Bernoulli **23**(4A), 2017, 2434–2465
DOI: 10.3150/16-BEJ814

A new rejection sampling method without using hat function

HONGSHENG DAI

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

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 *coupling from the past* method, the new method does not require constructing fast coalescence Markov chains. 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 certain problems. An application on exact simulation of the posterior of finite mixture models is presented.

Keywords: conditioned Brownian motion; coupling from the past; diffusion bridges; exact Monte Carlo simulation; Langevin diffusion; mixture models; rejection sampling

1. Introduction

1.1. Background of exact Monte Carlo simulation

Monte Carlo methods are widely used in statistics, engineering and physics for generation of realisations from a probability distribution, optimization, numerical integration and so on. They are particularly important for many complex models where analytical analyses are infeasible. Markov chain Monte Carlo (MCMC) methods have been the most popular methods in more than 20 years for analysis of complex probabilistic models. 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 exact Monte Carlo methods, that is, methods that can be guaranteed to provide independent samples from the target distribution.

A breakthrough in the search for exact Monte Carlo simulation methods was made by [26]. 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 with a partial order preserved by an appropriate Markov chain construction. Although in recent decades, there have been many theoretical developments and applications in this area

such as [4,8,15,16,20,24,28] and [9], the CFTP algorithm is still not practical for complex statistical models.

Exact simulation 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 [17].

For certain complicated problems, existing methods do not work. An example in Bayesian statistics, for which all existing exact Monte Carlo simulation fails, is the Monte Carlo simulation problem for the posterior of finite mixture models (see Section 5 for details). This motivates us to search for new exact Monte Carlo simulation methods.

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 the target density f, as $f(\cdot) = g_1(\cdot)g_2(\cdot)$, where g_1 and g_2 are also (proportional to) 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 multiplicative constant. 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 $\mathbf{x}_1$ and $\mathbf{x}_2$ from g_1 and g_2, respectively. If the two independent samples $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{y}$, then it is easy to show (at least for discrete variables and heuristically for continuous variables) that the value $\mathbf{y}$ must be from $f(\cdot) \propto g_1(\cdot)g_2(\cdot)$. Note that for discrete random variables, f, g_1 and g_2 correspond to probability mass functions and it is possible to simulate $\mathbf{y}$ from f using the above idea since $\mathrm{P}(\mathbf{x}_1 = \mathbf{x}_2) > 0$. For continuous random variables, however, this is impossible since $\mathrm{P}(\mathbf{x}_1 = \mathbf{x}_2) = 0$.

Although it is impossible to achieve $\mathbf{x}_1$ and $\mathbf{x}_2$ with distance 0 for continuous case, the simulated $\mathbf{x}_1$ and $\mathbf{x}_2$, if they are very *close* (defined in later sections), can be viewed as approximately from the target f. Our idea is to use $\mathbf{x}_1$ (or $\mathbf{x}_2$) as a proposal and then accept $\mathbf{x}_1$ (or $\mathbf{x}_2$) as a perfect sample from f based on exact Monte Carlo simulation of diffusion bridges [1,2]. 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 the CFTP algorithm. In Section 3, we present the detailed exact simulation algorithm. In Section 4, we provide a generalised version of the new method. Then we apply 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 $f(\mathbf{x})$ with support $\mathbf{R}^q$ for a q-dimensional random variable $\mathbf{X}$. Suppose that it is non-trivial to simulate from f and that f can be decomposed as a product of two proper density functions, $f(\mathbf{x}) \propto g_1(\mathbf{x})g_2(\mathbf{x})$. Note that here f, g_1 and g_2 are densities up to a multiplicative constant. Assume that g_1^2 is also a proper density up to a multiplicative constant and that we can easily simulate from g_1 and g_2. We can further write $f(\mathbf{x}) \propto f_1(\mathbf{x})f_2(\mathbf{x})$ with $f_1 = g_1^2$ and $f_2 = g_2/g_1$.

Let

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

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

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

$$d\mathbf{X}_t = \boldsymbol{\alpha}(\mathbf{X}_t)\, dt + d\mathbf{B}_t, \tag{2}$$

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

From the equations in (1), we know that the above $\mathbf{X}_t$ is a Langevin diffusion [18] with the invariant distribution $f_1(\mathbf{x})$, which means $\mathbf{X}_t \sim f_1(\mathbf{x})$ for any $t \in [0, T]$ if $\mathbf{X}_0 \sim f_1(\mathbf{x})$. Let $\mathbb{Q}_{\boldsymbol{\omega}_0}$ be the probability law induced by $\mathbf{X}_t, t \in [0, T]$, given $\mathbf{X}_0 = \boldsymbol{\omega}_0$. Let $\mathbb{Q}$ be the probability law induced by $\mathbf{X}_t, t \in [0, T]$, with $\mathbf{X}_0 = \boldsymbol{\omega}_0 \sim f_1(\cdot)$, that is, under $\mathbb{Q}$ we have $\mathbf{X}_t \sim f_1(\mathbf{x})$ 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*

$$\phi(\mathbf{x}) = \frac{1}{2}\left(\|\boldsymbol{\alpha}\|^2 + \mathbf{div}\,\boldsymbol{\alpha}\right)(\mathbf{x}) - l \geq 0, \tag{3}$$

where **div** *is the divergence of $\boldsymbol{\alpha}$, defined in (6).*

Condition 2.3. *The following*

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

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

Consider a *biased* diffusion process $\bar{\mathbf{X}} = \{\bar{\mathbf{X}}_t; 0 \le t \le T\}$ defined as follows. First, the joint density for the pair $(\bar{\mathbf{X}}_0, \bar{\mathbf{X}}_T)$ (the starting and ending points of the biased diffusion process), evaluated at point $(\mathbf{x}, \mathbf{y})$, is $f_1(\mathbf{x})\mathbf{t}^*(\mathbf{y}|\mathbf{x})f_2(\mathbf{y})$, where $\mathbf{t}^*(\mathbf{y}|\mathbf{x})$ is the transition density for the diffusion process $\mathbf{X}$ defined in (2) from $\mathbf{X}_0 = \mathbf{x}$ to $\mathbf{X}_T = \mathbf{y}$. Note that we must have $\int f_1(\mathbf{x})\mathbf{t}^*(\mathbf{y}|\mathbf{x})\,d\mathbf{x} = f_1(\mathbf{y})$, since $\mathbf{X}$ follows the stationary distribution f_1 at all time points, if $\mathbf{X}_0 \sim f_1(\mathbf{x})$.

Second, given $(\bar{\mathbf{X}}_0, \bar{\mathbf{X}}_T)$ the process $\{\bar{\mathbf{X}}_t, 0 < t < T\}$ is given by the diffusion bridge driven by (2).

Note that $\bar{\mathbf{X}}$ is actually a biased version of $\mathbf{X}$. Conditional on the ending points, the two processes $\bar{\mathbf{X}}$ and $\mathbf{X}$ have the same distribution. Clearly the marginal distribution for $\bar{\mathbf{X}}_T$ is $f(\mathbf{y})$. This is because $\int f_1(\mathbf{x})\mathbf{t}^*(\mathbf{y}|\mathbf{x})f_2(\mathbf{y})\,d\mathbf{x} = f_2(\mathbf{y})f_1(\mathbf{y}) \propto f(\mathbf{y})$.

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

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

$$\frac{d\bar{\mathbb{Q}}}{d\mathbb{Q}}(\vec{\boldsymbol{\omega}}) \propto f_2(\boldsymbol{\omega}_T). \tag{4}$$

Proof. The proof of the lemma follows easily from the proof of Proposition 3 in [3]. □

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

We here use similar rejection sampling ideas as that in [2] and [1]. Under Conditions 2.1 to 2.3 and following [2], we have

$$\frac{d\mathbb{Q}}{d\mathbb{W}}(\vec{\boldsymbol{\omega}}) = \exp\left[A(\boldsymbol{\omega}_T) - A(\boldsymbol{\omega}_0) - \frac{1}{2}\int_0^T \left(\|\boldsymbol{\alpha}\|^2 + \mathbf{div}\,\boldsymbol{\alpha}\right)(\boldsymbol{\omega}_t)\,dt\right] \tag{5}$$

where

$$\mathbf{div}\,\boldsymbol{\alpha}(\mathbf{x}) = \sum_{j=1}^{q} \frac{\partial \boldsymbol{\alpha}^{(j)}(\mathbf{x})}{\partial x^{(j)}} \tag{6}$$

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

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

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

$$\frac{d\mathbb{Z}}{d\mathbb{W}}(\vec{\boldsymbol{\omega}}) = \frac{h(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)}{f_1(\boldsymbol{\omega}_0)(1/\sqrt{2\pi T})e^{-\|\boldsymbol{\omega}_T - \boldsymbol{\omega}_0\|^2/(2T)}}. \tag{7}$$

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

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

Therefore,

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

which implies (7). □

By letting

$$h(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T) = f_2(\boldsymbol{\omega}_T)\exp\big[A(\boldsymbol{\omega}_T) - A(\boldsymbol{\omega}_0)\big] f_1(\boldsymbol{\omega}_0) \frac{1}{\sqrt{2\pi T}} e^{-\|\boldsymbol{\omega}_T - \boldsymbol{\omega}_0\|^2/(2T)} \tag{8}$$

and using (4), (5) and (7), we have

$$\begin{aligned} \frac{d\bar{\mathbb{Q}}}{d\mathbb{Z}}(\vec{\boldsymbol{\omega}}) &\propto \frac{d\bar{\mathbb{Q}}}{d\mathbb{Q}}(\vec{\boldsymbol{\omega}}) \frac{d\mathbb{Q}}{d\mathbb{W}}(\vec{\boldsymbol{\omega}}) \frac{d\mathbb{W}}{d\mathbb{Z}}(\vec{\boldsymbol{\omega}}) \\ &= f_2(\boldsymbol{\omega}_T) \cdot \exp\left[A(\boldsymbol{\omega}_T) - A(\boldsymbol{\omega}_0) - \frac{1}{2}\int_0^T \big(\|\boldsymbol{\alpha}\|^2 + \operatorname{div}\boldsymbol{\alpha}\big)(\boldsymbol{\omega}_t)\, dt\right] \\ &\quad \cdot \frac{f_1(\boldsymbol{\omega}_0)(1/\sqrt{2\pi T})e^{-\|\boldsymbol{\omega}_T - \boldsymbol{\omega}_0\|^2/(2T)}}{h(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)} \\ &= \exp\left[-\frac{1}{2}\int_0^T \big(\|\boldsymbol{\alpha}\|^2 + \operatorname{div}\boldsymbol{\alpha}\big)(\boldsymbol{\omega}_t)\, dt\right]. \end{aligned} \tag{9}$$

Then under Condition 2.2, we can rewrite (9) as

$$\frac{d\bar{\mathbb{Q}}}{d\mathbb{Z}}(\vec{\boldsymbol{\omega}}) \propto \exp\Big[-\int_0^T \phi(\boldsymbol{\omega}_t)\,dt\Big], \tag{10}$$

which has a value no more than 1. Now it is ready to use rejection sampling to simulate $\bar{\mathbf{X}}_t$ from $\bar{\mathbb{Q}}$. First, we simulate a proposal $\bar{\mathbf{B}}_t$ from $\mathbb{Z}$ and then we accept the proposal as $\bar{\mathbf{X}}_t$ according to the probability in (10). Note that this rejection sampling can be done using similar methods as that in [2] and [1].

