choose any set $\boldsymbol{F} \in \mathcal{B}(C[0,T]^q)$. We have

$$\mathbb{Z}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \boldsymbol{F}\} = \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \boldsymbol{F}\}.$$

Therefore

$$\begin{aligned} E_{\mathbb{Z}}[I[\vec{\boldsymbol{\omega}} \in \boldsymbol{F}]] &= \int_{\mathbf{R}^q}\int_{\mathbf{R}^q} E_{\mathbb{Z}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}}[I[\vec{\boldsymbol{\omega}} \in \boldsymbol{F}]]h(\boldsymbol{\omega}_0,\boldsymbol{\omega}_T)d\boldsymbol{\omega}_0 d\boldsymbol{\omega}_T \\ &= \int_{\mathbf{R}^q}\int_{\mathbf{R}^q} E_{\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}}[I[\vec{\boldsymbol{\omega}} \in \boldsymbol{F}]]h(\boldsymbol{\omega}_0,\boldsymbol{\omega}_T)d\boldsymbol{\omega}_0 d\boldsymbol{\omega}_T \\ &= E_{\mathbb{W}}\left[I[\vec{\boldsymbol{\omega}} \in \boldsymbol{F}]\frac{h(\boldsymbol{\omega}_0,\boldsymbol{\omega}_T)}{f_1(\boldsymbol{\omega}_0)\frac{1}{\sqrt{2\pi T}}e^{-\frac{||\boldsymbol{\omega}_T-\boldsymbol{\omega}_0||^2}{2T}}}\right] \end{aligned}$$

which implies (10). □

By letting

$$h(\boldsymbol{\omega}_0,\boldsymbol{\omega}_T) = f_2(\boldsymbol{\omega}_T)\exp\left[A(\boldsymbol{\omega}_T) - A(\boldsymbol{\omega}_0)\right]f_1(\boldsymbol{\omega}_0)\frac{1}{\sqrt{2\pi T}}e^{-\frac{||\boldsymbol{\omega}_T-\boldsymbol{\omega}_0||^2}{2T}} \tag{11}$$

and using (8), (10) and (7), we have

$$\begin{aligned} &\frac{d\bar{\mathbb{Q}}}{d\mathbb{Z}}(\vec{\boldsymbol{\omega}}) \propto \frac{d\bar{\mathbb{Q}}}{d\mathbb{Q}}(\vec{\boldsymbol{\omega}})\frac{d\mathbb{Q}}{d\mathbb{W}}(\vec{\boldsymbol{\omega}})\frac{d\mathbb{W}}{d\mathbb{Z}}(\vec{\boldsymbol{\omega}}) \\ = \;& f_2(\boldsymbol{\omega}_T)\cdot\exp\left[A(\boldsymbol{\omega}_T) - A(\boldsymbol{\omega}_0) - \frac{1}{2}\int_0^T(||\boldsymbol{\alpha}||^2 + \text{div }\boldsymbol{\alpha})(\boldsymbol{\omega}_t)dt\right]\cdot\frac{f_1(\boldsymbol{\omega}_0)\frac{1}{\sqrt{2\pi T}}e^{-\frac{||\boldsymbol{\omega}_T-\boldsymbol{\omega}_0||^2}{2T}}}{h(\boldsymbol{\omega}_0,\boldsymbol{\omega}_T)} \\ = \;& \exp\left[-\frac{1}{2}\int_0^T(||\boldsymbol{\alpha}||^2 + \text{div }\boldsymbol{\alpha})(\boldsymbol{\omega}_t)dt\right]. \end{aligned} \tag{12}$$

Then under Condition 2.2 we can rewrite (12) as

$$\frac{d\bar{\mathbb{Q}}}{d\mathbb{Z}}(\omega) \propto \exp\left[-\int_0^T \phi(\boldsymbol{\omega}_t)dt\right], \tag{13}$$

which has a value no more than 1. Now it is ready to use rejection sampling to simulate $\bar{\boldsymbol{X}}_t$ from $\bar{\mathbb{Q}}$. First we simulate a proposal $\bar{\boldsymbol{B}}_t$ from $\mathbb{Z}$ and then we accept the proposal as $\bar{\boldsymbol{X}}_t$ according to the probability in (13). Note that this rejection sampling can be done using similar methods as that in Beskos et al. (2006) and Beskos et al. (2008).

Note that to simulate a proposal $\bar{\boldsymbol{B}}_t$ from $\mathbb{Z}$, it is necessary to simulate $(\boldsymbol{\omega}_0,\boldsymbol{\omega}_T)$ from h given in (11). This is not difficult, because according to $f_1 = g_1^2$ and $f_2 = g_2/g_1$ we have

$$\begin{aligned} h(\boldsymbol{\omega}_0,\boldsymbol{\omega}_T) &= f_2(\boldsymbol{\omega}_T)\exp\left[A(\boldsymbol{\omega}_T) - A(\boldsymbol{\omega}_0)\right]f_1(\boldsymbol{\omega}_0)\frac{1}{\sqrt{2\pi T}}e^{-\frac{||\boldsymbol{\omega}_T-\boldsymbol{\omega}_0||^2}{2T}} \\ &= g_2(\boldsymbol{\omega}_T)g_1(\boldsymbol{\omega}_0)\frac{1}{\sqrt{2\pi T}}e^{-\frac{||\boldsymbol{\omega}_T-\boldsymbol{\omega}_0||^2}{2T}}. \end{aligned} \tag{14}$$

We can easily simulate $\boldsymbol{\omega}_0$ from g_1 and $\boldsymbol{\omega}_T$ from g_2 and then accept $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$ as a sample from h according to probability $\exp\left[-\frac{||\boldsymbol{\omega}_T - \boldsymbol{\omega}_0||^2}{2T}\right]$.

2.3. Rejection sampling for $f(\boldsymbol{x}) \propto g_1(\boldsymbol{x})g_2(\boldsymbol{x})$ without using hat function

The previous subsection demonstrated how to simulate $\bar{\boldsymbol{X}}_t, t \in [0,T]$ from $\bar{\mathbb{Q}}$ via rejection sampling technique. From the definition of $\bar{\boldsymbol{X}}_t$ in Section 2.1, we then have that $\bar{\boldsymbol{X}}_T$ is actually a sample from $f_1(\boldsymbol{x})f_2(\boldsymbol{x})$, the target distribution $f(\boldsymbol{x})$. Therefore the following rejection sampling algorithm (Algorithm 1) can be used to simulate $\boldsymbol{x}$ from $f \propto g_1 g_2 = f_1 f_2$.

```
1  Simulate $\boldsymbol{\omega}_0$ from $g_1$ and $\boldsymbol{\omega}_T$ from $g_2$;
2  Simulate a standard uniform variable $U$;
3  if $U \le \exp[-||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T||^2/(2T)]$ then
4      $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$ is from $h$;
5  else
6      Go to step 1;
7  end
8  Simulate the Brownian bridge $\bar{\boldsymbol{B}} = \{\boldsymbol{\omega}_t, t \in (0,T)\}$ conditional on $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$;
9  Simulate $\mathcal{I} = 1$ with probability given by (13);
10 if $\mathcal{I} = 1$ then
11     Output $\boldsymbol{\omega}_T$;
12 else
13     return to step 1;
14 end
```

Algorithm 1: Rejection sampling for $f \propto g_1 g_2 = f_1 f_2$.

Remark 1: Step 9 of Algorithm 1 can be done using the method in Beskos et al. (2006) and Beskos et al. (2008).

Remark 2: Algorithm 1 is a rejection sampling algorithm but it does not require finding a hat function to bound the target density, which is usually the main challenge of the traditional rejection sampling for complicated target densities. The above algorithm uses g_2 as the proposal density function, which does not have to bound the target f.

Choosing an appropriate value T is important for Algorithm 1 to achieve a larger acceptance probability. We can see that the proposal $\boldsymbol{y} = \boldsymbol{\omega}_T$ will be accepted if $U \le \exp[-||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T||^2/(2T)]$ and if $\mathcal{I} = 1$, where $\mathcal{I}$ is the indicator simulated in step 9 of Algorithm 1. Define

$$\begin{aligned} AP_1 &= \mathrm{P}\{U \le \exp[-||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T||^2/(2T)]\}, \\ AP_2 &= E_h\left[\mathrm{P}\{\mathcal{I} = 1|(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)\}\right] \end{aligned} \tag{15}$$

where E_h means the expectation is with respect to $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T) \sim h$. If T is large, the probability AP_1 will be relatively large, but the probability AP_2, the expected value of

$$E_h\left[\mathrm{P}(\mathcal{I} = 1|(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T))\right] = E_{\mathbb{Z}}\left[\exp\left(-\int_0^T \phi(\boldsymbol{\omega}_t)dt\right)\right], \tag{16}$$

will be small. On the contrary, if T is small, the probability AP_2 will be relatively large, but AP_1 will be small. Therefore it is important to choose an appropriate value of T to make the acceptance probability $AP_1 \cdot AP_2$ to be as large as possible. We will discuss the choice of T in later sections via simulation studies.

2.4. The advantage of the new algorithm and its relation to CFTP and direct sampling

2.4.1. The advantage of the new algorithm

Note that in the new algorithm, we do not need g_2 (or g_1) to bound the target density f. Instead, Algorithm 1 makes use of the proposals from both g_1 and g_2 and the acceptance/rejection of a diffusion bridge to draw samples exactly from the target. We can see that the acceptance probability AP_2 in (15) depends on the lower bound l for $(||\boldsymbol{\alpha}||^2 + \text{div } \boldsymbol{\alpha})/2$. Therefore this algorithm will be attractive when it is possible to find good lower bounds for $(||\boldsymbol{\alpha}||^2 + \text{div } \boldsymbol{\alpha})/2$, but difficult to find a good hat function for the target density f. In Section 3, we will demonstrate how to find good lower bounds for $(||\boldsymbol{\alpha}||^2 + \text{div } \boldsymbol{\alpha})/2$. The new method in Section 3 does not require any specified properties for the target function f or $\boldsymbol{\alpha}$, such as log-concativity. This makes the new method more practical than existing adaptive rejection sampling methods. We will also demonstrate this when dealing with the posterior of mixture models in Section 5.

2.4.2. The link to CFTP – a heuristic interpretation

In summary, Algorithm 1 first simulates $\boldsymbol{\omega}_0$ from g_1 and $\boldsymbol{\omega}_T$ from g_2 and then accept $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$ as a sample from h with probability $\exp(-||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T||^2/(2T))$. To accept the proposal $\boldsymbol{\omega}_T$, the algorithm simulate $\mathcal{I} = 1$ via acceptance/rejection of a diffusion bridge.

To explain the link of the new algorithm with CFTP, we temporarily assume that f_2 is a proper density. Note that this assumption is not required by the algorithm.

If we write (14) as

$$h(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T) \quad = \quad f_2(\boldsymbol{\omega}_T) \cdot \left\{ \exp\left[A(\boldsymbol{\omega}_T) - A(\boldsymbol{\omega}_0)\right] f_1(\boldsymbol{\omega}_0) \frac{1}{\sqrt{2\pi T}} e^{-\frac{||\boldsymbol{\omega}_T - \boldsymbol{\omega}_0||^2}{2T}} \right\} \tag{17}$$

the algorithm can be viewed heuristically as doing the following two steps independently: Step [1]: we simulate $\boldsymbol{\omega}_0$ from $g_1(\cdot) = f_1(\cdot)\exp(-A(\cdot))$, then simulate $\boldsymbol{\omega}_T$ from $\exp((\log f_1(\boldsymbol{\omega}_T))/2)\exp(-||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T||^2/(2T))$ and then simulate $\mathcal{I} = 1$ via acceptance/rejection of a diffusion bridge; Step [2]: simulate $\boldsymbol{\omega}_T$ from f_2. Step [1] is equivalent to simulating a diffusion process with invariant distribution f_1. We can also imagine that Step [2] simulates another diffusion process with invariant distribution f_2, but only output the process at time T. The two processes are simulated independently and coalesce at $\boldsymbol{\omega}_T$ where time T is a pre-determined value. This means that two random variables (but having the same values) $\boldsymbol{\omega}_T$ and $\boldsymbol{\omega}_T$ are simulated independently from f_1 and from f_2 respectively. Their joint distribution must be $f_1(\boldsymbol{\omega}_T)f_2(\boldsymbol{\omega}_T) = f(\boldsymbol{\omega}_T)$. Therefore $\boldsymbol{\omega}_T$ is a sample from f.