Note that to simulate a proposal $\bar{\mathbf{B}}_t$ from $\mathbb{Z}$, it is necessary to simulate $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$ from h given in (8). This is not difficult, because according to $f_1 = g_1^2$ and $f_2 = g_2/g_1$ we have

$$\begin{aligned} h(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T) &= f_2(\boldsymbol{\omega}_T)\exp\big[A(\boldsymbol{\omega}_T) - A(\boldsymbol{\omega}_0)\big] f_1(\boldsymbol{\omega}_0)\frac{1}{\sqrt{2\pi T}} e^{-\|\boldsymbol{\omega}_T - \boldsymbol{\omega}_0\|^2/(2T)} \\ &= g_2(\boldsymbol{\omega}_T) g_1(\boldsymbol{\omega}_0)\frac{1}{\sqrt{2\pi T}} e^{-\|\boldsymbol{\omega}_T - \boldsymbol{\omega}_0\|^2/(2T)}. \end{aligned} \tag{11}$$

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 the probability $\exp[-\frac{\|\boldsymbol{\omega}_T - \boldsymbol{\omega}_0\|^2}{2T}]$.

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

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

Remark 1. Step 9 of Algorithm 1 can be done using the method in [2] and [1].

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 $\boldsymbol{\omega}_T$ 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

$$\begin{aligned} AP_1 &= \mathrm{P}\big\{U \le \exp\big[-\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T\|^2/(2T)\big]\big\}, \\ AP_2 &= \mathrm{P}\big(\mathcal{I} = 1 | (\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)\big). \end{aligned} \tag{12}$$

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

1 Simulate $\boldsymbol{\omega}_0$ from g_1 and $\boldsymbol{\omega}_T$ from g_2;
2 Simulate a standard uniform variable U;
3 **if** $U \leq \exp[-\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T\|^2/(2T)]$ **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 $\bar{\mathbf{B}} = \{\boldsymbol{\omega}_t, t \in (0, T)\}$ conditional on $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$;
9 Simulate $\mathcal{I} = 1$ with probability given by (10);
10 **if** $\mathcal{I} = 1$ **then**
11 | Output $\boldsymbol{\omega}_T$;
12 **else**
13 | return to Step 1;
14 **end**

If T is large, the probability AP_1 will be relatively large, but the probability AP_2 which can also be written as

$$\mathrm{P}\big(\mathcal{I} = 1 | (\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)\big) = \exp\bigg(-\int_0^T \phi(\boldsymbol{\omega}_t)\, dt\bigg), \tag{13}$$

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 probabilities AP_1 and AP_2 to be as large as possible. In practice, it may be more desirable to have a larger value of AP_2, the acceptance probability for the diffusion bridge, since Steps 8 and 9 (simulation of the diffusion bridge) in Algorithm 1 are not easy to implement [1,2]. 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 (12) depends on the lower bound l for $(\|\boldsymbol{\alpha}\|^2 + \operatorname{div}\boldsymbol{\alpha})/2$. Therefore, this algorithm will be attractive when it is possible to find good lower bounds for $(\|\boldsymbol{\alpha}\|^2 + \operatorname{div}\boldsymbol{\alpha})/2$, 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 $(\|\boldsymbol{\alpha}\|^2 + \operatorname{div}\boldsymbol{\alpha})/2$. 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 accepts $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$ 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.

From Lemma 2.1, we know that $d\bar{\mathbb{Q}}(\vec{\boldsymbol{\omega}}) \propto f_2(\boldsymbol{\omega}_T) d\mathbb{Q}(\vec{\boldsymbol{\omega}})$. Therefore, the biased diffusion $\bar{\mathbf{X}}$ can be generated from $\bar{\mathbb{Q}}$ via the following heuristic steps. Step 1: we generate $\boldsymbol{\omega}_0 \sim f_1$ and then the diffusion process $\boldsymbol{\omega}_t, 0 \le t \le T$, which is governed by (2). This means that $\{\boldsymbol{\omega}_t, 0 \le t \le T\}$ is generated from $\mathbb{Q}$. Step 2: we generate $\boldsymbol{\omega}'_T \sim f_2$, independent of $\{\boldsymbol{\omega}_t, 0 \le t \le T\}$. Then if $\boldsymbol{\omega}_T = \boldsymbol{\omega}'_T$, we can accept that $\{\boldsymbol{\omega}_t, 0 \le t \le T\}$ as a realisation of the biased diffusion $\bar{\mathbf{X}}$.

We can also imagine that the above Step 2 simulates another diffusion process $\boldsymbol{\omega}'_t, 0 \le t \le T$ with invariant distribution f_2, but only output the process at time T. The two processes are simulated independently and coalesce at time T, a pre-determined value. This means that two random variables (but having the same value) $\boldsymbol{\omega}_T$ 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 the 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 predetermined 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 predetermined time point. This challenge is solved via rejection sampling for diffusions, that is, 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 independently 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 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 $f_p(\mathbf{p}) = p_1^4 p_2^4 (1 - p_1 - p_2)^4$, $0 \le p_1, p_2 \le 1$. Since Algorithm 1 requires that the target $f(\cdot)$ should have support in $\mathbf{R}^2$, we first consider the transformed variable $\mathbf{x} = (x_1, x_2)$ with

$$\begin{aligned} p_1 &= p_1(\mathbf{x}) := \exp(x_1) \big/ \big[1 + \exp(x_1) + \exp(x_2)\big], \\ p_2 &= p_2(\mathbf{x}) := \exp(x_2) \big/ \big[1 + \exp(x_1) + \exp(x_2)\big]. \end{aligned} \tag{14}$$

We have

$$\begin{vmatrix} \frac{\partial p_1}{\partial x_1} & \frac{\partial p_1}{\partial x_2} \\ \frac{\partial p_2}{\partial x_1} & \frac{\partial p_2}{\partial x_2} \end{vmatrix} = \begin{vmatrix} p_1(\mathbf{x}) - p_1(\mathbf{x})^2 & -p_1(\mathbf{x}) p_2(\mathbf{x}) \\ -p_1(\mathbf{x}) p_2(\mathbf{x}) & p_2(\mathbf{x}) - p_2(\mathbf{x})^2 \end{vmatrix} = p_1(\mathbf{x}) p_2(\mathbf{x}) \big(1 - p_1(\mathbf{x}) - p_2(\mathbf{x})\big)$$

since according to (14) we have

$$\begin{aligned} &\partial p_k(\mathbf{x}) / \partial x_k = p_k(\mathbf{x}) - p_k(\mathbf{x})^2, \\ &\partial p_j(\mathbf{x}) / \partial x_k = -p_j(\mathbf{x}) p_k(\mathbf{x}), \qquad j \ne k. \end{aligned} \tag{15}$$

Therefore, the Jacobin associated with the transformation is

$$J(\mathbf{x}) = p_1(\mathbf{x}) p_2(\mathbf{x}) \big(1 - p_1(\mathbf{x}) - p_2(\mathbf{x})\big).$$

Then the density function for $\mathbf{x}$ can be written as $f_{\mathbf{x}}(\mathbf{x}) = f_p(p_1(\mathbf{x}), p_2(\mathbf{x})) J(\mathbf{x})$. Therefore

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

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

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

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

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

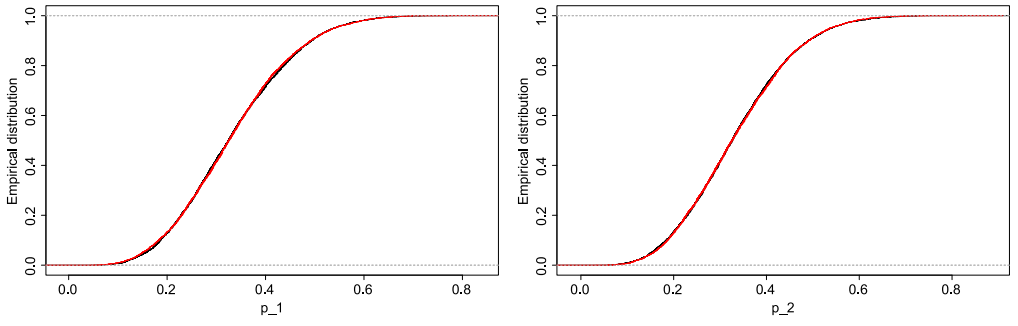

Figure 1. Left figure: marginal empirical distribution for p_1; right figure: marginal empirical distribution for p_2. Light curve: empirical distribution based on 'rdirichlet'; black curve: empirical distribution based on the new method. For both p_1 and p_2, the light and black curves overlap.

We further have

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

and $\|\boldsymbol{\alpha}(\mathbf{x})\|^2 + \operatorname{div}\boldsymbol{\alpha}(\mathbf{x}) \geq -3$. Therefore, the proposed algorithm can be applied.

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, that is, 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. The new rejection sampling algorithm with layered Brownian motion

We can see that in Algorithm 1, there are two rejection steps: line 3 and line 10. The acceptance probabilities are given in (12). These two probabilities can be very small, especially when the dimension, q, of $\boldsymbol{\omega}_t$ is large. In this section, we provide details on how to improve Algorithm 1.

3.1. Improvement on the diffusion bridge simulation – The one-dimensional case

The method proposed in this subsection relies on the concept of *layered Brownian motion* in [1]. So we first briefly introduce in Section 3.1.1 the *layers* defined in [1] and the methods developed therein.

3.1.1. *The layered Brownian motion in* [1]

Define the probability measure $\mathbb{W}_{0,T}^{\omega_0,\omega_T}$, under which ω_t is a Brownian bridge with (ω_0, ω_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 $\bar{x} = \omega_0 \wedge \omega_T$, $\bar{y} = \omega_0 \vee \omega_T$. Define the ith layer, $\mathcal{D}_i(\bar{x}, \bar{y}; 0, T) = \mathcal{U}_i(\bar{x}, \bar{y}; 0, T) \cup \mathcal{L}_i(\bar{x}, \bar{y}; 0, T)$, where

$$\begin{aligned}
&\mathcal{U}_i(\bar{x}, \bar{y}; 0, T) \\
&\quad = \Big\{\vec{\boldsymbol{\omega}} : \sup_{0\leq s\leq T} \omega_s \in [\bar{y} + a_{i-1}, \bar{y} + a_i]\Big\} \cap \Big\{\vec{\boldsymbol{\omega}} : \inf_{0\leq s\leq T} \omega_s > \bar{x} - a_i\Big\}, \\
&\mathcal{L}_i(x, y; 0, T) \\
&\quad = \Big\{\vec{\boldsymbol{\omega}} : \inf_{0\leq s\leq T} \omega_s \in [\bar{x} - a_i, \bar{x} - a_{i-1}]\Big\} \cap \Big\{\vec{\boldsymbol{\omega}} : \sup_{0\leq s\leq T} \omega_s < \bar{y} + a_i\Big\}.
\end{aligned} \tag{16}$$

For simplicity of notations, we use $\mathcal{D}_i$ to represent $\mathcal{D}_i(\bar{x}, \bar{y}; 0, T)$.

In fact, it will be easier for us to understand the event $\mathcal{O}_i$, defined as

$$\mathcal{O}_i = \bigcup_{k=1}^{i} \mathcal{D}_k. \tag{17}$$

If $\vec{\boldsymbol{\omega}} \in \mathcal{O}_i$, then the Brownian bridge is bounded by $[\bar{x} - a_i, \bar{y} + a_i]$. Because $\{\mathcal{D}_i, i = 1, \ldots\}$ forms a partition for the space of $\vec{\boldsymbol{\omega}}$, we can also write $\mathcal{D}_i = \mathcal{O}_i - \mathcal{O}_{i-1}$. Therefore, the Brownian bridge in layer i, $\{\omega_t, 0 < t < T\} \in \mathcal{D}_i$, means that $\omega_t \in [\bar{x} - a_i, \bar{y} + a_i]$, $0 < t < T$, but ω_t is not bounded by $[\bar{x} - a_{i-1}, \bar{y} + a_{i-1}]$.

In Algorithm 1, the acceptance indicator $\mathcal{I}$ can be simulated via the following subroutine (Algorithm 2) [1]. Algorithm 2 simulates a layer I according to the probability $\mathrm{P}(I = i|\omega_0, \omega_T) = W_{0,T}^{\omega_0,\omega_T}\{\vec{\boldsymbol{\omega}} \in \mathcal{D}_i\}$. Then conditional on the layer I, the Brownian bridge is simulated. Thus, the simulated Brownian bridge will be bounded by the boundaries of the Ith layer, that is, bounded by $[\bar{x} - a_I, \bar{y} + a_I]$. Based on the boundaries of the Brownian bridge, an upper bound r_I for $\sup_{t\in[0,T],\vec{\boldsymbol{\omega}}\in\mathcal{D}_I}\{[\alpha^2(\omega_t) + \alpha'(\omega_t)]/2 - l\}$ can then be found and a Poisson thinning algorithm can be used to simulate the event $\mathcal{I}$.

3.1.2. *Improve the diffusion bridge simulation by re-weighting the layer probabilities*

The acceptance probability for the diffusion bridge can be very small, if the lower bound l is very small. Therefore, Algorithm 2 may be very inefficient due to the low acceptance probability and the complexity of diffusion bridge simulation. To improve the efficiency, we should increase the lower bound l. For one-dimensional case, [10] 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 [10] to multi-dimensional processes in Section 3.2. Note that [25] proposed a method using more flexible *layers* to simulate one-dimensional jump diffusions.

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 $l_i \leq \inf_{s\in[0,T],\vec{\boldsymbol{\omega}}\in\mathcal{D}_i}\{(\alpha^2 + \alpha')(\omega_s)/2\}$ and $l_i \to l$. Obviously, for all i, such l_i satisfies $l_i \geq l$.

Algorithm 2: Simulation for $\mathcal{I}$, one-dimensional case

1 Given ω_0 and ω_T, simulate a Brownian bridge $\omega_t, t \in [0, T]$ via the following Steps 1a and 1b:;
   // 1a: Simulate layer I with probability $\mathrm{P}(I) = \mathbb{W}_{0,T}^{\omega_0,\omega_T}(\vec{\boldsymbol{\omega}} \in \mathcal{D}_I)$
   // 1b: Given ω_0 and ω_T, simulate a sample path $\vec{\boldsymbol{\omega}}$, from $\mathbb{W}_{0,T}^{\omega_0,\omega_T}$ conditional on $\vec{\boldsymbol{\omega}} \in D_I$, using the algorithm in [1]
2 Calculate $l = \inf[\boldsymbol{\alpha}^2(u) + \alpha'(u)]/2$, for all $u \in \mathbf{R}$;
3 Calculate r_I such that $r_I \geq \sup_{t\in[0,T],\vec{\boldsymbol{\omega}}\in\mathcal{D}_I}\{[\alpha^2(\omega_t) + \alpha'(\omega_t)]/2 - l\}$;
4 Simulate $\boldsymbol{\Psi} = \{\psi_1, \ldots, \psi_\rho\}$ uniformly distributed on $\mathbf{U}[0, T]$ and marks $\Upsilon = \{\nu_1, \ldots, \nu_\rho\}$ uniformly distributed on $\mathbf{U}[0, 1]$, where ρ is from $\mathrm{Poi}(r_I T)$;
5 Compute the acceptance indicator $\mathcal{I} := \prod_{j=1}^{\rho} I[r_I^{-1}\phi(\omega_{\psi_j}) < \nu_j]$;

Define a process $\{\widetilde{B}_t, 0 \leq t \leq T\}$ which has the probability law $\widetilde{\mathbb{Z}}$ given by the Radon–Nikodym derivative, with respect to $\mathbb{W}$,

$$\frac{d\widetilde{\mathbb{Z}}}{d\mathbb{W}}(\vec{\boldsymbol{\omega}}) \propto \frac{h(\omega_0, \omega_T)}{f_1(\omega_0)\cdot(1/\sqrt{2\pi T})e^{-(\omega_0-\omega_T)^2/(2T)}} \sum_{i=1}^{\infty} \exp\{-Tl_i\} I\{\vec{\boldsymbol{\omega}} \in \mathcal{D}_i\}.$$

We then have

$$\begin{aligned}
\frac{d\bar{\mathbb{Q}}}{d\widetilde{\mathbb{Z}}}(\vec{\boldsymbol{\omega}}) &\propto \frac{d\bar{\mathbb{Q}}}{d\mathbb{Q}}(\vec{\boldsymbol{\omega}})\frac{d\mathbb{Q}}{d\mathbb{W}}(\vec{\boldsymbol{\omega}})\frac{d\mathbb{W}}{d\widetilde{\mathbb{Z}}}(\vec{\boldsymbol{\omega}}) \\
&= f_2(\omega_T)\cdot\exp\left[A(\omega_T) - A(\omega_0) - \frac{1}{2}\int_0^T \left(\alpha^2 + \alpha'\right)(\omega_t)\,dt\right] \\
&\quad\cdot \frac{f_1(\omega_0)(1/\sqrt{2\pi T})e^{-|\omega_T-\omega_0|^2/(2T)}}{h(\omega_0, \omega_T)} \cdot \frac{1}{\sum_{i=1}^{\infty}\exp\{-Tl_i\}I\{\vec{\boldsymbol{\omega}} \in \mathcal{D}_i\}} \\
&\propto \sum_{i=1}^{\infty}\exp\left\{-\int_0^T\left[\frac{1}{2}\left(\alpha^2 + \alpha'\right)(\omega_s) - l_i\right]ds\right\} I\{\vec{\boldsymbol{\omega}} \in \mathcal{D}_i\},
\end{aligned} \tag{18}$$

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 $\widetilde{\mathbb{Z}}(\vec{\boldsymbol{\omega}})$. The acceptance probability ratio in (18) will be larger than the acceptance probability ratio in (10), since $l_i \geq l$.

Note that, simulating from $\widetilde{\mathbb{Z}}(\vec{\boldsymbol{\omega}})$ can actually be done based on the method in [10] if $\vec{\boldsymbol{\omega}}$ is a one-dimensional process.

3.2. Improvement on the diffusion bridge simulation, when $\mathbf{x} \in \mathbf{R}^q$

The methodology in Section 3.1 can be extended to q-dimensional processes. Suppose that $\vec{\boldsymbol{\omega}} = (\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(q)})$ and $\boldsymbol{\omega}^{(j)} = \{\omega_s^{(j)}, s \in [0, T]\}$. Given $\boldsymbol{\omega}_0$ and $\boldsymbol{\omega}_T$, define $\bar{x}^{(j)} = \omega_0^{(j)} \wedge \omega_T^{(j)}$ and $\bar{y}^{(j)} = \omega_0^{(j)} \vee \omega_T^{(j)}$.

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 that in (16) for each component $\boldsymbol{\omega}^{(j)}$. For simplicity, the sequence $\{a_i\}$ in (16) is chosen to be same for all components $\boldsymbol{\omega}^{(j)}$, $j = 1, \ldots, q$. With these definitions, the event $\boldsymbol{\omega}^{(j)} \in \bigcup_{k=1}^{i} \mathcal{D}_k^{(j)}$ means that the jth component of the q-dimensional Brownian bridge is bounded by $[\bar{x}^{(j)} - a_i, \bar{y}^{(j)} + a_i]$.