Recall that CFTP algorithm simulates Markov chains starting from all possible states and uses the same random numbers for each chain. The challenge of CFTP is to monitor coalescence for many different Markov chains. The new method can be viewed heuristically as running two independent diffusion processes, where the product of the invariant distributions of the two diffusions is the target distribution. When the two processes coalesce at a pre-determined time point T (independent of the diffusions), the coalesced point is from the target distribution. The challenge here is to guarantee that the two independent processes coalesce at a pre-determined time point. This challenge is solved via rejection sampling for diffusions, i.e. we choose a value of T first and then use rejection sampling to find the diffusion.

2.4.3. The link to sampling directly from f– a heuristic interpretation
Note that theoretically, we can choose any value of T in Algorithm 1, although T affects the algorithm efficiency. When we choose $T = 0$, Algorithm 1 actually ignores the diffusion simulations, but only involves simulation of $\boldsymbol{\omega}_0$ from g_1 and $\boldsymbol{\omega}_T$ from g_2. The proposal $\boldsymbol{\omega}_T$ will be accepted if $\boldsymbol{\omega}_0 = \boldsymbol{\omega}_T$, since $\exp(||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T||^2/(2T)) = 1$ with $T = 0$ and $||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T|| = 0$ if we define $0/0 = 0$. This means that we independent simulate $\boldsymbol{\omega}_0$ from g_1 and $\boldsymbol{\omega}_T$ from g_2. When $\boldsymbol{\omega}_0 = \boldsymbol{\omega}_T := \boldsymbol{\omega}_*$ we accept $\boldsymbol{\omega}_*$ as a sample from f. Although it is impossible to have $\boldsymbol{\omega}_0 = \boldsymbol{\omega}_T$, this approach can be viewed as simulate $\boldsymbol{\omega}_* \sim g_1 \cdot g_2 = f$.

2.5. A toy example

We end this section by providing a toy example to demonstrate the density decomposition and necessary variable transformation which is to guarantee conditions 2.1 and 2.2 are satisfied. The variable transformation will also be used in Section 5.

EXAMPLE 2.1.

Consider a Dirichlet distribution as the target, having density proportional to $f_p(\boldsymbol{p}) = p_1^4 p_2^4 (1 - p_1 - p_2)^4$, $0 \le p_1, p_2 \le 1$. Since Algorithm 1 requires that the target $f(\cdot)$ should have support in $\mathbf{R}^2$, we first consider the transformed variable $\boldsymbol{x} = (x_1, x_2)$ with

$$\begin{aligned} p_1 &= \exp(x_1)/[1+\exp(x_1)+\exp(x_2)], \\ p_2 &= \exp(x_2)/[1+\exp(x_1)+\exp(x_2)]. \end{aligned} \tag{18}$$

Then the density function for $\boldsymbol{x}$ can be written as

$$f_x(\boldsymbol{x}) \propto \left[\frac{\exp(x_1)}{1+\exp(x_1)+\exp(x_2)}\right]^5 \left[\frac{\exp(x_2)}{1+\exp(x_1)+\exp(x_2)}\right]^5 \left[\frac{1}{1+\exp(x_1)+\exp(x_2)}\right]^5$$

and can be decomposed as $f_x(\boldsymbol{x}) = g_1(\boldsymbol{x}) g_2(\boldsymbol{x})$, with

$$\begin{aligned} g_1(\boldsymbol{x}) &= \left[\frac{\exp(x_1)}{1+\exp(x_1)+\exp(x_2)}\right]^2 \left[\frac{\exp(x_2)}{1+\exp(x_1)+\exp(x_2)}\right]^2 \left[\frac{1}{1+\exp(x_1)+\exp(x_2)}\right]^2 \\ g_2(\boldsymbol{x}) &= \left[\frac{\exp(x_1)}{1+\exp(x_1)+\exp(x_2)}\right]^3 \left[\frac{\exp(x_2)}{1+\exp(x_1)+\exp(x_2)}\right]^3 \left[\frac{1}{1+\exp(x_1)+\exp(x_2)}\right]^3. \end{aligned}$$

Note that $\boldsymbol{\alpha}(\boldsymbol{x})$ satisfies conditions 2.1 and 2.2, since $A(\boldsymbol{x}) = \log(g_1(\boldsymbol{x})) = 2(x_1 + x_2) - 6[\log(1 + \exp(x_1) + \exp(x_2))]$ and

$$\boldsymbol{\alpha}(\boldsymbol{x}) = \begin{bmatrix} 2 \\ 2 \end{bmatrix} - 6 \begin{bmatrix} \dfrac{\exp(x_1)}{1+\exp(x_1)+\exp(x_2)} \\ \dfrac{\exp(x_2)}{1+\exp(x_1)+\exp(x_2)} \end{bmatrix}$$

We further have

$$\text{div } \boldsymbol{\alpha}(\boldsymbol{x}) = -6[1+\exp(x_1)+\exp(x_2)]^{-2}[\exp(x_1)(1+\exp(x_2)) + \exp(x_2)(1+\exp(x_1))]$$

and $||\boldsymbol{\alpha}(\boldsymbol{x})||^2 + \text{div } \boldsymbol{\alpha}(\boldsymbol{x}) \ge -3$.

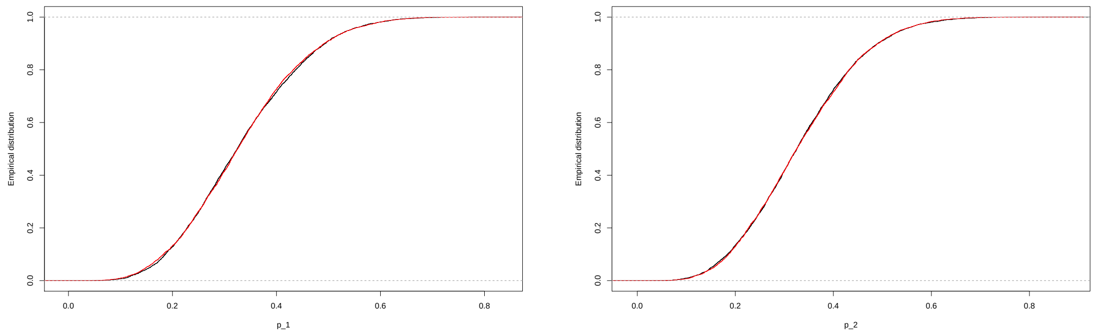

Fig. 1. Left figure: marginal empirical distribution for p_1; right figure: marginal empirical distribution for p_2. Red curve: empirical distribution based on 'rdirichlet'; black cure: empirical distribution based on the new method. For both p_1 and p_2 the red and black curves overlap.

We simulate 5000 realisations using the proposed new method and another 5000 realisations using the 'rdirichlet' command of MCMCpack package of R. The marginal empirical distributions for p_1 and p_2 are plotted for both methods. We can see that the empirical distributions under different methods overlap everywhere, i.e. the empirical distribution for the two methods are almost exactly the same for p_1 (the left figure) and p_2 (the right figure).

3. Improvement of the new rejection sampling algorithm

We can see that in Algorithm 1, there are two rejection steps: line 3 and line 10. The acceptance probabilities are given in (15). These two probabilities can be very small, especially when the dimension, q, of $\boldsymbol{x}$ is large. Therefore to make Algorithm 1 practical, we need to find methods to increase the acceptance probabilities AP_1 and AP_2.

In sections 3.1 and 3.2, we will develop an approach to increase AP_2. Then in Section 3.3, we will show how to increase AP_1 by finding appropriate decomposition of f.

3.1. Improvement for increasing AP_2, when $x \in \mathbf{R}$ – the one-dimensional case

The method proposed in this subsection relies on the concept of *layered Brownian motion* in Beskos et al. (2008). So we first briefly introduce in Section 3.1.1 the *layers* defined in Beskos et al. (2008) and the methods developed therein.

3.1.1. The layered Brownian motion in Beskos et al. (2008)

Let $x = \omega_0, y = \omega_T$, where $(x = \omega_0, y = \omega_T)$ is simulated from line 1 to line 7 of Algorithm 1. Define the probability measure $\mathbb{W}^{x,y}_{0,T}$, under which $\boldsymbol{\omega}_t$ is a Brownian bridge with $(x = \omega_0, y = \omega_T)$ as the starting and ending points.

Let $\{a_i\}_{i\geq 1}$ be an increasing sequence of positive numbers and $a_0 = 0$. Let $\bar{x} = x \wedge y$, $\bar{y} = x \vee y$.

Define the events as $\mathcal{D}_i(\bar{x},\bar{y};0,T) = \mathcal{U}_i(\bar{x},\bar{y};0,T) \cup \mathcal{L}_i(\bar{x},\bar{y};0,T)$, where

$$\begin{aligned} \mathcal{U}_i(x,y;0,T) &= \left\{\vec{\boldsymbol{\omega}} : \sup_{0\le s\le T} \omega_s \in [\bar{y}+a_{i-1}, \bar{y}+a_i]\right\} \cap \left\{\vec{\boldsymbol{\omega}} : \inf_{0\le s\le T} \omega_s > \bar{x}-a_i\right\}, \\ \mathcal{L}_i(x,y;0,T) &= \left\{\vec{\boldsymbol{\omega}} : \inf_{0\le s\le T} \omega_s \in [\bar{x}-a_i, \bar{x}-a_{i-1}]\right\} \cap \left\{\vec{\boldsymbol{\omega}} : \sup_{0\le s\le T} \omega_s < \bar{y}+a_i\right\}. \end{aligned} \quad (19)$$

We say that the Brownian bridge $\vec{\boldsymbol{\omega}}$ is in layer I if $\vec{\boldsymbol{\omega}} \in \mathcal{D}_I$.

In Algorithm 1, the acceptance indicator $\mathcal{I}$ can be simulated via the following subroutine (Beskos et al., 2008).

```
1 Given $\omega_0 = x$ and $\omega_T = y$, simulate a Brownian bridge $\omega_t, t \in [0,T]$ via the following
  steps 1a and 1b:;
  // 1a:  Simulate layer $I$ with probability $\mathrm{P}(I) = \mathbb{W}^{x,y}_{0,T}(\vec{\boldsymbol{\omega}} \in \mathcal{D}_I)$
  // 1b:  Given $\omega_0$ and $\omega_T$, simulate a sample path $\vec{\boldsymbol{\omega}}$, from $\mathbb{W}^{x,y}_{0,T}$
     conditional on $\vec{\boldsymbol{\omega}} \in D_I$, using the algorithm in Beskos et al. (2008)
2 Calculate $l = \inf[\boldsymbol{\alpha}^2(u) + \alpha'(u)]/2$, for all $u \in \mathbf{R}$;
3 Calculate $r_I$ such that $r_I \ge \sup_{t\in[0,T],\vec{\boldsymbol{\omega}}\in\mathcal{D}_I}\{[\alpha^2(\omega_t) + \alpha'(\omega_t)]/2 - l\}$;
4 Simulate $\boldsymbol{\Psi} = \{\psi_1, \cdots, \psi_\rho\}$ uniformly distributed on $\mathbf{U}[0,T]$ and marks
  $\Upsilon = \{\nu_1, \cdots, \nu_\rho\}$ uniformly distributed on $\mathbf{U}[0,1]$, where $\rho$ is from $\mathrm{Poi}(r_I T)$;
5 Compute the acceptance indicator $\mathcal{I} := \prod_{j=1}^{\rho} I[r_I^{-1}\phi(\omega_{\psi_j}) < \nu_j]$;
```

Algorithm 2: Subroutine for steps 8 and 9 in Algorithm 1: simulation for $\mathcal{I}$

3.1.2. *Increase the acceptance probability by re-weighting the layer probabilities*

The acceptance probability $\mathcal{I} = 1$, given by (13), can be very small, if the lower bound l is very small. Therefore, Algorithm 2 may be very inefficient (having acceptance probability close to 0). To increase $\mathrm{P}(\mathcal{I} = 1)$, we should increase the lower bound l. For one-dimensional case, Dai (2013) proposed an adaptive approach to increase the lower bound, which uses different lower bounds of $(\alpha^2 + \alpha')(\omega_s)/2$ for different layers. We here briefly introduce the idea as follows and then extend the method in Dai (2013) to multi-dimensional processes in Section 3.2.