Further, by defining

$$\mathcal{O}_i^{(j)} := \bigcup_{k=1}^{i} \mathcal{D}_k^{(j)}, \tag{19}$$

the event $\vec{\boldsymbol{\omega}} \in \bigotimes_{j=1}^{q} \mathcal{O}_i^{(j)}$ means that all components of the q-dimensional Brownian bridge are bounded by the ith layer boundaries, $[\bar{x}^{(j)} - a_i, \bar{y}^{(j)} + a_i]$, $j = 1, \ldots, q$, respectively. Here the sign $\otimes$ is the direct product.

Define $\boldsymbol{\mathcal{Q}}_i = \bigotimes_{j=1}^{q} \mathcal{O}_i^{(j)} - \bigotimes_{j=1}^{q} \mathcal{O}_{i-1}^{(j)}$. Clearly, $\{\boldsymbol{\mathcal{Q}}_i, i = 1, \ldots\}$ form a partition for the space of the q-dimensional $\vec{\boldsymbol{\omega}}$. We say that $\boldsymbol{\mathcal{Q}}_i$ is the ith layer.

We can write

$$\begin{aligned}
\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0, \boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\} &= \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0, \boldsymbol{\omega}_T}\left(\vec{\boldsymbol{\omega}} \in \bigotimes_{j=1}^{q} \mathcal{O}_i^{(j)}\right) \\
&\quad - \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0, \boldsymbol{\omega}_T}\left(\vec{\boldsymbol{\omega}} \in \bigotimes_{j=1}^{q} \mathcal{O}_{i-1}^{(j)}\right) \\
&= \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0, \boldsymbol{\omega}_T}\left(\vec{\boldsymbol{\omega}} \in \bigotimes_{j=1}^{q} \left[\bigcup_{k=1}^{i} \mathcal{D}_k^{(j)}\right]\right) \\
&\quad - \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0, \boldsymbol{\omega}_T}\left(\vec{\boldsymbol{\omega}} \in \bigotimes_{j=1}^{q} \left[\bigcup_{k=1}^{i-1} \mathcal{D}_k^{(j)}\right]\right).
\end{aligned} \tag{20}$$

On the other hand, with the definitions above, we can find l_i such that

$$l_i \leq \inf_{s \in [0,T], \vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i} \left\{\left(\|\boldsymbol{\alpha}\|^2 + \mathbf{div}\,\boldsymbol{\alpha}\right)(\boldsymbol{\omega}_s)/2\right\}$$

and $l_i \to l$. Let $\{\widetilde{\mathbf{B}}_t, 0 \le t \le T\}$ be the process having probability law $\widetilde{\mathbb{Z}}$, given by

$$\frac{d\widetilde{\mathbb{Z}}}{d\mathbb{W}}(\vec{\boldsymbol{\omega}}) \propto \frac{h(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)}{f_1(\boldsymbol{\omega}_0)e^{-\|\boldsymbol{\omega}_0-\boldsymbol{\omega}_T\|^2/(2T)}} \sum_{i=1}^{\infty} \exp\{-Tl_i\} I\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}. \tag{21}$$

We have that the Radon–Nikodym derivative of $\bar{\mathbb{Q}}$ to $\widetilde{\mathbb{Z}}$ becomes

$$\frac{d\bar{\mathbb{Q}}}{d\widetilde{\mathbb{Z}}}(\vec{\boldsymbol{\omega}}) \propto \sum_{i=1}^{\infty} \exp\left\{-\int_0^T \left[\frac{1}{2}\big(\|\boldsymbol{\alpha}\|^2 + \mathbf{div}\,\boldsymbol{\alpha}\big)(\boldsymbol{\omega}_s) - l_i\right] ds\right\} I\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}. \tag{22}$$

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

3.2.1. *Simulation from* $\widetilde{\mathbb{Z}}(\vec{\boldsymbol{\omega}})$ *given by* (21)

We can rewrite (21) as

$$\begin{aligned} d\widetilde{\mathbb{Z}}(\vec{\boldsymbol{\omega}}) &\propto \frac{h^*(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)}{f_1(\boldsymbol{\omega}_0)e^{-\|\boldsymbol{\omega}_0-\boldsymbol{\omega}_T\|^2/(2T)}} \frac{\sum_{i=1}^{\infty} \exp\{-Tl_i\} I\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}}{\sum_{i=1}^{\infty} \exp\{-Tl_i\} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}} d\mathbb{W}(\vec{\boldsymbol{\omega}}) \\ &= h^*(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T) \cdot \frac{\sum_{i=1}^{\infty} \exp\{-Tl_i\} I\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}}{\sum_{i=1}^{\infty} \exp\{-Tl_i\} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}} d\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}(\vec{\boldsymbol{\omega}}) \\ &= h^*(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T) \cdot d\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}(\vec{\boldsymbol{\omega}}), \end{aligned} \tag{23}$$

where $h^*(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T) = h(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T) \cdot \sum_{i=1}^{\infty} \exp\{-Tl_i\} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}$, $\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}$ is the Brownian bridge measure and $d\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}$ is given by

$$\frac{d\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}}{d\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}}(\vec{\boldsymbol{\omega}}) = \frac{\sum_{i=1}^{\infty} \exp\{-Tl_i\} I\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}}{\sum_{i=1}^{\infty} \exp\{-Tl_i\} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}}. \tag{24}$$

To simulate from $\widetilde{\mathbb{Z}}(\vec{\boldsymbol{\omega}})$, we can first simulate $\boldsymbol{\omega}_0, \boldsymbol{\omega}_T$ from $h^*(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$ and then conditional on $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$, we simulate $\{\boldsymbol{\omega}_t, 0 < t < T\}$ from $\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}(\vec{\boldsymbol{\omega}})$ given by (24). It is simple to simulate from h^* via rejection sampling since we can simulate from h and $h^* \cdot \exp(Tl) \le h$.

Now the key step to be solved is to simulate from $\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}(\vec{\boldsymbol{\omega}})$ given by (24). By rewriting (24) as,

$$\frac{d\widetilde{\mathbb{Z}}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}}{d\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}}(\vec{\boldsymbol{\omega}}) = \sum_{i=1}^{\infty} \left\{ \frac{\exp\{-Tl_i\} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}}{\sum_{k=1}^{\infty} \exp\{-Tl_k\} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_k\}} \frac{I\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}}{\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}} \right\} \tag{25}$$

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

$$\widetilde{\mathrm{P}}(I = i) = \frac{\exp\{-Tl_i\} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_i\}}{\sum_{k=1}^{\infty} \exp\{-Tl_k\} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \mathcal{Q}_k\}}; \tag{26}$$

this step can be done using the same method in [10].

Step 2: then conditional on the layer $I = i$ we simulate $\vec{\boldsymbol{\omega}}$ from $d\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}(\vec{\boldsymbol{\omega}})I\{\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\}$, which will be discussed later in Lemma 3.2.

Remark. Recalling $\boldsymbol{\mathcal{Q}}_i = \bigotimes_{j=1}^q \mathcal{O}_i^{(j)} - \bigotimes_{j=1}^q \mathcal{O}_{i-1}^{(j)}$, the event $\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i$ means that $\boldsymbol{\omega}^{(j)} \in \mathcal{O}_i^{(j)}$, $j = 1, \ldots, q$, but $\boldsymbol{\omega}^{(j)} \notin \mathcal{O}_{i-1}^{(j)}$ for at least one component j, say $\boldsymbol{\omega}^{(j')} \notin \mathcal{O}_{i-1}^{(j')}$. Therefore, for the above Step 2, a simple way of simulating $\vec{\boldsymbol{\omega}}$ conditional on layer $I = i$ (i.e., conditional on $\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i$) is to independently simulate each component, conditional on that $\vec{\boldsymbol{\omega}} \in \bigotimes_{j=1}^q \mathcal{O}_i^{(j)}$. Then if all simulated component are such that $\vec{\boldsymbol{\omega}} \in \bigotimes_{j=1}^q \mathcal{O}_{i-1}^{(j)}$, we have to reject the simulated $\vec{\boldsymbol{\omega}}$ and restart; otherwise we accept it. This simple approach, however, is not efficient as there is a rejection step involved. Thus, we consider a more efficient approach in Lemma 3.2.

First, we need to provide another expression for $\boldsymbol{\mathcal{Q}}_i$ in Lemma 3.1.

Lemma 3.1. *We denote* $\boldsymbol{\mathcal{S}}_i^{(j')} = [\bigotimes_{j \neq j'} \mathcal{O}_i^{(j)}] \otimes \mathcal{D}_i^{(j')}$ *for* $j' \in \{1, \ldots, q\}$. *Then we have*

$$\begin{aligned} \boldsymbol{\mathcal{Q}}_i &= \bigcup_{j'=1}^q \left(\boldsymbol{\mathcal{S}}_i^{(j')} - \bigcup_{h=j'+1}^q \boldsymbol{\mathcal{S}}_i^{(h)} \right) \\ &= \bigcup_{j'=1}^q \left\{ \left[\bigotimes_{j=1}^{j'-1} \mathcal{O}_i^{(j)} \right] \otimes \mathcal{D}_i^{(j')} \otimes \left[\bigotimes_{j=j'+1}^q \mathcal{O}_{i-1}^{(j)} \right] \right\}, \end{aligned} \tag{27}$$

where the operator $\bigotimes_{j=q+1}^q$ *gives an empty set.*

Proof. Recalling the definition of $\boldsymbol{\mathcal{Q}}_i$ and the fact that $\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i$ means $\boldsymbol{\omega}^{(j)} \in \mathcal{O}_i^{(j)}$, $j = 1, \ldots, q$, but $\boldsymbol{\omega}^{(j)} \notin \mathcal{O}_{i-1}^{(j)}$ for at least one component j, we have

$$\begin{aligned} \boldsymbol{\mathcal{Q}}_i &= \bigcup_{j'=1}^q \left\{ \left[\bigotimes_{j \neq j'} \mathcal{O}_i^{(j)} \right] \otimes \mathcal{D}_i^{(j')} \right\} \\ &= \bigcup_{j'=1}^q \boldsymbol{\mathcal{S}}_i^{(j')} = \bigcup_{j'=1}^q \left(\boldsymbol{\mathcal{S}}_i^{(j')} - \bigcup_{h=j'+1}^q \boldsymbol{\mathcal{S}}_i^{(h)} \right), \end{aligned} \tag{28}$$

where the last equality sign is just an exercise of expressing a union of sets to a union of non-intersecting sets.

Given any two sets $\mathcal{Y}_1 \supset \mathcal{Y}_2$, using the facts $\mathcal{O}_i^{(j')} = \mathcal{D}_i^{(j')} \cup \mathcal{O}_{i-1}^{(j')}$ and that $\mathcal{D}_i^{(j')} \cap \mathcal{O}_{i-1}^{(j')}$ is empty, we have $\mathcal{D}_i^{(j')} \otimes \mathcal{Y}_1 - \mathcal{O}_i^{(j')} \otimes \mathcal{Y}_2 = \mathcal{D}_i^{(j')} \otimes \mathcal{Y}_1 - \mathcal{D}_i^{(j')} \otimes \mathcal{Y}_2 = \mathcal{D}_i^{(j')} \otimes (\mathcal{Y}_1 - \mathcal{Y}_2)$.

Therefore, we have

$$
\begin{aligned}
&\boldsymbol{\mathcal{S}}_i^{(j')} - \bigcup_{h=j'+1}^{q} \boldsymbol{\mathcal{S}}_i^{(h)} \\
&= \left[\bigotimes_{j=1}^{j'-1} \mathcal{O}_i^{(j)}\right] \\
&\quad \otimes \left\{ \mathcal{D}_i^{(j')} \otimes \underbrace{\left(\bigotimes_{j=j'+1}^{q} \mathcal{O}_i^{(j)}\right)}_{\mathcal{Y}_1} - \mathcal{O}_i^{(j')} \otimes \underbrace{\bigcup_{h=j'+1}^{q} \left(\left(\bigotimes_{j=j'+1;\, j\neq h}^{q} \mathcal{O}_i^{(j)}\right) \otimes \mathcal{D}_i^{(j)}\right)}_{\mathcal{Y}_2} \right\} \qquad (29) \\
&= \left[\bigotimes_{j=1}^{j'-1} \mathcal{O}_i^{(j)}\right] \otimes \mathcal{D}_i^{(j')} \otimes \left\{ \underbrace{\bigotimes_{j=j'+1}^{q} \mathcal{O}_i^{(j)}}_{\mathcal{Y}_1} - \underbrace{\bigcup_{h=j'+1}^{q} \left(\left(\bigotimes_{j=j'+1;\, j\neq h}^{q} \mathcal{O}_i^{(j)}\right) \otimes \mathcal{D}_i^{(j)}\right)}_{\mathcal{Y}_2} \right\} \\
&= \left[\bigotimes_{j=1}^{j'-1} \mathcal{O}_i^{(j)}\right] \otimes \mathcal{D}_i^{(j')} \otimes \left\{ \bigotimes_{j=j'+1}^{q} \mathcal{O}_{i-1}^{(j)} \right\},
\end{aligned}
$$

by noting that $\mathcal{Y}_1 \supset \mathcal{Y}_2$ and their difference is $\bigotimes_{j=j'+1}^{q} \mathcal{O}_{i-1}^{(j)}$. This together with (28) proves the lemma. □

Lemma 3.1 implies that we can represent $\boldsymbol{\mathcal{Q}}_i$ as a union of q mutually exclusive events, $\boldsymbol{\mathcal{S}}_i^{(j')} - \bigcup_{h=j'+1}^{q} \boldsymbol{\mathcal{S}}_i^{(h)}$, $j' = 1, \ldots, q$. Then the simulation of $\vec{\boldsymbol{\omega}}$, conditional on $\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i$, can be carried out via the following steps. First, we can simulate an event from these mutually exclusive events, which tell us the layers (or boundaries) of each component of $\vec{\boldsymbol{\omega}}$. This step can be done via a uniform random variable simulation, if we can work out the probabilities of these mutually exclusive events. Second, conditional on the layers or boundaries for each component of $\vec{\boldsymbol{\omega}}$, the path for each component of $\vec{\boldsymbol{\omega}}$ can be simulated.

Now it is ready to introduce the following lemma.

Lemma 3.2. *Suppose that we already simulated a layer i, that is, $\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i$ and suppose that random variable U is uniformly distributed in the interval*

$$
\left[\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\vec{\boldsymbol{\omega}} \in \bigotimes_{j=1}^{q} \mathcal{O}_{i-1}^{(j)}\right), \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\vec{\boldsymbol{\omega}} \in \bigotimes_{j=1}^{q} \mathcal{O}_{i}^{(j)}\right) \right],
$$

that is, the interval

$$\left[\prod_{j=1}^{q} W_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i-1}^{(j)}\right), \prod_{j=1}^{q} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i}^{(j)}\right)\right]. \tag{30}$$

For a given value j', we define

$$\Delta_l^{(j')} = \prod_{j=1}^{j'-1} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i}^{(j)}\right) \cdot \prod_{j=j'}^{q} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i-1}^{(j)}\right)$$

$$\Delta_r^{(j')} = \prod_{j=1}^{j'} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i}^{(j)}\right) \cdot \prod_{j=j'+1}^{q} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i-1}^{(j)}\right).$$

Note that $\Delta_l^{(j')}$ and $\Delta_r^{(j')}$ depend on the layer i. For simplicity of notations, we omit the subscript i.

Then we have

$$\begin{aligned}&\mathrm{P}\left(U \in \left[\Delta_l^{(j')}, \Delta_r^{(j')}\right]\right)\\ &= \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\bigotimes_{j=1}^{j'-1}\left\{\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i}^{(j)}\right\}, \bigotimes_{j=j'+1}^{q}\left\{\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i-1}^{(j)}\right\}, \boldsymbol{\omega}^{(j')} \in \mathcal{D}_i^{(j')} | \vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\right).\end{aligned}$$

Proof. The density function of U is, with s belonging to the interval (30),

$$\begin{aligned}f_U(s) &= \left[\prod_{j=1}^{q} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i}^{(j)}\right) - \prod_{j=1}^{q} W_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i-1}^{(j)}\right)\right]^{-1}\\ &= \frac{1}{\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega} \in \boldsymbol{\mathcal{Q}}_i\right)}.\end{aligned} \tag{31}$$

We also have

$$\begin{aligned}\Delta_r^{(j')} - \Delta_l^{(j')} &= \prod_{j=1}^{j'-1} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i}^{(j)}\right) \cdot \prod_{j=j'+1}^{q} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i-1}^{(j)}\right)\\ &\quad \cdot \left[\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j')} \in \mathcal{O}_{i}^{(j')}\right) - \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j')} \in \mathcal{O}_{i-1}^{(j')}\right)\right]\\ &= \prod_{j=1}^{j'-1} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i}^{(j)}\right) \cdot \prod_{j=j'+1}^{q} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i-1}^{(j)}\right)\\ &\quad \cdot \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\left(\boldsymbol{\omega}^{(j')} \in \mathcal{D}_i^{(j')}\right).\end{aligned}$$

Therefore, $\Delta_r^{(j')} - \Delta_l^{(j')}$ corresponds to the probability of the event $\boldsymbol{\mathcal{S}}_i^{(j')} - \bigcup_{h=j'+1}^{q} \boldsymbol{\mathcal{S}}_i^{(h)}$ given in Lemma 3.1.

On the other hand, we have $\Delta_l^{(j')} \le \Delta_r^{(j')} = \Delta_l^{(j'+1)}$, $\Delta_l^{(1)} \le \cdots \le \Delta_l^{(q)}$ and $\Delta_r^{(1)} \le \cdots \le \Delta_r^{(q)}$. Using (31) and by noticing that $[\Delta_l^{(1)}, \Delta_r^{(q)}]$ is the same as the interval in (30), and $\{\Delta_l^{(1)}, \Delta_l^{(2)} = \Delta_r^{(1)}, \ldots, \Delta_l^{(j+1)} = \Delta_r^{(j)}, \ldots, \Delta_r^{(q)}\}$ forms a partition of the interval in (30), we have

$$\begin{aligned}
\mathrm{P}\big(U \in \big[\Delta_l^{(j')}, \Delta_r^{(j')}\big]\big) &= \frac{\Delta_r^{(j')} - \Delta_l^{(j')}}{\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}(\boldsymbol{\omega} \in \boldsymbol{\mathcal{Q}}_i)} \\
&= \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\Bigg(\big(\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(j'-1)}\big) \in \bigotimes_{j=1}^{j'-1} \mathcal{O}_i^{(j)}, \\
&\qquad \big(\boldsymbol{\omega}^{(j'+1)}, \ldots, \boldsymbol{\omega}^{(q)}\big) \in \bigotimes_{j=j'+1}^{q} \mathcal{O}_{i-1}^{(j)}, \boldsymbol{\omega}^{(j')} \in \mathcal{D}_i^{(j')} | \vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\Bigg).
\end{aligned}$$

□

Note that the variable U in Lemma 3.2 follows a uniform distribution in the interval $[\prod_{j=1}^{q} W_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i-1}^{(j)}), \prod_{j=1}^{q} \mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}(\boldsymbol{\omega}^{(j)} \in \mathcal{O}_i^{(j)})]$. The two boundary points of this interval are limits of certain alternating sequences [1]. Therefore, the random variable U in Lemma 3.2 can be easily simulated. Given $\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i$, if U belongs to the interval $[\Delta_l^{(j')}, \Delta_r^{(j')}]$, then we have that from Lemma 3.2

$$\begin{aligned}
&\boldsymbol{\omega}^{(j')} \in \mathcal{D}_i^{(j')}, \\
&\boldsymbol{\omega}^{(j)} \in \mathcal{O}_i^{(j)} \qquad \text{for } j < j', \\
&\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i-1}^{(j)} \qquad \text{for } j > j'
\end{aligned} \tag{32}$$

which tells us the boundaries or layers for each $\boldsymbol{\omega}^{(j)}$. Then each component $\boldsymbol{\omega}^{(j)}$ can be sampled conditional on (32), using the method in [1]. The methodology in this subsection is summarized in the algorithms provided in the following subsection.

3.2.2. *The improved algorithm*

In summary, an improved version of Algorithm 2 is given below (see Algorithm 3 and Algorithm 4).