Given $\vec{\boldsymbol{\omega}} \in \mathcal{D}_i$ (the Brownian bridge is in layer i), Condition 2.2 implies that we can find l_i such that $l_i \le \inf_{s\in[0,T],\boldsymbol{\omega}\in\mathcal{D}_i}\{(\alpha^2 + \alpha')(\omega_s)/2\}$ and $l_i \to l$. Obviously such $l_i \ge \boldsymbol{l}$ for all i. Based on the layers and the lower bounds l_i, we consider the following proposal process, $\widetilde{B}_t$ with $(\widetilde{B}_0 = x, \widetilde{B}_T = y) \sim h(x,y)$ and $\widetilde{B}_t, 0 < t < T$ to be a process with measure $\widetilde{\mathbb{Z}}^{x,y}_{0,T}$, where

$$\frac{d\widetilde{\mathbb{Z}}^{x,y}_{0,T}}{d\mathbb{W}^{x,y}_{0,T}}(\vec{\boldsymbol{\omega}}) \quad \propto \quad \sum_{i=1}^{\infty} \exp\{-Tl_i\} I\{\vec{\boldsymbol{\omega}} \in \mathcal{D}_i\}. \quad (20)$$

Then we have the following lemma.

LEMMA 3.1. *Let $\widetilde{\mathbb{Z}}$ be the probability law induced by $\{\widetilde{B}_t, 0 \le t \le T\}$. We have that the Radon-Nikodym derivative of $\widetilde{\mathbb{Z}}$ with respect to $\mathbb{W}$ is given by*

$$\frac{d\widetilde{\mathbb{Z}}}{d\mathbb{W}}(\vec{\boldsymbol{\omega}}) \quad \propto \quad \frac{h(x,y)}{f_1(x) \cdot \frac{1}{\sqrt{2\pi T}} e^{-(x-y)^2/(2T)}} \sum_{i=1}^{\infty} \exp\{-Tl_i\} I\{\vec{\boldsymbol{\omega}} \in \mathcal{D}_i\}$$

PROOF. Using (20), the proof is similar as that of Lemma 2.2. □

We then have

$$\begin{aligned}
&\frac{d\bar{\mathbb{Q}}}{d\widetilde{\mathbb{Z}}}(\vec{\boldsymbol{\omega}}) \propto \frac{d\bar{\mathbb{Q}}}{d\mathbb{Q}}(\vec{\boldsymbol{\omega}})\frac{d\mathbb{Q}}{d\mathbb{W}}(\vec{\boldsymbol{\omega}})\frac{d\mathbb{W}}{d\widetilde{\mathbb{Z}}}(\vec{\boldsymbol{\omega}}) \\
= \ & f_2(\omega_T)\cdot \exp\left[A(\omega_T)-A(\omega_0)-\frac{1}{2}\int_0^T(\alpha^2+\alpha')(\omega_t)dt\right]\cdot\frac{f_1(\omega_0)\frac{1}{\sqrt{2\pi T}}e^{-\frac{|\omega_T-\omega_0|^2}{2T}}}{h(\omega_0,\omega_T)} \\
& \cdot\frac{1}{\sum_{i=1}^{\infty}\exp\{-Tl_i\}I\{\vec{\boldsymbol{\omega}}\in\mathcal{D}_i\}} \\
= \ & \sum_{i=1}^{\infty}\exp\left\{-\int_0^T\left[\frac{1}{2}(\alpha^2+\alpha')(\omega_s)-l_i\right]ds\right\}I\{\vec{\boldsymbol{\omega}}\in\mathcal{D}_i\}, \qquad (21)
\end{aligned}$$

which is also a value no more than 1. Therefore we can also use rejection sampling to simulate from $\bar{\mathbb{Q}}$ if we can simulate from $\widetilde{\mathbb{Z}}(\boldsymbol{\omega})$. The acceptance probability is now given by

$$\sum_{i=1}^{\infty}E_{\mathbb{Z}}\left[\exp\left\{-\int_0^T\left[\frac{1}{2}(\alpha^2+\alpha')(\omega_s)-l_i\right]ds\right\}I\{\boldsymbol{\omega}\in\mathcal{D}_i\}\right], \qquad (22)$$

which will be larger than the acceptance probability in (16).

Note that, simulating from $\widetilde{\mathbb{Z}}(\vec{\boldsymbol{\omega}})$ can actually be done with the method in Dai (2013) if $\vec{\boldsymbol{\omega}}$ is a one dimensional process.

3.2. *Improvement for increasing AP_2, when $\boldsymbol{x}\in\mathbf{R}^q$*

The methodology in Section 3.1 can be extended for q-dimensional processes. Suppose that $\vec{\boldsymbol{\omega}}=(\boldsymbol{\omega}^{(1)},\cdots,\boldsymbol{\omega}^{(q)})$ and $\boldsymbol{\omega}^{(j)}=\{\omega_s^{(j)},s\in[0,T]\}$. Let $\boldsymbol{x}=\boldsymbol{\omega}_0$ and $\boldsymbol{y}=\boldsymbol{\omega}_T$ be simulated from line 1 to line 7 of Algorithm 1. Define $\bar{x}^{(j)}=x^{(j)}\wedge y^{(j)}$ and $\bar{y}^{(j)}=x^{(j)}\vee y^{(j)}$, where $x^{(j)}$ and $y^{(j)}$ are the jth component of $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively.

We define the events $\mathcal{D}_i^{(j)}(\bar{x}^{(j)},\bar{y}^{(j)};0,T)=\mathcal{U}_i^{(j)}(\bar{x}^{(j)},\bar{y}^{(j)};0,T)\cup\mathcal{L}_i^{(j)}(\bar{x}^{(j)},\bar{y}^{(j)};0,T)$, where $\mathcal{U}_i^{(j)}(\cdot,\cdot;0,T)$ and $\mathcal{L}_i^{(j)}(\cdot,\cdot;0,T)$ are defined similarly as before for each component $\boldsymbol{\omega}^{(j)}$. For simplicity the sequence $\{a_i\}$ in (19) is chosen to be same for all components $\boldsymbol{\omega}^{(j)}, j=1,\cdots,q$.

Define event $\boldsymbol{\mathcal{Q}}_i=\otimes_j\left[\cup_{k=1}^{i}\mathcal{D}_k^{(j)}\right]-\otimes_j\left[\cup_{k=1}^{i-1}\mathcal{D}_k^{(j)}\right]$, where the sign $\otimes$ is the direct product for all $j=1,\cdots,q$. Then the q-dimensional Brownian bridge $\vec{\boldsymbol{\omega}}$ belongs to $\boldsymbol{\mathcal{Q}}_i$ is equivalent to that each component of $\boldsymbol{\omega}^{(j)}$ belongs to $\cup_{k=1}^{i}\mathcal{D}_k^{(j)}$ and at least one component $\boldsymbol{\omega}^{(j')}$ belongs to $\mathcal{D}_i^{(j')}$. Clearly $\{\boldsymbol{\mathcal{Q}}_i,i=1,\cdots\}$ form a partition for the space of the q-dimensional $\vec{\boldsymbol{\omega}}$.

With the definition above, we can find l_i such that $l_i\le\inf_{s\in[0,T],\vec{\boldsymbol{\omega}}\in\boldsymbol{\mathcal{Q}}_i}\{(||\boldsymbol{\alpha}||^2+\textbf{div }\boldsymbol{\alpha})(\boldsymbol{\omega}_s)/2\}$ and $l_i\to l$. The same as the one-dimensional case, we consider a process $\widetilde{\boldsymbol{B}}_t$ with $(\widetilde{\boldsymbol{B}}_0=\boldsymbol{x},\widetilde{\boldsymbol{B}}_T=\boldsymbol{y})$ having distribution $h(\boldsymbol{x},\boldsymbol{y})$ and $\{\widetilde{\boldsymbol{B}}_t,0<t<T\}$ having probability law $\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}(\vec{\boldsymbol{\omega}})$ given by

$$\begin{aligned}
\frac{d\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}}{d\mathbb{W}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}}(\vec{\boldsymbol{\omega}}) &= \frac{\sum_{i=1}^{\infty}\exp\{-Tl_i\}I\{\vec{\boldsymbol{\omega}}\in\boldsymbol{\mathcal{Q}}_i\}}{\sum_{i=1}^{\infty}\exp\{-Tl_i\}\mathbb{W}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}\{\vec{\boldsymbol{\omega}}\in\boldsymbol{\mathcal{Q}}_i\}} \qquad (23) \\
&= \sum_{i=1}^{\infty}\left\{\frac{\exp\{-Tl_i\}\mathbb{W}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}\{\vec{\boldsymbol{\omega}}\in\boldsymbol{\mathcal{Q}}_i\}}{\sum_{i=1}^{\infty}\exp\{-Tl_i\}\mathbb{W}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}\{\vec{\boldsymbol{\omega}}\in\boldsymbol{\mathcal{Q}}_i\}}\frac{I\{\vec{\boldsymbol{\omega}}\in\boldsymbol{\mathcal{Q}}_i\}}{\mathbb{W}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}\{\vec{\boldsymbol{\omega}}\in\boldsymbol{\mathcal{Q}}_i\}}\right\}
\end{aligned}$$

where $\mathbb{W}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}$ is the Brownian bridge measure and $\mathbb{W}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}\{\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\}$ is the probability that the Brownian bridge belongs to the event $\boldsymbol{\mathcal{Q}}_i$ (in layer i).

LEMMA 3.2. *Let $\widetilde{\mathbb{Z}}$ be the probability law induced by $\{\widetilde{\boldsymbol{B}}_t, 0 \le t \le T\}$. We have*

$$\frac{d\widetilde{\mathbb{Z}}}{d\mathbb{W}}(\vec{\boldsymbol{\omega}}) \quad \propto \quad \frac{h(\boldsymbol{x},\boldsymbol{y})}{f_1(\boldsymbol{x})e^{-||\boldsymbol{x}-\boldsymbol{y}||^2/(2T)}} \sum_{i=1}^{\infty} \exp\{-Tl_i\} I\{\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\} \tag{24}$$

PROOF. Using (23), the proof is similar as that of Lemma 2.2. □

Then the Radon-Nikodym derivative of $\bar{\mathbb{Q}}$ to $\widetilde{\mathbb{Z}}$ becomes

$$\frac{d\bar{\mathbb{Q}}}{d\widetilde{\mathbb{Z}}}(\vec{\boldsymbol{\omega}}) \propto \sum_{i=1}^{\infty} \exp\left\{-\int_0^T \left[\frac{1}{2}(||\boldsymbol{\alpha}||^2 + \textbf{ div } \boldsymbol{\alpha})(\boldsymbol{\omega}_s) - l_i\right] ds\right\} I\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}, \tag{25}$$

Similarly as before, if we can simulate from $\widetilde{\mathbb{Z}}(\vec{\boldsymbol{\omega}})$ then we can simulate from $\widetilde{\mathbb{Q}}(\vec{\boldsymbol{\omega}})$ via rejection sampling.

To simulate from $\widetilde{\mathbb{Z}}(\vec{\boldsymbol{\omega}})$, we can first simulate $\boldsymbol{x}, \boldsymbol{y}$ from $h(\boldsymbol{x},\boldsymbol{y})$ and then conditional on $(\boldsymbol{x},\boldsymbol{y})$, we simulate from $\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}(\vec{\boldsymbol{\omega}})$ given by (23).

3.2.1. Simulation from $\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}(\vec{\boldsymbol{\omega}})$ given by (23)

Now the key step to be solved is to simulate from $\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{x},\boldsymbol{y}}(\vec{\boldsymbol{\omega}})$ given by (23). By observing (23), we know that its simulation can be achieved via the following two steps.

Step 1: we can first simulate the layer I according to the probability

$$\widetilde{\mathrm{P}}(I = i) = \frac{\exp\{-Tl_i\}\mathbb{W}_{0,T}^{x,y}\{\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\}}{\sum_{i=1}^{\infty} \exp\{-Tl_i\}\mathbb{W}_{0,T}^{x,y}\{\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\}}; \tag{26}$$

Step 2: then conditional on the layer I we simulate $\vec{\boldsymbol{\omega}}$ from $d\mathbb{W}_{0,T}^{x,y}(\vec{\boldsymbol{\omega}})I\{\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\}$ as follows.

Step 1

This step can be done using the same method in Dai (2013).

Step 2: Conditional on the layer I, simulation of $\vec{\boldsymbol{\omega}}$ from $d\mathbb{W}_{0,T}^{x,y}(\vec{\boldsymbol{\omega}})I\{\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\}$

First we need simulate the layers for each component. This is because for multi-dimensional case, the layer $I = i$ only tells us $\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i$, i.e. all the components $\boldsymbol{\omega}^{(j)}$ belong to $\cup_{k=1}^{i}\mathcal{D}_k^{(j)}$ and at least one component $\boldsymbol{\omega}^{(j')}$ belongs to $\mathcal{D}_i^{(j')}$, but we do not know which component it is. Given $I = i$, the simulation for the layers of each component can be done via the following approach. We simulate a uniform variable U from the interval $\left[\mathrm{P}\left(\vec{\boldsymbol{\omega}} \in \otimes_j \left[\cup_{k=1}^{i-1}\mathcal{D}_k^{(j)}\right]\right), \mathrm{P}\left(\vec{\boldsymbol{\omega}} \in \otimes_j \left[\cup_{k=1}^{i}\mathcal{D}_k^{(j)}\right]\right)\right]$, i.e. simulate U from

$$\left[\prod_j \mathrm{P}\left(\boldsymbol{\omega}^{(j)} \in \cup_{k=1}^{i-1}\mathcal{D}_k^{(j)}\right), \prod_j \mathrm{P}\left(\boldsymbol{\omega}^{(j)} \in \cup_{k=1}^{i}\mathcal{D}_k^{(j)}\right)\right].$$

Note that such U can be simulated, since the two boundary points of the above interval are limits of certain alternating sequences (Beskos et al., 2008).

This means U is simulated by conditioning on $\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i$. Then use this U to simulate the layer of each component $\boldsymbol{\omega}^{(j)}$. If U belongs to the interval $[U_l, U_r]$,

$$\begin{aligned} U_l &= \prod_{j=1}^{j'-1} \mathrm{P}(\boldsymbol{\omega}^{(j)} \in \cup_{k=1}^{i} \mathcal{D}_k^{(j)}) \cdot \prod_{j=j'}^{q} \mathrm{P}(\boldsymbol{\omega}^{(j)} \cup_{k=1}^{i-1} \mathcal{D}_k^{(j)}), \\ U_r &= \prod_{j=1}^{j'} \mathrm{P}(\boldsymbol{\omega}^{(j)} \in \cup_{k=1}^{i} \mathcal{D}_k^{(j)}) \cdot \prod_{j=j'+1}^{q} \mathrm{P}(\boldsymbol{\omega}^{(j)} \cup_{k=1}^{i-1} \mathcal{D}_k^{(j)}), \end{aligned}$$

then we have that

$$\begin{aligned} &\boldsymbol{\omega}^{(j')} \in \mathcal{D}_k^{(j')} \\ &\boldsymbol{\omega}^{(j)} \in \cup_{k=1}^{i} \mathcal{D}_k^{(j)} \text{ for } j < j' \\ &\boldsymbol{\omega}^{(j)} \in \cup_{k=1}^{i-1} \mathcal{D}_k^{(j)} \text{ for } j > j'. \end{aligned} \tag{27}$$

Second, once the boundaries for each component $\boldsymbol{\omega}^{(j)}$ has been sampled, we can simulate a Brownian bridge $\vec{\boldsymbol{\omega}}$ with $\boldsymbol{\omega}_0 = \boldsymbol{x}$ and $\boldsymbol{\omega}_T = \boldsymbol{y}$ by conditional on the boundaries using the method in (Beskos et al., 2008).

3.2.2. The algorithm improvement

Using the above method, such a simulated multi-dimensional process $\vec{\boldsymbol{\omega}} = (\boldsymbol{\omega}^{(1)}, \cdots, \boldsymbol{\omega}^{(q)})$ is drawn from $\widetilde{\mathbb{Z}}$. Then use the rejection sampling idea in Beskos et al. (2006), we can use a rejection sampling to accept $\vec{\boldsymbol{\omega}}$ as a process from $\bar{\mathbb{Q}}$ or to reject it. The acceptance-rejection ratio is given by (25).

In summary, an improved version of Algorithm 2 is given below, which is a subroutine to replace steps 8 and 9 in Algorithm 1.

```
1 Given $\widetilde{\boldsymbol{B}}_0 = \boldsymbol{x}$ and $\widetilde{\boldsymbol{B}}_T = \boldsymbol{y}$, simulate a process $\widetilde{\boldsymbol{B}}_t, t \in [0, T]$ via the following steps 1a
  and 1b:;
  // 1a:  Simulate layer $I$ with probability $\widetilde{\mathrm{P}}(I)$ given in (26) (Dai, 2013)
  // 1b:  Simulate the layers for each component (given by (27))
  // 1c:  Given $\widetilde{\boldsymbol{B}}_0$ and $\widetilde{\boldsymbol{B}}_T$, simulate a sample path $\widetilde{\boldsymbol{B}}_t, 0 < t < T$, from $\mathbb{W}_{0,T}^{x,y}$
      conditional on $\boldsymbol{\omega}^{(j')} \in \mathcal{D}_k^{(j')}, \boldsymbol{\omega}^{(j)} \in \cup_{k=1}^{i} \mathcal{D}_k^{(j)}$ for $j < j'$ and
      $\boldsymbol{\omega}^{(j)} \in \cup_{k=1}^{i-1} \mathcal{D}_k^{(j)}$ for $j > j'$, using the algorithm in Beskos et al. (2008)
2 Calculate $l_I$ such that $l_I \leq \inf_{s \in [0,T], \vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_I} \{(||\boldsymbol{\alpha}||^2 + \textbf{div } \boldsymbol{\alpha})(\boldsymbol{\omega}_s)/2\}$;
3 Calculate $r_I$ such that $r_I \geq \sup_{t \in [0,T], \vec{\boldsymbol{\omega}} \in \mathcal{D}_I} \{[||\boldsymbol{\alpha}||^2(\boldsymbol{\omega}_t) + \text{div } \boldsymbol{\alpha}'(\boldsymbol{\omega}_t)]/2 - l_I\}$;
4 Simulate $\boldsymbol{\Psi} = \{\psi_1, \cdots, \psi_\rho\}$ uniformly distributed on $\mathbf{U}[0, T]$ and marks
  $\Upsilon = \{\nu_1, \cdots, \nu_\rho\}$ uniformly distributed on $\mathbf{U}[0, 1]$, where $\rho$ is from $\mathrm{Poi}(r_I T)$;
5 Compute the acceptance indicator $\mathcal{I} := \prod_{j=1}^{\rho} I[r_I^{-1}[(||\boldsymbol{\alpha}||^2 + \textbf{div } \boldsymbol{\alpha})(\boldsymbol{\omega}_{\psi_j})/2 - l_I] < \nu_j]$;
```

Algorithm 3: Improved version of Algorithm 2: simulation for $\mathcal{I}$

3.2.3. Why such an improvement is important

Note that, based on the methods in Section 3.1 and Section 3.2, the new method is more efficient than existing rejection sampling methods when a good hat function for f is not readily available. The posterior of finite mixture models is not log-concave. It is nontrivial to find a good hat function for it by partitioning the support of the posterior into several subsets and finding a bound for each subsets. However, it is always possible to partition the space of a Brownian bridge into many different layers and then we can find lower bounds for each layer. This makes the new method to be practical for complicated target distributions.

3.3. Improvement for increasing AP_1

The algorithm simulates $\boldsymbol{\omega}_T$ from g_2 as a proposal. We can see that the proposal is more likely to be accepted if the distance $||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T||^2$ becomes smaller. Therefore, to increase the acceptance probability, we need to find a good decomposition, $f = g_1 \cdot g_2$, to make AP_1 as large as possible, where

$$\begin{aligned} AP_1 &= \mathrm{P}\{U \le \exp[-||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T||^2/(2T)]\} \\ &= \mathrm{P}\{||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T||^2 \le -2T\log(U)\} \end{aligned} \tag{28}$$

where $U \sim U[0,1]$, $\boldsymbol{\omega}_0 \sim g_1$, $\boldsymbol{\omega}_T \sim g_2$.

Note that it is nontrivial to find the best decomposition $f = g_1 \cdot g_2$ to achieve the maximum value of AP_1, since there are infinite decompositions. However, we can find the best one under a subset of all possible decompositions of f, which will provide us a direction of finding a good decomposition.

First we introduce the following notations: $E_{g_1} = \int \boldsymbol{x} g_1(\boldsymbol{x})d\boldsymbol{x}$ and $E_{g_2} = \int \boldsymbol{x} g_2(\boldsymbol{x})d\boldsymbol{x}$. Then we have the following lemma.

Lemma 3.3. *Define $\mathcal{A} = \{(g_1, g_2) :$ such that $f = g_1 \cdot g_2$ and $E_{g_1} = E_{g_2}\}$. Then for all $(g_1, g_2) \in \mathcal{A}$ and for independent variables $\boldsymbol{\omega}_0 \sim g_1(\cdot)$ and $\boldsymbol{\omega}_T \sim g_2(\cdot)$, the expectation $E||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_1||^2$ reaches the minimum when $g_1(\cdot) = g_2(\cdot) = \sqrt{f(\cdot)}$.*

Proof. We have

$$\begin{aligned} &E||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_1||^2 = E_{g_1}||\boldsymbol{\omega}_0||^2 + E_{g_2}||\boldsymbol{\omega}_T||^2 - 2\langle E_{g_1}, E_{g_2}\rangle \\ = &\int ||\boldsymbol{x} - E_{g_1}||^2 g_1(\boldsymbol{x})d\boldsymbol{x} + \int ||\boldsymbol{x} - E_{g_2}||^2 g_2(\boldsymbol{x})d\boldsymbol{x} + ||E_{g_1} - E_{g_2}||^2 \\ = &\int ||\boldsymbol{x} - E_{g_1}||^2 \left[g_1(\boldsymbol{x}) + \frac{f(\boldsymbol{x})}{g_1(\boldsymbol{x})}\right] d\boldsymbol{u} \end{aligned}$$

which reaches the minimum when $g_1(\boldsymbol{x}) = f(\boldsymbol{x})/g_1(\boldsymbol{x})$, i.e. $g_1 = g_2 = \sqrt{f}$. □

The above result implies that we should choose a decomposition to make g_1 and g_2 as close to each other as possible. Indeed, we find that this is true in our simulation studies for mixture models.

4. Rejection sampling for the general case $f = \prod_{l=1}^{\iota} g_l$

In general, the target density f may be decomposed as a product of ι terms, $f(\boldsymbol{x}) = \prod_{l=1}^{\iota} g_l$, where we can easily draw a sample from g_l. To draw a sample from f, we can use the following recursive algorithm. First we decompose f as $f = f_1 f_2$, where $f_1 = g_1^2, f_2 = g_1^{-1} \prod_{j=2}^{\iota} g_j$. To use Algorithm 1, we need to simulate from $\prod_{j=2}^{\iota} g_j$, which can be further decomposed as $g_2 \cdot \prod_{j=3}^{\iota} g_j$. Keep simplifying the target until it becomes $g_{\iota-1} \cdot g_\iota$. When running such a recursive algorithm, we actually do it in the reverse procedure, i.e. simulate samples from $\prod_{j=l}^{\iota} g_j$, l from $\iota - 1$ to 1. This is given by the following algorithm.

```
1  Simulate $\boldsymbol{y}$ from $g_\iota$;
2  for $l \leftarrow \iota - 1$ to 1 do
3      Simulate $\boldsymbol{x}_l$ from $g_l$;
4      Simulate standard uniform variable $U_l$;
5      if $U_l > \exp(-||\boldsymbol{x}_l - \boldsymbol{y}||^2/(2T))$ then
6          Goto Step 1;
7      end
8      Simulate the Brownian bridge $\bar{\boldsymbol{B}} = \{\boldsymbol{\omega}_t, t \in (0,T)\}$ given $(\boldsymbol{\omega}_0 = \boldsymbol{x}_l, \boldsymbol{\omega}_T = \boldsymbol{y})$;
9      Simulate $\mathcal{I}_l = 1$ with probability given by (13), with $\boldsymbol{\alpha}(\boldsymbol{x}) = \nabla A(\boldsymbol{x})$ and
       $A(\boldsymbol{x}) = \log g_l(\boldsymbol{x})$;
       // the above two steps can be improved using Algorithm 3
10     if $\mathcal{I}_l = 1$ then
11         $\boldsymbol{y}$ can be viewed as a sample from $\prod_{j=l}^{\iota} g_j$;
12     else
13         Goto Step 1;
14     end
15 end
16 Output $\boldsymbol{y}$.
```

Algorithm 4: Rejection sampling for $f = \prod_{l=1}^{\iota} g_\iota$.

Note that in the *for* loop of Algorithm 4, the code tries to draw a sample from $\prod_{j=l}^{\iota} g_j$. If a sample is successfully drawn from $\prod_{j=l}^{\iota} g_j$ then l decreases by 1; otherwise the algorithm goes back to the beginning since the proposal from $\prod_{j=l}^{\iota} g_j$ is rejected.