Then further the improved version of Algorithm 1 is given by Algorithm 5.

Remark. 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 practical for complicated target distributions.

Algorithm 3: Simulation for $\mathcal{I}$, high dimensional case

Given $\widetilde{\mathbf{B}}_0 = \mathbf{x}$ and $\widetilde{\mathbf{B}}_T = \mathbf{y}$, simulate a process $\widetilde{\mathbf{B}}_t, t \in [0, T]$ via the following Steps 1a and 1b:;

// 1a: Simulate layer I with probability $\widetilde{\mathrm{P}}(I)$ given in (26) [10]

// 1b: Simulate a value $j' \in \{1, \ldots, q\}$ via Algorithm 4

// 1c: Given $\widetilde{\mathbf{B}}_0$ and $\widetilde{\mathbf{B}}_T$, simulate a sample path $\widetilde{\mathbf{B}}_t, 0 < t < T$, from $\mathbb{W}_{0,T}^{x,y}$ conditional on $\boldsymbol{\omega}^{(j')} \in \mathcal{D}_k^{(j')}, \boldsymbol{\omega}^{(j)} \in \mathcal{O}_i^{(j)}$ for $j < j'$ and $\boldsymbol{\omega}^{(j)} \in \mathcal{O}_{i-1}^{(j)}$ for $j > j'$, using the algorithm in [1]

Calculate l_I such that $l_I \le \inf_{s\in[0,T],\vec{\boldsymbol{\omega}}\in\boldsymbol{\mathcal{Q}}_I}\{(\|\boldsymbol{\alpha}\|^2 + \mathbf{div}\,\boldsymbol{\alpha})(\boldsymbol{\omega}_s)/2\}$;

Calculate r_I such that $r_I \ge \sup_{t\in[0,T],\vec{\boldsymbol{\omega}}\in\mathcal{D}_I}\{[\|\boldsymbol{\alpha}\|^2(\boldsymbol{\omega}_t) + \operatorname{div}\boldsymbol{\alpha}'(\boldsymbol{\omega}_t)]/2 - l_I\}$;

Simulate $\boldsymbol{\Psi} = \{\psi_1, \ldots, \psi_\rho\}$ uniformly distributed on $\mathbf{U}[0, T]$ and marks $\Upsilon = \{\nu_1, \ldots, \nu_\rho\}$ uniformly distributed on $\mathbf{U}[0, 1]$, where ρ is from $\mathrm{Poi}(r_I T)$;

Compute the acceptance indicator $\mathcal{I} := \prod_{j=1}^{\rho} I[r_I^{-1}[(\|\boldsymbol{\alpha}\|^2 + \mathbf{div}\,\boldsymbol{\alpha})(\boldsymbol{\omega}_{\psi_j})/2 - l_I] < \nu_j]$;

Algorithm 4: Simulation of layers (or boundaries for each component), conditional on layer i, based on Lemma 3.2

Simulate a uniform random variable U from the interval given in (30);

Find the value j', $j' \in \{1, 2, \ldots, q\}$ such that $U \in [\Delta_l^{(j')}, \Delta_r^{(j')}]$;

Output j'.

3.3. Simulating the event with probability AP_1

Algorithm 1 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

$$\begin{aligned} AP_1 &= \mathrm{P}\big\{U \le \exp\big[-\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T\|^2/(2T)\big]\big\} \\ &= \mathrm{P}\big\{\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T\|^2 \le -2T\log(U)\big\}, \end{aligned} \tag{33}$$

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: $E_{g_1} = \int \mathbf{x} g_1(\mathbf{x})\, d\mathbf{x}$ and $E_{g_2} = \int \mathbf{x} g_2(\mathbf{x})\, d\mathbf{x}$. Then we have the following lemma.

Algorithm 5: Rejection sampling for $f \propto g_1 g_2 = f_1 f_2$

Simulate $\boldsymbol{\omega}_0$ from g_1 and $\boldsymbol{\omega}_T$ from g_2, and a standard uniform variable U;
if $U \leq \exp[-\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T\|^2/(2T)]$ **then**
  $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$ is from h;
else
  return to Step 1;
end
Simulate a standard uniform variable U' ;
if $U' \leq \sum_{i=1}^{\infty} \exp\{-T(l_i - l)\}\mathbb{W}_{0,T}^{\boldsymbol{\omega}_0,\boldsymbol{\omega}_T}\{\vec{\boldsymbol{\omega}} \in \boldsymbol{\mathcal{Q}}_i\}$ **then**
  Simulate the reweighted layered Brownian bridge $\bar{\mathbf{B}} = \{\boldsymbol{\omega}_t, t \in (0, T)\}$ conditional on $(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$;
  Simulate $\mathcal{I} = 1$ with probability given by (22) ;
  `// The above two steps are based on Algorithm 3`
  if $\mathcal{I} = 1$ **then**
    Output $\boldsymbol{\omega}_T$;
  else
    return to Step 1;
  end
else
  return to Step 1.
end

Lemma 3.3. *Define* $\mathcal{A} = \{(g_1, g_2) : \text{ such that } f = g_1 \cdot g_2 \text{ and } E_{g_1} = E_{g_2}\}$. *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* $E\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T\|^2$ *reaches the minimum when* $g_1(\cdot) = g_2(\cdot) = \sqrt{f(\cdot)}$.

Proof. We have

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

which reaches the minimum when $g_1(\mathbf{x}) = f(\mathbf{x})/g_1(\mathbf{x})$, that is, $g_1 = g_2 = \sqrt{f}$. □

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, $f = \prod_{l=1}^{\iota} g_l$, 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, that is, simulate samples from $\prod_{j=l}^{\iota} g_j$, l from $\iota - 1$ to 1. This is given by Algorithm 6.

Note that in the *for* loop of Algorithm 6, 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 6 simulates $\{\mathbf{x}_l\}_{l=1}^{\iota-1}$ and $\mathbf{y}$ independently. The proposal $\mathbf{y}$ will be accepted if $U_l \le \exp(-\|\mathbf{x}_l - \mathbf{y}\|^2/(2T))$ and $\mathcal{I}_l = 1$ for $l = 1, \ldots, \iota - 1$. Since simulating the event $\mathcal{I}_l = 1$ using [1,2] or the more efficient Algorithm 3 is usually complicated and time consuming, we can revise Algorithm 6 as follows to increase the efficiency: First, simulate $\mathbf{x}_l$, $l = 1, \ldots, \iota - 1$ and $\mathbf{y}$; second, check if $U_l \le \exp(-\|\mathbf{x}_l - \mathbf{y}\|^2/(2T))$ for $l = 1, \ldots, \iota - 1$; third, simulate $I_l = 1$, for $l = 1, \ldots, \iota - 1$. The revised algorithm is given below (see Algorithm 7).

Note that Algorithm 6 and Algorithm 7 can be improved via the methods in Section 3.

Algorithm 6: Rejection sampling for $f = \prod_{l=1}^{\iota} g_l$

```
Simulate y from g_ι;
for l ← ι − 1 to 1 do
    Simulate x_l from g_l;
    Simulate standard uniform variable U_l;
    if U_l > exp(−‖x_l − y‖²/(2T)) then
        Goto Step 1;
    end
    Simulate the Brownian bridge B̄ = {ω_t, t ∈ (0, T)} given (ω_0 = x_l, ω_T = y);
    Simulate I_l = 1 with probability given by (10), with α(x) = ∇A(x) and
    A(x) = log g_l(x);
    if I_l = 1 then
        y can be viewed as a sample from ∏_{j=l}^{ι} g_j;
    else
        Goto Step 1;
    end
end
Output y.
```

Algorithm 7: Revised rejection sampling for $f = \prod_{l=1}^{\iota} g_\iota$

Simulate $\mathbf{y}$ from $g_{(\iota)}$;
for $l \leftarrow \iota - 1$ **to** 1 **do**
  Simulate $\mathbf{x}_l$ from $g_{(l)}$;
  Simulate standard uniform variable U_l;
  if $U_l > \exp(-\|\mathbf{x}_l - \mathbf{y}\|^2/(2T))$ **then**
    Goto Step 1
  end
end
for $l \leftarrow \iota - 1$ **to** 1 **do**
  Simulate the Brownian bridge $\bar{\mathbf{B}} = \{\boldsymbol{\omega}_t, t \in (0, T)\}$ given $(\boldsymbol{\omega}_0 = \mathbf{x}_l, \boldsymbol{\omega}_T = \mathbf{y})$;
  Simulate $\mathcal{I}_l = 1$ with probability given by (10), with $\boldsymbol{\alpha}(\mathbf{x}) = \nabla A(\mathbf{x})$ and $A(\mathbf{x}) = \log g_{(l)}(\mathbf{x})$;
  if $\mathcal{I}_l = 0$ **then**
    Goto Step 1;
  end
end
Output $\mathbf{y}$;

5. Application – Exact Monte Carlo simulation for finite mixture models

5.1. The finite mixture models and existing methods

Consider the following mixture of normal densities, where the data $\{z_i, i = 1, \ldots, n\}$ are from the finite mixture of normal densities

$$h(z_i; \boldsymbol{\Theta}) = \sum_{k=1}^{K} p_k h_k(z_i; \theta_k, \nu), h_k(z_i; \theta_k, \nu) = |\nu| e^{-(\nu^2/2)(z_i - \theta_k)^2} \tag{34}$$

with $\boldsymbol{\Theta} = (\mathbf{p}, \nu, \boldsymbol{\theta})$, where ν and θ_k range in $\mathbf{R}$ and p_k is the component proportion with $0 \le p_k \le 1$ and $\sum_k p_k = 1$. Such mixture models provide a statistical description of data obtained when sampling successively from randomly selected sub-populations. These models arise naturally in the areas of statistical classification and clustering. More detailed introduction about mixture models can be found in [22].