We can also see that Algorithm 4 simulates $\{\boldsymbol{x}_l\}_{l=1}^{\iota-1}$ and $\boldsymbol{y}$ independently. The proposal $\boldsymbol{y}$ will be accepted if $U_l \leq \exp(-||\boldsymbol{x}_l - \boldsymbol{y}||^2/(2T))$ and $\mathcal{I}_l = 1$ for $l = 1, \cdots, \iota - 1$. Since simulating the event $\mathcal{I}_l = 1$ using Beskos et al. (2006, 2008) or the more efficient Algorithm 3 is usually complicated and time consuming, we can revise Algorithm 4 as follows to increase the efficiency: First, simulate $\boldsymbol{x}_l, l = 1, \cdots, \iota - 1$ and $\boldsymbol{y}$; second, check if $U_l \leq \exp(-||\boldsymbol{x}_l - \boldsymbol{y}||^2/(2T))$ for $l = 1, \cdots, \iota - 1$; third, simulate $I_l = 1$, for $l = 1, \cdots, \iota - 1$. The revised algorithm is given below.

Note that Algorithm 4 and Algorithm 5 can be improved via the methods in Section 3.

```
1  Simulate $\boldsymbol{y}$ from $g_{(\iota)}$;
2  for $l \leftarrow \iota - 1$ to 1 do
3      Simulate $\boldsymbol{x}_l$ from $g_{(l)}$;
4      Simulate standard uniform variable $U_l$;
5      if $U_l > \exp(-||\boldsymbol{x}_l - \boldsymbol{y}||^2/(2T))$ then
6          Goto Step 1
7      end
8  end
9  for $l \leftarrow \iota - 1$ to 1 do
10     Simulate the Brownian bridge $\bar{\boldsymbol{B}} = \{\boldsymbol{\omega}_t, t \in (0, T)\}$ given $(\boldsymbol{\omega}_0 = \boldsymbol{x}_l, \boldsymbol{\omega}_T = \boldsymbol{y})$;
11     Simulate $\mathcal{I}_l = 1$ with probability given by (13), with $\boldsymbol{\alpha}(\boldsymbol{x}) = \nabla A(\boldsymbol{x})$ and
       $A(\boldsymbol{x}) = \log g_{(l)}(\boldsymbol{x})$;
       // the above two steps can be improved using Algorithm 3
12     if $\mathcal{I}_l = 0$ then
13         Goto Step 1;
14     end
15 end
16 Output $\boldsymbol{y}$;
```

Algorithm 5: Revised rejection sampling for $f = \prod_{l=1}^{\iota} g_\iota$.

5. Exact Monte Carlo simulation for finite mixture models

We here focus on the mixture of normal densities. The method developed here can be easily extended to the mixture of other density functions. Suppose that the data $\{z_i, i = 1, \cdots, n\}$ are from the finite mixture of normal densities

$$h(z_i; \boldsymbol{\Theta}) = \sum_{k=1}^{K} p_k h_k(z_i; \theta_k, \nu), \; h_k(z_i; \theta_k, \nu) = |\nu| e^{-\frac{\nu^2}{2}(z_i - \theta_k)^2} \tag{29}$$

with $\boldsymbol{\Theta} = (\boldsymbol{p}, \nu, \boldsymbol{\theta})$, where ν and θ_k range in $\mathbf{R}$ and p_k is the component proportion with $0 \le p_k \le 1$ and $\sum_k p_k = 1$. When K is unknown, the methods in Richardson and Green (1997) and Stephens (2000b) can draw approximate samples from the posterior of the mixture model in (29) via MCMC, for which the diagnostic of convergence for the Markov chains may not be easy since the label switching problem makes it difficult to confirm the convergence of the Markov chain. This makes it extremely challenging to draw exact realisations from the posterior with K unknown. Therefore here we assume K is known and leave the exact simulation for unknown K as a future work.

The Dirichlet distribution for $\boldsymbol{p}$ and the normal-gamma distribution for $(\boldsymbol{\theta}, |\nu|)$ are widely used as the prior for $\boldsymbol{\Theta}$. They are given by

$$\pi_0(\boldsymbol{\Theta}) \propto |\nu|^{2a-1} e^{-b\nu^2} |\nu|^K \prod_{k=1}^{K} \left[e^{-\frac{\sigma_k \nu^2}{2}(\theta_k - \mu_k)^2} p_k^{\varrho_k - 1} \right], \tag{30}$$

where $(\sigma_k, \mu_k, \varrho_k, a, b)$ is known. We focus on (30) in this paper for simplicity as it is conjugate to the mixture components, though other choices of prior based on reparameterisations of mixture models are available (Mengersen and Robert, 1996). Based on (30), the posterior distribution can be written as

$$f(\boldsymbol{\Theta}) \propto \prod_{i=1}^{n} \left[\sum_{k=1}^{K} p_k h_k(z_i; \boldsymbol{\Theta}) \right] \pi_0(\boldsymbol{\Theta}). \tag{31}$$

Traditional methods treat the mixture models in a latent variable framework (Diebolt and Robert, 1994) by assuming that there is a vector of i.i.d. latent variables $\boldsymbol{\xi} = (\xi_1, \cdots, \xi_n)$, each element of which has the discrete distribution given by $P(\xi_i = k) = p_k$, and such that the conditional density of z_i is $h_k(z_i; \theta_k, \nu)$. Then Gibbs sampling method can be applied to sampling approximately from (31). Traditional MCMC methods, however, are not satisfactory for mixture models due to the challenge in diagnostic of the convergence of the Markov chain.

In recent years, various perfect sampling methods, which draw realisations exactly from the target distribution, have been proposed for Bayesian analysis of mixture models. These methods include: for the simple mixture models with components all having known parameters, the CFTP method in Hobert et. al. (1999), the direct sampling method in Fearnhead (2005), the 'catalytic perfect sampling' in Breyer and Roberts (2001) and the GAM method in Dai (2007); for mixture models with components involving unknown parameters, the CFTP method in Casella et al. (2002), the direct simulation method in Fearnhead (2005). However, all above methods are either only valid for specific type of mixture models or not practical. Therefore there is a demand of developing new perfect simulation method for the posterior of mixture models. In this section, we will demonstrate how to use the proposed method in early sections to draw realisations exactly from the posterior of normal mixture models.

5.1. Reparameterisation and density function decomposition

Consider the following transformation $\boldsymbol{\Theta} = \boldsymbol{\Theta}(\boldsymbol{x})$, where $\boldsymbol{x} = (\boldsymbol{u}, \eta, \boldsymbol{\delta})$,

$$\begin{aligned} p_k &:= p_k(\boldsymbol{x}) = \frac{e^{u_k}}{\sum_{k=1}^{K-1} e^{u_k} + 1}, \ k = 1, \cdots, K-1, \\ \nu &:= \nu(\boldsymbol{x}) = \eta, \\ \theta_k &:= \theta_k(\boldsymbol{x}) = \delta_k \eta^{-1}, \ k = 1, \cdots, K. \end{aligned} \tag{32}$$

We consider such a transformation to make the support of the posterior $f(\boldsymbol{x})$ to be $\mathbf{R}^q$.

Then using the change-of-variable formula and with the definition

$$\hbar_k(z_i; \boldsymbol{x}) = e^{-\frac{1}{2}(z_i\eta - \delta_k)^2}, \ \hbar(z_i; \boldsymbol{x}) = \sum_{k=1}^{K} p_k(\boldsymbol{x}) \hbar_k(z_i; \boldsymbol{x}) \tag{33}$$

the posterior distribution becomes

$$\begin{aligned} f(\boldsymbol{x}) &\propto |\eta|^n \prod_{i=1}^{n} \left[\sum_{k=1}^{K} p_k(\boldsymbol{x}) \hbar_k(z_i; \boldsymbol{x}) \right] \pi_0(\boldsymbol{\Theta}(\boldsymbol{x})) \cdot J(\boldsymbol{x}) \qquad (34) \\ &= |\eta|^n \prod_{i=1}^{n} \left[\sum_{k=1}^{K} p_k(\boldsymbol{x}) e^{-\frac{1}{2}(z_i\eta - \delta_k)^2} \right] \left\{ |\eta|^{2a-1} e^{-b\eta^2} \left[\prod_{k=1}^{K} e^{-\frac{\sigma_k}{2}(\delta_k - \eta\mu_k)^2} p_k(\boldsymbol{x})^{\varrho_k} \right] \right\} \end{aligned}$$

where the Jacobin $J(\boldsymbol{x}) = \left[\prod_{k=1}^{K} p_k(\boldsymbol{x}) \right] \cdot |\eta|^{-K}$ (See Appendix A).

5.1.1. A decomposition of f, which only works theoretically

To use the proposed method, one may consider to the following decomposition to a product of two terms,

$$\begin{aligned} f(\boldsymbol{x}) \;\propto\; & \prod_{i=1}^{n'} \left[\sum_{k=1}^{K} p_k(\boldsymbol{x}) e^{-\frac{1}{2}(z_i\eta-\delta_k)^2} \right] \left\{ e^{-b\eta^2} \left[\prod_{k=1}^{K} e^{-\frac{\sigma_k}{2}(\delta_k-\eta\mu_k)^2} p_k(\boldsymbol{x})^{\varrho_k} \right] \right\} \\ & \cdot |\eta|^{n+2a-1} \prod_{i=n'+1}^{n} \left[\sum_{k=1}^{K} p_k(\boldsymbol{x}) e^{-\frac{1}{2}(z_i\eta-\delta_k)^2} \right] \left\{ e^{-b\eta^2} \left[\prod_{k=1}^{K} e^{-\frac{\sigma_k}{2}(\delta_k-\eta\mu_k)^2} p_k(\boldsymbol{x})^{\varrho_k} \right] \right\}. \end{aligned} \tag{35}$$

The first term can be viewed as g_1 and the second term can be viewed as g_2. We here put all the non-differentiable terms, related to $|\eta|$, into g_2 to guarantee that the log-transformation of the first term is differentiable. Although we can simulate $\boldsymbol{x}$ from g_1 and $\boldsymbol{y}$ from g_2, such a decomposition will not work. The reason is that g_1 is not close to g_2 and the simulated $\boldsymbol{x}$ and $\boldsymbol{y}$ are almost always far away from each other. This makes the probability AP_1 very small. For this reason we consider the following decomposition.

5.1.2. A practical decomposition for a hat function of f

We need to guarantee that the log-transformation of g_1 is differentiable (Condition 2.1) and that g_1 and g_2 are similar (for large AP_1). In order to make g_1 and g_2 similar we need to put the term $|\eta|^{(n+2a-1)/2}$ into g_1 and into g_2 as well, but this will make $\log g_1$ not differentiable. This makes it non-trivial to find a good decomposition for f. Therefore, to draw samples from (34), we consider the following hat function

$$\hat{f}(\boldsymbol{x}) \;\propto\; (\eta^2+c)^{\frac{n+2a-1}{2}} \prod_{i=1}^{n} \left[\sum_{k=1}^{K} p_k(\boldsymbol{x}) e^{-\frac{1}{2}(z_i\eta-\delta_k)^2} \right] \left\{ e^{-b\eta^2} \left[\prod_{k=1}^{K} e^{-\frac{\sigma_k}{2}(\delta_k-\eta\mu_k)^2} p_k(\boldsymbol{x})^{\varrho_k} \right] \right\}$$

for some value $c > 0$. We can always choose a very small value of c to make $\hat{f}$ similar to f. Clearly $\hat{f}$ always bounds f and its log-transformation is differentiable.

Let $0 = n_0 < n_1 < \cdots < n_\iota = n$ be a sequence of positive integers. Let $m_l = n_l - n_{l-1}$ for $l = 1, \cdots, \iota$. Then the posterior density can be decomposed as a product of ι terms, $\hat{f}(\boldsymbol{x}) \propto \prod_{l=1}^{\iota} g_l(\boldsymbol{x})$, with

$$\begin{aligned} g_l(\boldsymbol{x}) \;=\; & (\eta^2+c)^{\frac{m_l}{2}+\frac{m_l}{2n}(2a-1)} \left[\prod_{i=n_{l-1}+1}^{n_l} \hbar(z_i;\boldsymbol{x}) \right] \left\{ e^{-\frac{m_l}{n}b\eta^2} \prod_{k=1}^{K} e^{-\frac{\sigma_k m_l}{2n}(\delta_k-\eta\mu_k)^2} p_k(\boldsymbol{x})^{\frac{m_l\varrho_k}{n}} \right\}, \\ & l = 1, \cdots, \iota. \end{aligned} \tag{36}$$

5.2. Simulation from g_l, a density based on the observations in the lth group

Let ξ_i be the latent allocation variables for the mixture model. Define the following statistics: the number of observations in group l from component k,

$$\breve{n}_{l,k} = \sum_{i=n_{l-1}+1}^{n_l} I[\xi_i = k];$$

the first sample moment for the observations in group j from component k,

$$\bar{Z}_{l,k} = \breve{n}_{l,k}^{-1} \sum_{i=n_{l-1}+1}^{n_l} I[\xi_i = k] z_i;$$

and

$$Z_l^2 = \sum_{i=n_{l-1}+1}^{n_l} z_i^2.$$

Then we can further write

$$\begin{aligned} g_l(\boldsymbol{x}) \;=\;& (\eta^2 + c)^{\frac{m_l}{2} + \frac{m_l}{2n}(2a-1)} e^{-\frac{m_l}{n} b\eta^2} \\ & \cdot \sum_{\breve{\boldsymbol{n}}_l, \bar{\boldsymbol{Z}}_l, Z_l^2} \left\{ \prod_{k=1}^{K} p_k(\boldsymbol{x})^{\breve{n}_{l,k} + \frac{m_j \varrho_k}{n}} \exp\left[-\frac{1}{2} \sum_k \left(\breve{n}_{l,k} + \frac{\sigma_k m_l}{n} \right) \left[\delta_k - \frac{\breve{n}_{l,k} \bar{Z}_{l,k} + \frac{\sigma_k m_l \mu_k}{n}}{\breve{n}_{l,k} + \frac{\sigma_k m_l}{n}} \eta \right]^2 \right] \right. \\ & \left. \cdot \exp\left[-\frac{1}{2} \left(Z_l^2 + \sum_k \left(\frac{\sigma_k m_l}{n} \mu_k^2 - \frac{\left(\breve{n}_{l,k} \bar{Z}_{l,k} + \frac{\sigma_k m_l \mu_k}{n} \right)^2}{\breve{n}_{l,k} + \frac{\sigma_k m_l}{n}} \right) \right) \eta^2 \right] \right\}. \end{aligned} \tag{37}$$

Note that we can simulate from (37) directly when K and m_j are small, i.e. when the number of statistics $(\breve{\boldsymbol{n}}_l, \bar{\boldsymbol{Z}}_l, Z_l^2)$ is not large. For example, when the mixture model has $K = 2$ components and $m_l = 25$, the number of different $(\breve{\boldsymbol{n}}_l, \bar{\boldsymbol{Z}}_l, Z_l^2)$ is about $2^{25} = 33554432$ terms, which can be dealt with by a standard desktop.

To simulate $\boldsymbol{x}$ from (37), the only challenge is to do the integration for η over $g_l(\boldsymbol{x})$. This can be done via a recursive approach. The method of simulation from g_l is given in the supplementary file.

5.3. *Simulate diffusions with invariant distribution* $g_l(\boldsymbol{x})$

To use Algorithm 5, we also need to simulate the multivariate diffusion, having g_l as the invariant distribution,

$$d\boldsymbol{X}_t^{(l)} = \boldsymbol{\alpha}^{(l)}(\boldsymbol{X}_t^{(l)})dt + d\boldsymbol{B}_t^{(l)}, \tag{38}$$

where $\boldsymbol{\alpha}^{(l)}(\boldsymbol{x}) = \nabla A^{(l)}(\boldsymbol{x})$ and $A^{(l)}(\boldsymbol{x}) = \log g_l(\boldsymbol{x})$.

Note that from the definition of $\boldsymbol{x} = (\boldsymbol{u}, \eta, \boldsymbol{\delta})$, we have that the vector function $\boldsymbol{\alpha}^{(l)}(\boldsymbol{x}) = (\alpha_{u_1}^{(l)}(\boldsymbol{x}), \cdots, \alpha_{u_{K-1}}^{(l)}(\boldsymbol{x}), \alpha_\eta^{(l)}(\boldsymbol{x}), \alpha_{\delta_1}^{(l)}(\boldsymbol{x}), \cdots, \alpha_{\delta_K}^{(l)}(\boldsymbol{x}))$ is given by

$$\begin{aligned} \alpha_{u_k}^{(l)}(\boldsymbol{x}) &= \frac{\partial \log g_l(\boldsymbol{x})}{\partial u_k} = \frac{\partial g_l(\boldsymbol{x})/\partial u_k}{g_l(\boldsymbol{x})}, \\ \alpha_{\eta}^{(l)}(\boldsymbol{x}) &= \frac{\partial \log g_l(\boldsymbol{x})}{\partial \eta} = \frac{\partial g_l(\boldsymbol{x})/\partial \eta}{g_l(\boldsymbol{x})}, \\ \alpha_{\delta_k}^{(l)}(\boldsymbol{x}) &= \frac{\partial \log g_l(\boldsymbol{x})}{\partial \delta_k} = \frac{\partial g_l(\boldsymbol{x})/\partial \delta_k}{g_l(z_i; \boldsymbol{x})}. \end{aligned} \tag{39}$$

According to

$$\text{div } \boldsymbol{\alpha}^{(l)}(\boldsymbol{x}) = \sum_{k=1}^{K-1} \frac{\partial \alpha_{u_k}^{(l)}(\boldsymbol{x})}{\partial u_k} + \frac{\partial \alpha_\eta^{(l)}(\boldsymbol{x})}{\partial \eta} + \sum_{k=1}^{K} \frac{\partial \alpha_{\delta_k}^{(l)}(\boldsymbol{x})}{\partial \delta_k} \tag{40}$$

we further have

$$||\boldsymbol{\alpha}^{(l)}(\boldsymbol{x})||^2 + \text{div } \boldsymbol{\alpha}^{(l)}(\boldsymbol{x}) = \sum_{k=1}^{K-1} \frac{\partial^2 g_l(\boldsymbol{x})/\partial u_k^2}{g_l(\boldsymbol{x})} + \frac{\partial^2 g_l(\boldsymbol{x})/\partial \eta^2}{g_l(\boldsymbol{x})} + \sum_{k=1}^{K} \frac{\partial^2 g_l(\boldsymbol{x})/\partial \delta_k^2}{g_l(\boldsymbol{x})} \tag{41}$$

whose expression can be found in Appendix B. We also show that (41) is bounded below in Appendix B.

Note that we also need to find the upper bound (required by Algorithm 2) and lower bounds (required by (24)) for $||\boldsymbol{\alpha}(\boldsymbol{\omega}_t)||^2 + \text{div } \boldsymbol{\alpha}(\boldsymbol{\omega}_s)$ under each layer $\boldsymbol{\mathcal{Q}}_i$. This is also straightforward and the details are provided in the supplementary file.

5.4. Simulation results for mixture models

5.4.1. Justification of the correctness of the new algorithm

We consider a mixture model with 2 components, $h(z_i; \boldsymbol{\Theta}) = \sum_{k=1}^{2} p_k \mathcal{N}(\theta_i, \nu^{-2})$, with the means $\theta_1 = 1.0$, $\theta_2 = 0.0$, $p_1 = 0.6$ and the variance $\nu^{-2} = 0.2^2$. We consider a small sample size $n = 20$ since when $n = 20$ we can easily sample directly from the posterior distribution. The results of using direct simulation can be compared with the results of the new method and we can then justify the correctness of the proposed algorithm. We use the prior distribution in (30) with $a = 1.5$, $b = 1.0$, $\sigma_1 = \sigma_2 = 1$, $\mu_1 = 1.0, \mu_2 = 0.0$ and $\rho_1 = \rho_2 = 2.0$.

To use the new method, we partition the 20 samples into two groups. By doing this, the hat function $\hat{f}$ of the posterior can be decomposed into a product of $g_1 g_2$ where g_k is a density based on the kth group of data (see (36) for example). We partition the samples into two groups in the following way: first we order the samples to $z_{(1)}, z_{(2)}, \cdots, z_{(19)}, z_{(20)}$ and then the first group is $\{z_{(2k-1)}, k = 1 \cdots, 10\}$ and the second group is $\{z_{(2k)}, k = 1, \cdots 10\}$. By doing this, the two functions g_1 and g_2 will be similar and this can increase the acceptance probability AP_1. See Lemma 3.3 and the arguments in Section 3.3. This is also demonstrated by the simulation results in Section 5.4.2, where we found that the algorithm would not work if we simply randomly allocate the samples into two groups but it works well if we do the sample allocation as above.

For the hat function in (36) we choose $c = 0.05$. and a layer value $a_i = 0.1$ and $T = 0.03$.

For both methods, the proposed new method and the direct simulation method, we simulate 5000 realisations. Then we plot the marginal empirical distribution functions for each parameter, based on the two simulation methods. The results are shown in Figure 2. We can see that the new method and the direct simulation method output almost identical empirical distributions. Note that the outputs are based on raw realisations simulated from the posterior distribution and label switching is not considered here. Stephens (1997) and Stephens (2000a) can be used to deal with the label switching problem. We here did not consider it as this is not a main aim of this paper.

5.4.2. Running time comparisons; two-component mixture models and $n = 40$

For the same model and priors as that in Section 5.4.1, now we choose sample size $n = 40$ and compare the running times taken by the algorithm under different grouping of samplings and under different choices of T, c and a_i, which are all parameters governing the efficiency of the algorithms.

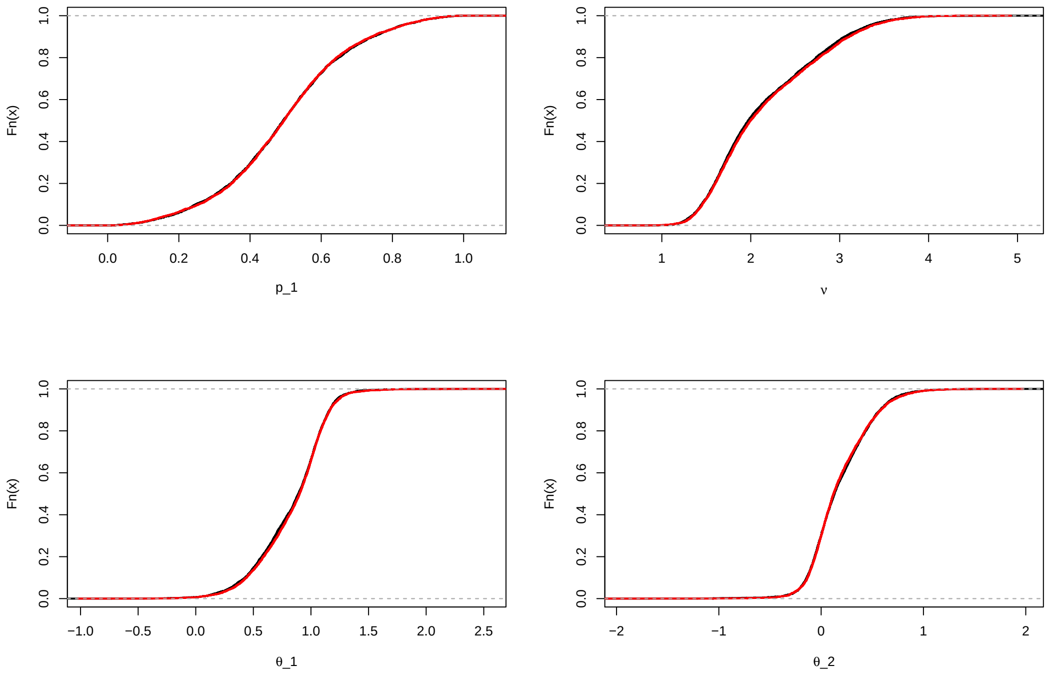

Fig. 2. Comparison with direct sampling; two methods output almost the same results

Table 1. Running times in seconds for simulation of one realisation from the posterior and acceptance probabilities: (i) AP_1 and (ii) AP_2; $c = 0.05$ which gives $AP_3 \approx 0.8$

$c = 0.05$	$T = 0.01$	$T = 0.02$	$T = 0.03$
(i) AP_1, time	7.8e-5	3.0e-4	0.0006
$a_i = 0.15$	244s	246s	284s
$a_i = 0.15$, (ii)	0.821	0.421	0.331
$a_i = 0.60$	337s	310s	474s
$a_i = 0.60$, (ii)	0.738	0.409	0.268

Running time comparisons under for different sample partitions
To use the new method, we partition the 40 samples into two groups in the following way: first we order the samples and then the first group is the ordered statistics with odd ranks and the second group is the ordered statistics with even ranks. Clearly such a partition will make the two samples very similar and thus make g_1 and g_2 similar. We therefore suggest such a partition of samples based on ordered statistics as a general approach. Without doing this (for example just randomly allocate samples into two groups) we found that the new algorithm will not work due to low acceptance probability AP_1.