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

$$\pi_0(\boldsymbol{\Theta}) \propto |\nu|^{2a-1} e^{-b\nu^2} |\nu|^K \prod_{k=1}^{K} \left[e^{-(\sigma_k \nu^2/2)(\theta_k - \mu_k)^2} p_k^{\varrho_k - 1} \right], \tag{35}$$

where $(\sigma_k, \mu_k, \varrho_k, a, b)$ is known. We focus on (35) 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 in [23]. Based on (35), the posterior distribution can be written as

$$f(\boldsymbol{\Theta}) \propto \prod_{i=1}^{n}\left[\sum_{k=1}^{K} p_k h_k(z_i;\boldsymbol{\Theta})\right]\pi_0(\boldsymbol{\Theta}). \tag{36}$$

A Gibbs sampler for (36) is readily available in [12] and a more general sampler is available in [27] with K being unknown. 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 [14]. Celeux et al. [6] 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 (36).

Note that for certain simpler versions of (34), exact simulation from its posterior is readily available. For example for the simple mixture models with known $\boldsymbol{\Theta}$, methods in [19] and [13] and the method of adaptive rejection sampling for log-concave densities base on the algorithm in [21] 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 [7] 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 $\boldsymbol{\Theta}$. Casella et al. [5] investigate the use of a CFTP algorithm, called the perfect slice sampler, to simulate from (36), but this method only works theoretically.

5.2. Reparameterisation and density function decomposition

To the proposed method in this paper, we first consider the following parameter transformation $\boldsymbol{\Theta} = \boldsymbol{\Theta}(\mathbf{x})$, where $\mathbf{x} = (\mathbf{u}, \eta, \boldsymbol{\delta})$,

$$\begin{aligned} p_k &:= p_k(\mathbf{x}) = \frac{e^{u_k}}{\sum_{k=1}^{K-1} e^{u_k} + 1}, \qquad k = 1, \ldots, K-1, \\ \nu &:= \nu(\mathbf{x}) = \eta, \\ \theta_k &:= \theta_k(\mathbf{x}) = \delta_k \eta^{-1}, \qquad k = 1, \ldots, K. \end{aligned} \tag{37}$$

We consider such a transformation to make the support of the posterior $f(\mathbf{x})$ to be $\mathbf{R}^q$, where $q = 2K$.

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

$$\hbar_k(z_i;\mathbf{x}) = e^{-(1/2)(z_i\eta-\delta_k)^2}, \qquad \hbar(z_i;\mathbf{x}) = \sum_{k=1}^{K} p_k(\mathbf{x})\hbar_k(z_i;\mathbf{x}) \tag{38}$$

the posterior distribution becomes

$$\begin{aligned} f(\mathbf{x}) &\propto |\eta|^n \prod_{i=1}^{n}\left[\sum_{k=1}^{K} p_k(\mathbf{x})\hbar_k(z_i;\mathbf{x})\right]\pi_0\big(\boldsymbol{\Theta}(\mathbf{x})\big)\cdot J(\mathbf{x}) \\ &= |\eta|^n \prod_{i=1}^{n}\left[\sum_{k=1}^{K} p_k(\mathbf{x}) e^{-(1/2)(z_i\eta-\delta_k)^2}\right] \\ &\quad\times\left\{|\eta|^{2a-1}e^{-b\eta^2}\left[\prod_{k=1}^{K} e^{-(\sigma_k/2)(\delta_k-\eta\mu_k)^2} p_k(\mathbf{x})^{\varrho_k}\right]\right\}, \end{aligned} \tag{39}$$

where the Jacobin $J(\mathbf{x}) = [\prod_{k=1}^{K} p_k(\mathbf{x})]\cdot|\eta|^{-K}$ (see Section 2.5 for the Jacobin term related to the transformation from $\mathbf{p}$ to $\mathbf{x}$).

5.2.1. *A decomposition of f, which only works theoretically*

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

$$\begin{aligned} f(\mathbf{x}) &\propto \left\{\prod_{i=1}^{n'}\left[\sum_{k=1}^{K} p_k(\mathbf{x}) e^{-(1/2)(z_i\eta-\delta_k)^2}\right]\right\} \\ &\quad\times\left\{e^{-b\eta^2/2}\left[\prod_{k=1}^{K} e^{-(\sigma_k/4)(\delta_k-\eta\mu_k)^2} p_k(\mathbf{x})^{\varrho_k/2}\right]\right\} \\ &\quad\cdot|\eta|^{n+2a-1}\left\{\prod_{i=n'+1}^{n}\left[\sum_{k=1}^{K} p_k(\mathbf{x}) e^{-(1/2)(z_i\eta-\delta_k)^2}\right]\right\} \\ &\quad\times\left\{e^{-b\eta^2/2}\left[\prod_{k=1}^{K} e^{-(\sigma_k/4)(\delta_k-\eta\mu_k)^2} p_k(\mathbf{x})^{\varrho_k/2}\right]\right\}. \end{aligned} \tag{40}$$

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 $\mathbf{x}$ from g_1 and $\mathbf{y}$ from g_2, such a decomposition will not work. The reason is that g_1 is not close to g_2 and the simulated $\mathbf{x}$ and $\mathbf{y}$ are almost always far away from each other. This makes the probability AP_1 very small. For this reason, we consider the following decomposition.

5.2.2. *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 (39), we consider the following hat function

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

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 $l = 1, \ldots, \iota$. Then the posterior density can be decomposed as a product of ι terms, $\hat{f}(\mathbf{x}) \propto \prod_{l=1}^{\iota} g_l(\mathbf{x})$, with

$$\begin{aligned} g_l(\mathbf{x}) = {} & \left(\eta^2 + c\right)^{m_l/2+(m_l/2n)(2a-1)}\left[\prod_{i=n_{l-1}+1}^{n_l} \hbar(z_i;\mathbf{x})\right] \\ & \cdot \left\{e^{-(m_l/n)b\eta^2}\prod_{k=1}^{K} e^{-(\sigma_k m_l/(2n))(\delta_k-\eta\mu_k)^2} p_k(\mathbf{x})^{m_l\varrho_k/n}\right\}, \qquad l = 1, \ldots, \iota. \end{aligned} \tag{41}$$

5.3. 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

$$\begin{aligned} g_l(\mathbf{x}) = {} & \left(\eta^2 + c\right)^{m_l/2+(m_l/2n)(2a-1)} e^{-(m_l/n)b\eta^2} \\ & \cdot \sum_{\breve{\mathbf{n}}_l, \bar{\mathbf{Z}}_l, Z_l^2}\left\{\prod_{k=1}^{K} p_k(\mathbf{x})^{\breve{n}_{l,k}+m_j\varrho_k/n}\right. \\ & \times \exp\left[-\frac{1}{2}\sum_k\left(\breve{n}_{l,k} + \frac{\sigma_k m_l}{n}\right)\left[\delta_k - \frac{\breve{n}_{l,k}\bar{Z}_{l,k} + (\sigma_k m_l\mu_k)/n}{\breve{n}_{l,k} + (\sigma_k m_l)/n}\eta\right]^2\right] \\ & \left.\times \exp\left[-\frac{1}{2}\left(Z_l^2 + \sum_k\left(\frac{\sigma_k m_l}{n}\mu_k^2 - \frac{(\breve{n}_{l,k}\bar{Z}_{l,k} + (\sigma_k m_l\mu_k)/n)^2}{\breve{n}_{l,k} + (\sigma_k m_l)/n}\right)\right)\eta^2\right]\right\}. \end{aligned} \tag{42}$$

Note that we can simulate from (42) directly when K and m_j are small, that is, when the number of statistics $(\check{\mathbf{n}}_l, \bar{\mathbf{Z}}_l, Z_l^2)$ is not large. For example, when the mixture model has $K = 2$ components and $m_l = 25$, the number of different $(\check{\mathbf{n}}_l, \bar{\mathbf{Z}}_l, Z_l^2)$ is about $2^{25} = 33\,554\,432$ terms, which can be dealt with by a standard desktop.

To simulate $\mathbf{x}$ from (42), the only challenge is to do the integration for η over $g_l(\mathbf{x})$. This can be done via a recursive approach. The method of simulation from g_l is given in Section 1 of the supplementary file [11].

5.4. Simulate diffusions with invariant distribution $g_l(\mathbf{x})$

To use Algorithm 7, we also need to simulate the multivariate diffusion, having g_l as the invariant distribution,

$$d\mathbf{X}_t^{(l)} = \boldsymbol{\alpha}^{(l)}\big(\mathbf{X}_t^{(l)}\big)\,dt + d\mathbf{B}_t^{(l)}, \tag{43}$$

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

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

$$\begin{aligned}
\alpha_{u_k}^{(l)}(\mathbf{x}) &= \frac{\partial \log g_l(\mathbf{x})}{\partial u_k} = \frac{\partial g_l(\mathbf{x})/\partial u_k}{g_l(\mathbf{x})}, \qquad \alpha_{\eta}^{(l)}(\mathbf{x}) = \frac{\partial \log g_l(\mathbf{x})}{\partial \eta} = \frac{\partial g_l(\mathbf{x})/\partial \eta}{g_l(\mathbf{x})}, \\
\alpha_{\delta_k}^{(l)}(\mathbf{x}) &= \frac{\partial \log g_l(\mathbf{x})}{\partial \delta_k} = \frac{\partial g_l(\mathbf{x})/\partial \delta_k}{g_l(\mathbf{x})}.
\end{aligned} \tag{44}$$

According to

$$\operatorname{div} \boldsymbol{\alpha}^{(l)}(\mathbf{x}) = \sum_{k=1}^{K-1} \frac{\partial \alpha_{u_k}^{(l)}(\mathbf{x})}{\partial u_k} + \frac{\partial \alpha_{\eta}^{(l)}(\mathbf{x})}{\partial \eta} + \sum_{k=1}^{K} \frac{\partial \alpha_{\delta_k}^{(l)}(\mathbf{x})}{\partial \delta_k} \tag{45}$$

we further have

$$\begin{aligned}
&\big\|\boldsymbol{\alpha}^{(l)}(\mathbf{x})\big\|^2 + \operatorname{div} \boldsymbol{\alpha}^{(l)}(\mathbf{x}) \\
&= \sum_{k=1}^{K-1} \frac{\partial^2 g_l(\mathbf{x})/\partial u_k^2}{g_l(\mathbf{x})} + \frac{\partial^2 g_l(\mathbf{x})/\partial \eta^2}{g_l(\mathbf{x})} + \sum_{k=1}^{K} \frac{\partial^2 g_l(\mathbf{x})/\partial \delta_k^2}{g_l(\mathbf{x})}
\end{aligned} \tag{46}$$

whose expression can be found in the supplementary file [11]. We also show that (46) is bounded below in Section 3 of the supplementary file [11].