Running time comparisons under different values of c
Note that when dealing with mixture model, we actually use the proposed algorithm to sample from $\hat{f}$ first and then use acceptance/rejection sampling method to decide whether the proposal is a sample from f. Therefore there is an extra acceptance/rejection step involved here. Suppose that the acceptance probability, for $\boldsymbol{x} \sim \hat{f}$ in (36) to be accepted as $\boldsymbol{x} \sim f$ in (34), is denoted as AP_3. The value c is used in (36) governs the acceptance probability AP_3 and the smaller value of c the larger AP_3. However, it does not mean an algorithm with smaller values of c will be more efficient. From the lower bound (47) in Appendix B, we can see that the smaller values of c, the smaller lower bound for $(||\boldsymbol{\alpha}||^2 + \boldsymbol{\alpha}')(\cdot)$ i.e. the smaller acceptance probability AP_2. This is shown by the simulation results summarised in Table 1 and Table 2. By comparing the results in both tables, we can see that it is more efficient to choose $c = 0.05$ than to choose $c = 0.03$, for all different choices of a_i and T.

Note that AP_1 will be larger if g_1 and g_2 have smaller variation. The value $c = 0.03$ gives smaller variances for g_1 and g_2 therefore we expect that AP_1 should be larger with $c = 0.03$. This is confirmed by the simulation results: The acceptance probability AP_1 slightly increases by changing $c = 0.05$ to $c = 0.03$.

Running time comparisons under different values of T
As we discussed in early sections, the value of T is very important for the efficiency of the algorithm. In practice, we can always roughly estimate a value of T. To do this, we need to roughly estimate the order of $(||\boldsymbol{\alpha}||^2 + \boldsymbol{\alpha}')(\boldsymbol{\omega}_t)/2 - l_i$, which actually depends on the sample size n, the layer parameter a_i and the value c. In some pilot simulation studies, we found that the value of this function is roughly between 60 and 65 with $a_i = 0.15$ and $c = 0.03$. This means that if we choose value T around $(0.01 \sim 0.03)$, the acceptance probability AP_2 is roughly around $[\exp(-65 \cdot 0.03) = 0.14, \exp(-65 \cdot 0.01) = 0.52]$ (or an even larger value), which is not a very tiny

Table 2. Running times in seconds for simulation of one realisation from the posterior and acceptance probabilities:: (i) AP_1 and (ii) AP_2; $c = 0.03$ which gives $AP_3 \approx 0.9$

$c = 0.03$	$T = 0.01$	$T = 0.02$	$T = 0.03$
(i) AP_1	8.3e-5	3.5e-4	0.0007
$a_i = 0.15$, time	284s	275s	503s
$a_i = 0.15$, (ii)	0.656	0.357	0.211
$a_i = 0.60$, time	355s	375s	589s
$a_i = 0.60$, (ii)	0.564	0.334	0.171

acceptance probability. This is confirmed by the acceptance probability estimates (based on the Monte Carlo simulations) in tables 1 and 2.

If we choose $T = 1$, the algorithm never returns a value in a realistic time period, since AP_2 is too small. On the other hand, the efficiency of the algorithm also depends on AP_1, which will be very tiny if T is very small. For example, if we choose $T = 0.001$, the algorithm is not efficient either, since AP_1 is too small. We choose T ranges from 0.01 to 0.03 in our simulation studies, as it makes AP_1 and AP_2 both in an acceptable range. In our simulation studies, we found that with $T = 0.02$ the algorithm is the most efficient.

Running time comparisons under for different values of a_i

The value of a_i is used to defined the layers for the layered Brownian motion. Theoretically, the smaller value of a_i will give a larger acceptance probability AP_2. We compare the simulation results under two scenarios $a_i = 0.15$ and $a_i = 0.60$. The simulation results in tables 1 and 2 confirm that smaller values of a_i give larger acceptance probability AP_2. However, as Dai (2013) pointed out, to sample a reweighted layered Brownian motion the algorithm will do a search from layer 1 to the layer to be sampled. Suppose that when choose $a_i = 0.6$, the algorithm is likely to sample a layer value, say $I = 4$. Then if we choose $a_i = 0.15$, the algorithm will be likely to sample a layer value ranges from $I = 13$ to $I = 16$. Clearly the algorithm takes more time to search until finding the target layer. Therefore, choosing very tiny values for a_i will not be a good choice.

5.4.3. The challenges: mixture models with more than two components or with larger sample sizes

The new method still works, although the running time is much longer, if the mixture model has two components and $n = 60$, for which no existing methods can handle. When $n = 60$, we cannot partition the samples into two groups (each having thirty samples) since most existing desktops cannot deal with the expanded posterior for a 2 component mixture with 2^{30} terms. However, the new method in Section 4 works if we partition the samples into three groups (each has 20 samples). If we consider the same mixture model and priors used in the previous subsection and if we choose $T = 0.02$, $a_i = 0.15$ and $c = 0.05$, Algorithm 5 (together with Algorithm 3) takes 8.7e+4 seconds (24 hours) to draw a single sample from the posterior.

5.4.4. Summary for simulations
In our simulation studies, we try to choose the values of T, a_i and c to give a large acceptance probability of AP_2 (about 0.5), which means having a low acceptance probability of AP_1 (only about 0.0001). This is because it usually takes a long time to sample the proposal diffusion processes (then decide whether it should be accepted; AP_2 is involved), but it is easy to sample $\boldsymbol{\omega}_0 \sim g_1$ and $\boldsymbol{\omega}_T \sim g_2$ (then decide whether they should be accepted; AP_1 is involved).

The proposed method still cannot efficiently handle a mixture with more than 2 components or a 2-component mixture model with more than 60 samples. The reason is that in such a case both AP_1 and AP_2 will be very small. Therefore although it is more efficient than existing method, it is still far from being practical. The new method, however, surely provides a possible direction and brings new insights for drawing exact realisations from the posterior of mixture models. In the next section, we will discuss the possible directions of improving the proposed method.

6. Discussion

This paper proposes a new rejection sampling method, which does not require a hat function to bound the target function f but to decompose f into a product of density functions which are easy to simulate from. The new method is more efficient than existing rejection sampling methods when a good hat function for f is not readily available. The new method proposed in this paper transfers the difficulty of finding the hat function to finding the lower bounds of $||\boldsymbol{\alpha}||^2(\boldsymbol{\omega}_s) + \boldsymbol{\alpha}'(\boldsymbol{\omega}_s)$. We can always partition the space of Brownian bridges into many layers and find the lower bound for each layer, which makes the new method practical for complicated target distributions.

In practice, many complicated distribution densities may not have support in $\mathbf{R}^q$ which is required by the new method, but we can usually find a transformation and use change-of-variable formula to obtain the new target density with support in $\mathbf{R}^q$. We achieve this even for the very complicated posterior of the mixture of normal densities. Therefore, such a constraint will not limit the application of the method.

The new method brings new insights for rejection sampling and coupling from the past, but it still faces many challenges which could limit its applications. The main challenge is that to simulate the starting and ending points $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$ from $h(\cdot,\cdot)$, we have to simulate $\boldsymbol{\omega}_0 \sim g_1$ and $\boldsymbol{\omega}_T \sim g_2$ and accept $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$ with probability $\exp(-||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T||^2/(2T))$. This acceptance probability will be small if we choose small value of T. To solve this problem, we have to choose a larger value of T, but large T will make the acceptance probability AP_2 very small. Although we proposed a method to partition the space of Brownian bridges into many layers, which increase AP_2 significantly, it is still not practical to run an algorithm with large T. Therefore there is a demand to develop an efficient exact simulation algorithm for diffusion bridges with large T. We are currently working on this.

When sampling from the posterior of finite mixture model, we actually applied the new method to the hat function (36), since the log-transformation of the target function is not differentiable. By using the hat function (36) Condition 2.1 is satisfied, but we need an extra acceptance/rejection step to draw a sample from the target function (34). Therefore to improve the efficiency of the algorithm for the posterior of mixture models, it is important to develop a new algorithm for exact simulation of diffusions with piecewise differentiable drift coefficient $\boldsymbol{\alpha}$. If such an method is available, we can apply the new method directly on the target distribution (34). We also leave this as a future work.

A. Derivation of $J(\boldsymbol{X})$ in (34)

We have

$$\begin{vmatrix} \dfrac{\partial p_1}{\partial u_1} & \cdots & \dfrac{\partial p_1}{\partial u_{K-1}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial p_{K-1}}{\partial u_1} & \cdots & \dfrac{\partial p_{K-1}}{\partial u_{K-1}} \end{vmatrix} = \begin{vmatrix} p_1 - p_1^2 & -p_1p_2 & \cdots & -p_1p_{K-1} \\ -p_1p_2 & p_2 - p_2^2 & \cdots & -p_2p_{K-1} \\ \vdots & \vdots & \ddots & \vdots \\ -p_1p_{K-1} & -p_2p_{K-1} & \cdots & p_{K-1} - p_{K-1}^2 \end{vmatrix} = \prod_{k=1}^{K} p_k(\boldsymbol{u}) \quad (42)$$

since according to (32) we have

$$\begin{aligned} \partial p_k/\partial u_k &= p_k - p_k^2, \\ \partial p_j/\partial u_k &= -p_jp_k, j \neq k. \end{aligned} \quad (43)$$

Therefore it is obvious that

$$J(\boldsymbol{X}) = \prod_{k=1}^{K} [p_k(\boldsymbol{u})]\, |\eta|^{-K}.$$

B. Lower bound for (41)

B.1. Lower bound for $\sum_{k=1}^{K-1} \dfrac{\partial^2 g_l(\boldsymbol{x})/\partial u_k^2}{g_l(\boldsymbol{x})}$

From the formula (36), we have

$$\partial g_l(\boldsymbol{x})/\partial u_k = g_l(\boldsymbol{x}) \left[\sum_{i=n_{l-1}+1}^{n_l} \hbar(z_i;\boldsymbol{x})^{-1} \frac{\partial \hbar(z_i;\boldsymbol{x})}{\partial u_k} + \sum_{\upsilon=1}^{K} p_\upsilon(\boldsymbol{x})^{-\frac{m_l \varrho_\upsilon}{n}} \frac{\partial \left(p_\upsilon(\boldsymbol{x})^{\frac{m_l\varrho_\upsilon}{n}}\right)}{\partial u_k} \right].$$

Using the results in Appendix A, we have $\frac{\partial \left(p_\upsilon(\boldsymbol{x})^{\frac{m_l\varrho_\upsilon}{n}}\right)}{\partial u_k} = \frac{m_j\varrho_\upsilon}{n}\left(p_\upsilon(\boldsymbol{x})^{\frac{m_l\varrho_\upsilon}{n}}\right)(1-p_\upsilon)$, if $\upsilon = k$; and $\frac{\partial \left(p_\upsilon(\boldsymbol{x})^{\frac{m_l\varrho_\upsilon}{n}}\right)}{\partial u_k} = -\frac{m_j\varrho_\upsilon}{n}\left(p_\upsilon(\boldsymbol{x})^{\frac{m_l\varrho_\upsilon}{n}}\right)p_k$, if $\upsilon \neq k$. We also have $\frac{\partial \hbar(z_i;\boldsymbol{x})}{\partial u_k} = p_k(\boldsymbol{x})\hbar_k(z_i;\boldsymbol{x}) - p_k(\boldsymbol{x})\hbar(z_i;\boldsymbol{x})$. Therefore, we can further write $\partial g_l(\boldsymbol{x})/\partial u_k$ as

$$\partial g_l(\boldsymbol{x})/\partial u_k = g_l(\boldsymbol{x}) \left[\sum_{i=n_{l-1}+1}^{n_l} \left(\frac{p_k(\boldsymbol{x})\hbar_k(z_i;\boldsymbol{x})}{\hbar(z_i;\boldsymbol{x})} - p_k(\boldsymbol{x}) \right) + \left(\frac{m_l\varrho_k}{n} - \sum_{\upsilon=1}^{K} \frac{m_l\varrho_\upsilon}{n} p_\upsilon(\boldsymbol{x}) \right) \right].$$

Then we have the following equation, where for simplicity we denote $p_k := p_k(\boldsymbol{x})$ and $\hbar_k(z_i) := \hbar_k(z_i;\boldsymbol{x})$ and $\hbar(z_i) := \hbar(z_i,\boldsymbol{x})$,

$$\begin{aligned}
&\partial^2 g_l(\boldsymbol{x})/\partial u_k^2 \\
= \;& g_l(\boldsymbol{x})\left[\sum_{i=n_{l-1}+1}^{n_l}\left(\frac{p_k\hbar_k(z_i)}{\hbar(z_i)} - p_k\right) + \left(\frac{m_l\varrho_k}{n} - \sum_{\upsilon=1}^{K}\frac{m_l\varrho_\upsilon}{n}p_\upsilon\right)\right]^2 \\
&+g_l(\boldsymbol{x})\left[\sum_{i=n_{l-1}+1}^{n_l}\left(\frac{(p_k-p_k^2)\hbar_k(z_i)\hbar(z_i) - p_k\hbar_k(z_i)(p_k\hbar_k - p_k\hbar(z_i))}{\hbar(z_i)^2} - (p_k-p_k^2)\right)\right. \\
&\left.-\left(\frac{m_l\varrho_k}{n}p_k - p_k\sum_{\upsilon=1}^{K}\frac{m_l\varrho_\upsilon}{n}p_\upsilon\right)\right], \qquad (44)
\end{aligned}$$

Therefore we have

$$\begin{aligned}
&\sum_{k=1}^{K-1}\frac{\partial^2 g_l(\boldsymbol{x})/\partial u_k^2}{g_l(\boldsymbol{x})} \\
= \;& \sum_{k=1}^{K-1}\left[\sum_{i=n_{l-1}+1}^{n_l}\left(\frac{p_k\hbar_k(z_i)}{\hbar(z_i)} - p_k\right) + \left(\frac{m_l\varrho_k}{n} - \sum_{\upsilon=1}^{K}\frac{m_l\varrho_\upsilon}{n}p_\upsilon\right)\right]^2 \qquad (45)\\
&+\sum_{k=1}^{K-1}\left[\sum_{i=n_{l-1}+1}^{n_l}\left(\frac{p_k\hbar_k(z_i)\hbar(z_i) - (p_k\hbar_k(z_i))^2}{\hbar(z_i)^2} - (p_k-p_k^2)\right) - \left(\frac{m_l\varrho_k}{n}p_k - p_k\sum_{\upsilon=1}^{K}\frac{m_l\varrho_\upsilon}{n}p_\upsilon\right)\right]
\end{aligned}$$

which is surely bounded below by some constant. The constant can be evaluated explicitly. For example, if we choose a uniform prior for $\boldsymbol{p}$ ($\varrho_\upsilon = 1, \upsilon = 1, \cdots, K$) the above formula can be further simplified as

$$\begin{aligned}
&\sum_{k=1}^{K-1}\frac{\partial^2 g_l(\boldsymbol{x})/\partial u_k^2}{g_l(\boldsymbol{x})} \\
= \;& \sum_{k=1}^{K-1}\left\{\left[\sum_{i=n_{l-1}+1}^{n_l}\left(\frac{p_k\hbar_k(z_i)}{\hbar(z_i)} - p_k\right)\right]^2 + \sum_{i=n_{l-1}+1}^{n_l}\left(\frac{p_k\hbar_k(z_i)\hbar(z_i) - (p_k\hbar_k(z_i))^2}{\hbar(z_i)^2} - (p_k-p_k^2)\right)\right\} \\
\geq \;& -\frac{m_j}{4} \qquad (46)
\end{aligned}$$

B.2. Lower bound for $\sum_{k=1}^{K}\dfrac{\partial^2 g_l(\boldsymbol{x})/\partial\delta_k^2}{g_l(\boldsymbol{x})}$

From the formula (36), we have

$$\partial g_l(\boldsymbol{x})/\partial\delta_k \;=\; g_l(\boldsymbol{x})\left[\sum_{i=n_{l-1}+1}^{n_l}\hbar(z_i;\boldsymbol{x})^{-1}\frac{\partial\hbar(z_i;\boldsymbol{x})}{\partial\delta_k} - \frac{\sigma_k m_l}{n}(\delta_k - \eta\mu_k)\right]$$

and

$$\begin{aligned}\partial^2 g_l(\boldsymbol{x})/\partial\delta_k^2 &= g_l(\boldsymbol{x})\left[\sum_{i=n_{l-1}+1}^{n_l}\frac{p_k\hbar_k(z_i)[z_i\eta-\delta_k]}{\hbar(z_i)}-\frac{\sigma_k m_l}{n}(\delta_k-\eta\mu_k)\right]^2\\ &\quad+g_l(\boldsymbol{x})\left[\sum_{i=n_{l-1}+1}^{n_l}\frac{p_k\hbar_k(z_i)\{[z_i\eta-\delta_k]^2-1\}\hbar(z_i)-p_k^2\hbar_k(z_i)^2[z_i\eta-\delta_k]^2}{\hbar(z_i)^2}-\frac{\sigma_k m_l}{n}\mu_k\right]\end{aligned}$$

Therefore

$$\begin{aligned}&\sum_{\upsilon=1}^{K}\frac{\partial^2 g_l(\boldsymbol{x})/\partial\delta_\upsilon^2}{g_l(\boldsymbol{x})}\\ \geq\ &\sum_{\upsilon=1}^{K}\sum_{i=n_{l-1}+1}^{n_l}\frac{p_\upsilon\hbar_\upsilon(z_i)\{[z_i\eta-\delta_\upsilon]^2-1\}\hbar(z_i)-p_\upsilon^2\hbar_\upsilon(z_i)^2[z_i\eta-\delta_\upsilon]^2}{\hbar(z_i)^2}-\sum_{\upsilon=1}^{K}\frac{\sigma_\upsilon m_l}{n}\mu_k\\ \geq\ &-m_l-\frac{\sigma_k m_l}{n}\sum_k\mu_k\end{aligned}$$

B.3. Lower bound for $\dfrac{\partial^2 g_l(\boldsymbol{x})/\partial\eta}{g_l(\boldsymbol{x})}$

From the formula (36), we have

$$\begin{aligned}\partial g_l(\boldsymbol{x})/\partial\eta &= g_l(\boldsymbol{x})\left[\sum_{i=n_{l-1}+1}^{n_l}\hbar(z_i;\boldsymbol{x})^{-1}\frac{\partial\hbar(z_i;\boldsymbol{x})}{\partial\eta}-2\frac{m_l b}{n}\eta+\sum_{k=1}^{K}\frac{\sigma_k m_l}{n}(\delta_k-\eta\mu_k)\mu_k\right.\\ &\quad\left.+\left(\frac{m_l}{2}+\frac{m_l}{2n}(2a-1)\right)\frac{2\eta}{\eta^2+c}\right]\end{aligned}$$

and

$$\begin{aligned}&\partial^2 g_l(\boldsymbol{x})/\partial\delta_k^2\\ =\ &g_l(\boldsymbol{x})\left[\sum_{i=n_{l-1}+1}^{n_l}\frac{\sum_{k=1}^{k}p_k\hbar_k(z_i)(\delta_k-z_i\eta)z_i}{\hbar(z_i)}-2\frac{m_l b}{n}\eta+\sum_{k=1}^{K}\frac{\sigma_k m_l}{n}(\delta_k-\eta\mu_k)\mu_k\right.\\ &\left.+\left(\frac{m_l}{2}+\frac{m_l}{2n}(2a-1)\right)\frac{2\eta}{\eta^2+c}\right]^2\\ &+g_l(\boldsymbol{x})\left[\sum_{i=n_{l-1}+1}^{n_l}\frac{\sum_{k=1}^{K}p_k\hbar_k(z_i)\{[\delta_k-z_i\eta]^2z_i^2-z_i^2\}\hbar(z_i)-\left(\sum_k p_k\hbar_k(z_i)[\delta_k-z_i\eta]z_i\right)^2}{\hbar(z_i)^2}-2\frac{m_l b}{n}\right.\\ &\left.-\sum_{k=1}^{K}\frac{\sigma_k m_l}{n}\mu_k^2-\left(\frac{m_l}{2}+\frac{m_l}{2n}(2a-1)\right)\frac{2(c-\eta^2)}{(\eta^2+c)^2}\right].\end{aligned}$$

Therefore

$$\begin{aligned}
&\sum_{v=1}^{K} \frac{\partial^2 g_l(\boldsymbol{x})/\partial\eta^2}{g_l(\boldsymbol{x})} \\
\geq\ & \sum_{i=n_{l-1}+1}^{n_l} z_i^2 - 2\frac{m_l b}{n} - \sum_{v=1}^{K} \frac{\sigma_v m_l}{n}\mu_v^2 - \left(\frac{m_l}{2} + \frac{m_l}{2n}(2a-1)\right)\frac{1}{c}. \qquad (47)
\end{aligned}$$

References

Beskos A., Papaspiliopoulos O., Roberts G. O. and Fearnhead P. (2006). Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processess. *Journal of Royal Statistical Society*, B, **68**: 333–382.

Beskos A. and Roberts G. O. (2005). Exact simulation on diffusions. *The Annals of Applied Probability*, **15**: 2422–2444.

Beskos A., Papaspiliopoulos O. and Roberts G. O. (2008). A factorisation of diffusion measure and finite sample path constructions. *Methodol Comput Appl Probab*, **10**: 85–104.

Breyer L. A. and Roberts G. O. (2001). Catalytic perfect simulation. *Methodology and Computing in Applied Probability*, **3**: 161–177.

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

Celeux G. and Hurn M. and Robert C. P. (2000). Computational and inferential difficulties with mixtures posterior distributions. *Journal of the American Statistical Association*, **95**: 957–970.

Diebolt J. and Roberts C. P. (1994). Estimation of finite mixture distributions through Bayesian sampling. *Journal of the Royal Statistical Society, B*, **56**: 363–375.

Dai H. (2007). *Perfect Simulation Methods for Bayesian Applications*, D.Phil Thesis, University of Oxford, supervisor: Peter Clifford.

Dai H. (2008). Perfect sampling methods for random forests, *Advances in Applied Probability*, **40**:897-917, 2008.

Dai H. (2011). Exact Monte Carlo simulation for fork-join networks, *Advances in Applied Probability*, **43**:483-503, 2011.

Dai H. (2013). *Exact simulation for diffusion bridges – an adaptive approach*, in revision.

Fearnhead P. (2005). Direct simulation for discrete mixture distributions. *Statistics and Computing*, **15**(2):125-133.

Fearnhead P. and Meligkotsidou L. (2007). Filtering methods mixture distributions. *Journal of Computational and Graphical Statistics*, **16**(3):586-607.

Gilks W. R. and Wild P. (1992). Adaptive rejection sampling for Gibbs Sampling. *Applied Statistics*, **41**(4):337-348.

Hansen. N. R. (2003). Geometric ergodicity of discrete-time approximations to multivariate diffusions. *Bernoulli*, **9**(4):725-743.

Hobert J. and Robert C. and Titterington D. (1999). On perfect simulation for some mixture of distributions. *Statistics and Computing*, **9**:287-298.

Huber M. (2004). Perfect sampling using bounding chains. *The Annals of Applied probability*, **14**:734-753.

Leydold J. (1998). A rejection technique for sampling from log-concave multivariate distributions. *Modeling and Computer Simulation*, **8**(3):254-280.

Mengersen K. L. and Robert C. P. (1996). Test for mixtures: A Bayesian entropic approach. *Bayesian Statistics 5, J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith eds.*, 255-276.

Mira A. and Moller J. and Roberts G. O. (2001). Perfect slice samplers. *Journal of the Royal Statistical Society*, B, **63**:593-606.

Potzelberger K. and Wang L. (2001). Boundary crossing probability for Brownian motion. *Journal of applied probability*, **38**:152-164.

Propp J. G. and Wilson D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structure and Algorithms*, **9**:223-252.

Richardson S. and Green P. J. (1997). On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society*, B, **59**:731-792.

Roberts G. O. and Stramer O. (2001). On inference for partially observed nonlinear diffusion models using the Metropolis-Hastings algorithm. *Biometrika*, **88**:603-621.

Stephens M. (1997). Bayesian Methods for Mixtures of Normal Distributions, D.Phil. thesis. Department of Statistics, University of Oxford.

Stephens M. (2000a). Dealing with label switching in mixture models. *Journal of the Royal Statistical Society*, B: **62**:795-809.

Stephens M. (2000b). Bayesian analysis of mixture models with an unknown number of components - an alternative to reversible jump methods. *The Annals of Statistics*, B: **28**:40-74.

Wilson D. B. (2000). How to couple from the past using a Read-Once source of randomness. *Random Structures and Algorithms*, **16**:85-113.