Note that we also need to find the upper bound (required by Algorithm 2) and lower bounds (required by (21)) for $\|\boldsymbol{\alpha}(\boldsymbol{\omega}_t)\|^2 + \operatorname{div} \boldsymbol{\alpha}(\boldsymbol{\omega}_s)$ under each layer $\mathcal{Q}_i$. This is also straightforward and the details are provided in Section 2 of the supplementary file [11].

5.5. Simulation results for mixture models

We consider a mixture model with 2 components, $h(z_i; \mathbf{\Theta}) = \sum_{k=1}^{2} p_k \mathcal{N}(\theta_k, \nu^{-2})$, 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 (35) 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 (e.g., see (41)). We partition the samples into two groups in the following way: first we order the samples to $z_{(1)}, z_{(2)}, \ldots, z_{(19)}, z_{(20)}$ and then the first group is $\{z_{(2k-1)}, k = 1, \ldots, 10\}$ and the second group is $\{z_{(2k)}, k = 1, \ldots, 10\}$. 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 Table 1 and Table 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 (41), 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.

For model and priors mentioned above in the beginning of Section 5.5, 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.

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 (e.g., 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 $\mathbf{x} \sim \hat{f}$ in (41) to be accepted as $\mathbf{x} \sim f$ in (39), is denoted as AP_3. The value c used in (41) 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

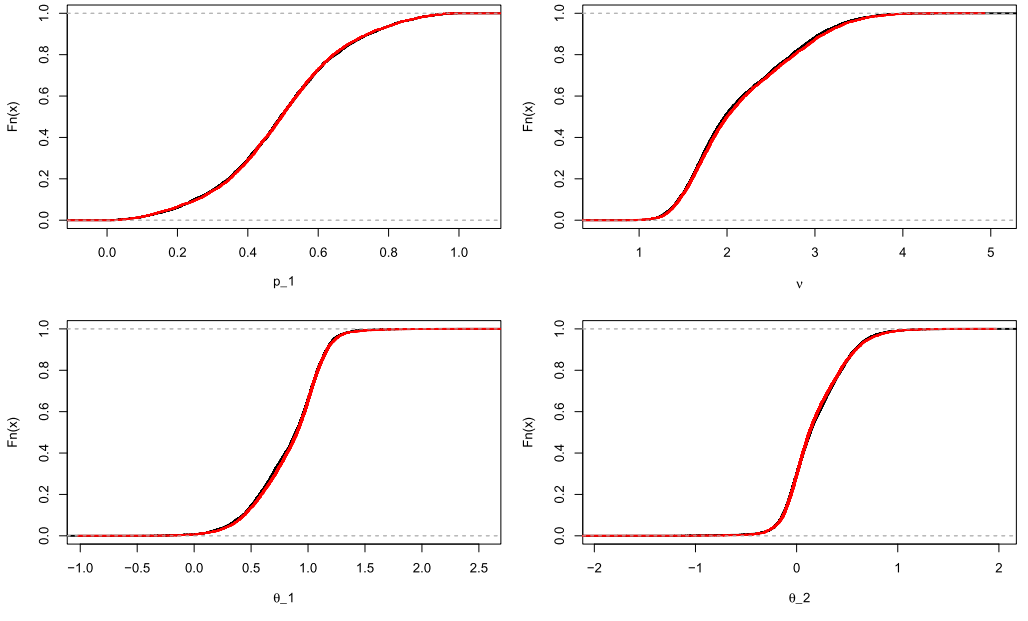

Figure 2. Comparison with direct sampling; two methods output almost the same results.

efficient. From the lower bound (7) in the supplementary file [11], we can see that the smaller values of c, the smaller lower bound for $(\|\boldsymbol{\alpha}\|^2 + \boldsymbol{\alpha}')(\cdot)$ that is, 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 $c = 0.03$.

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	7.8e−5	3.0e−4	0.0006
$a_i = 0.15$, time	244 s	246 s	284 s
$a_i = 0.15$, (ii)	0.821	0.421	0.331
$a_i = 0.60$, time	337 s	310 s	474 s
$a_i = 0.60$, (ii)	0.738	0.409	0.268

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	284 s	275 s	503 s
$a_i = 0.15$, (ii)	0.656	0.357	0.211
$a_i = 0.60$, time	355 s	375 s	589 s
$a_i = 0.60$, (ii)	0.564	0.334	0.171

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 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 [10] 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.

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. We demonstrate this using the mixture models, for which no other practical methods are available. The new method proposed in this paper transfers the difficulty of finding the hat function to finding the lower bounds of $\|\boldsymbol{\alpha}\|^2(\boldsymbol{\omega}_s) + \operatorname{div}\boldsymbol{\alpha}(\boldsymbol{\omega}_s)$. 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 and it leads to the following possible future works. When simulating 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 $\exp(-\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T\|^2/(2T))$.

When sampling from the posterior of finite mixture model, we actually applied the new method to the hat function (41), since the log-transformation of the target function is not differentiable. By using the hat function (41) Condition 2.1 is satisfied, but we need an extra acceptance/rejection step to draw a sample from the target function (39). 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 could possibly apply the new method directly on the target distribution (39). We also leave this as a future work.

Acknowledgements

The author thanks the two reviewers and the Editor for their valuable comments and suggestions which improved significantly the final form of the paper.

Supplementary Material

Supplement to: "A new rejection sampling method without using hat function" (DOI: 10.3150/16-BEJ814SUPP; .pdf). The supplement file includes some necessary proofs for applying the proposed method to the simulation from the posterior for Gaussian mixture models.

References

[1] 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. MR2394037

[2] Beskos, A., Papaspiliopoulos, O., Roberts, G.O. and Fearnhead, P. (2006). Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **68** 333–382. With discussions and a reply by the authors. MR2278331

[3] Beskos, A. and Roberts, G.O. (2005). Exact simulation of diffusions. *Ann. Appl. Probab.* **15** 2422–2444. MR2187299
[4] Breyer, L.A. and Roberts, G.O. (2001). Catalytic perfect simulation. *Methodol. Comput. Appl. Probab.* **3** 161–177. MR1868568
[5] Casella, G., Mengersen, K.L., Robert, C.P. and Titterington, D.M. (2002). Perfect samplers for mixtures of distributions. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **64** 777–790. MR1979386
[6] Celeux, G., Hurn, M. and Robert, C.P. (2000). Computational and inferential difficulties with mixture posterior distributions. *J. Amer. Statist. Assoc.* **95** 957–970. MR1804450
[7] Dai, H. (2007). Perfect simulation methods for Bayesian applications. Ph.D. thesis, University of Oxford. Supervisor: Peter Clifford.
[8] Dai, H. (2008). Perfect sampling methods for random forests. *Adv. in Appl. Probab.* **40** 897–917. MR2454038
[9] Dai, H. (2011). Exact Monte Carlo simulation for fork-join networks. *Adv. in Appl. Probab.* **43** 484–503. MR2848387
[10] Dai, H. (2014). Exact simulation for diffusion bridges: An adaptive approach. *J. Appl. Probab.* **51** 346–358. MR3217771
[11] Dai, H. (2016). Supplement to "A new rejection sampling method without using hat function." DOI:10.3150/16-BEJ814SUPP.
[12] Diebolt, J. and Robert, C.P. (1994). Estimation of finite mixture distributions through Bayesian sampling. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **56** 363–375. MR1281940
[13] Fearnhead, P. (2005). Direct simulation for discrete mixture distributions. *Stat. Comput.* **15** 125–133. MR2137276
[14] Fearnhead, P. and Meligkotsidou, L. (2007). Filtering methods for mixture models. *J. Comput. Graph. Statist.* **16** 586–607. MR2351081
[15] Fill, J.A. (1998). An interruptible algorithm for perfect sampling via Markov chains. *Ann. Appl. Probab.* **8** 131–162. MR1620346
[16] Fill, J.A., Machida, M., Murdoch, D.J. and Rosenthal, J.S. (2000). Extension of Fill's perfect rejection sampling algorithm to general chains. In *Proceedings of the Ninth International Conference "Random Structures and Algorithms"* (*Poznan*, 1999) **17** 290–316. MR1801136
[17] Gilks, W.R. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Appl. Statist.* **41** 337–348.
[18] Hansen, N.R. (2003). Geometric ergodicity of discrete-time approximations to multivariate diffusions. *Bernoulli* **9** 725–743. MR1996277
[19] Hobert, J., Robert, C. and Titterington, D. (1999). On perfect simulation for some mixture of distributions. *Stat. Comput.* **9** 287–298.
[20] Huber, M. (2004). Perfect sampling using bounding chains. *Ann. Appl. Probab.* **14** 734–753. MR2052900
[21] Leydold, J. (1998). A rejection technique for sampling from log-concave multivariate distributions. *ACM Trans. Model. Comput. Simul.* **8** 254–280.
[22] McLachlan, G. and Peel, D. (2000). *Finite Mixture Models*. *Wiley Series in Probability and Statistics*: *Applied Probability and Statistics*. New York: Wiley. MR1789474
[23] Mengersen, K.L. and Robert, C.P. (1996). Testing for mixtures: A Bayesian entropic approach. In *Bayesian Statistics* (*Alicante*, 1994) **5**. *Oxford Sci. Publ.* 255–276. New York: Oxford Univ. Press. MR1425410
[24] Mira, A., Møller, J. and Roberts, G.O. (2001). Perfect slice samplers. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **63** 593–606. MR1858405
[25] Some Monte Carlo methods for jump diffusions. Ph.D. thesis, ProQuest LLC, Ann Arbor, MI, University of Warwick, UK. MR3389378

[26] Propp, J.G. and Wilson, D.B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. In *Proceedings of the Seventh International Conference on Random Structures and Algorithms* (*Atlanta*, *GA*, 1995) **9** 223–252. MR1611693

[27] Richardson, S. and Green, P.J. (1997). On Bayesian analysis of mixtures with an unknown number of components. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **59** 731–792. MR1483213

[28] Wilson, D.B. (2000). How to couple from the past using a read-once source of randomness. *Random Structures Algorithms* **16** 85–113. MR1728354

Received May 2014 and revised October 2015