J. Appl. Prob. **56,** 174–191 (2019)
*doi:*10.1017/jpr.2019.12
© *Applied Probability Trust* 2019

MONTE CARLO FUSION

HONGSHENG DAI,* *University of Essex*
MURRAY POLLOCK ** *** AND
GARETH ROBERTS, ** **** *University of Warwick*

Abstract

In this paper we propose a new theory and methodology to tackle the problem of unifying Monte Carlo samples from distributed densities into a single Monte Carlo draw from the target density. This surprisingly challenging problem arises in many settings (for instance, expert elicitation, multiview learning, distributed 'big data' problems, etc.), but to date the framework and methodology proposed in this paper (Monte Carlo fusion) is the first general approach which avoids any form of approximation error in obtaining the unified inference. In this paper we focus on the key theoretical underpinnings of this new methodology, and simple (direct) Monte Carlo interpretations of the theory. There is considerable scope to tailor the theory introduced in this paper to particular application settings (such as the big data setting), construct efficient parallelised schemes, understand the approximation and computational efficiencies of other such unification paradigms, and explore new theoretical and methodological directions.

Keywords: Fork-and-join; fusion; Langevin diffusion; Monte Carlo

2010 Mathematics Subject Classification: Primary 65C05; 65C60
Secondary 62C10; 65C30

1. Introduction

A common problem arising in statistical inference is the need to unify distributed analyses and inferences on shared parameters from multiple sources into a single coherent inference. This unification (or what we term 'fusion') problem can arise either explicitly due to the nature of a particular application, or artificially as a consequence of the approach a practitioner takes to tackling an application.

Typically, there will exist no closed-form analytical approach to unifying distributed inferences, and so we focus on a Monte Carlo approach. Stated generally, in this paper we are interested in sampling (without error) the d-dimensional (fusion) target density

$$f(\boldsymbol{x}) \propto f_1(\boldsymbol{x}) \cdots f_C(\boldsymbol{x}), \tag{1}$$

where each $f_c(\boldsymbol{x})$ ($c \in \{1, \dots, C\}$) is a density (up to a multiplicative constant) representing one of the C distributed inferences we wish to unify. Each $f_c(\boldsymbol{x})$ (which we term a sub-posterior) may in practice itself be represented by a Monte Carlo sample ($\tilde{f}_c(\boldsymbol{x})$), and in this paper we assume that we are able to sample (directly and exactly) from each $f_c(\boldsymbol{x})$.

Received 6 February 2018; revision received 15 November 2018.
The supplementary material for this article can be found at http://doi.org/10.1017/jpr.2019.12.

* Postal address: Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, UK. Email address: hdaia@essex.ac.uk
** Postal address: Department of Statistics, University of Warwick, Gibbet Hill Road, Coventry, CV4 7ES, UK.
*** Email address: m.pollock@warwick.ac.uk
**** Email address: gareth.o.roberts@warwick.ac.uk

Specific examples of this problem arising naturally in an application include expert elicitation ([2] and [13]), in which the (distributional) views of multiple experts on a topic (or set of parameters) have to be pooled into a single view before a decision-maker can make an informed decision; and, multiview learning ([29] and [16]) and meta-analysis ([12] and [24]), in which an interpretation could be that we are synthesising multiple inferences on a particular parameter set (computed on datasets which may or may not be of the same type), but the underlying raw data is not directly available for the unified inference. Obtaining the raw data may itself be an insoluble problem due to reasons including the nature of the original publication, data confidentiality, or simply time and storage constraints.

This fusion problem also arises artificially in a number of settings, particularly within modern statistical methodologies tackling 'big data'. The computational cost of algorithms such as the Metropolis–Hastings, which is an iterative algorithm requiring full access at every iteration to the full dataset, scale poorly with increasing volumes of data unless a modification is found.

One common modification in light of the challenge of big data is to deploy a 'divide-and-conquer' approach (or more accurately termed 'fork-and-join' approach ([26])). In this setting the full dataset is artificially split into a large number of smaller data sets, inference is then conducted on each smaller data set in isolation, and the resulting inferences are unified (see, for instance, [1], [15], [18], [19], [23], [25] and [28]). The rationale for such an approach is that inference on each small data set can be conducted independently, and in parallel, and so one could exploit large clusters of computing cores to greatly reduce the elapsed time to conduct the full inference. The weakness of these approaches is that the fusion of the separately conducted inferences is inexact. It should be noted that divide-and-conquer methodologies will typically have additional constraints due to hardware concerns—such as minimising or removing any communication between computing cores to reduce the effect of *latency*. We focus in this paper on the general fusion problem, and so do not fully address the problem in the context of big data (to which we return in subsequent work).

The framework and methodology we outline in this paper (Monte Carlo fusion) for sampling exactly from (1) can be viewed as a simple rejection sampling scheme on an extended space—we develop and sample from efficient proposal densities for (1), the samples from which we retain according to an appropriate acceptance probability. The mathematical complication in this paper is in computing the intractable acceptance probability—which requires the auxiliary simulation of collections of Brownian (or Ornstein–Uhlenbeck) bridges. Our fusion approach provides a principled way to understand the error in existing unification schemes, using a simple linear combination and correction of Monte Carlo samples (analogous to a traditional meta-analysis approach).

The presentation of this paper broadly follows this pedagogy. In Section 2 we present a density on an extended space which admits (1) as a marginal. In the remainder of the paper we then develop a rejection sampler for the extended density in Section 2. In Section 3 we develop general theory and methodology for sampling (1) based on a collection of independent Brownian bridges. In Section 4 we present a modification of the theory developed in Section 3 using Ornstein–Uhlenbeck bridges, resulting in sampling efficiencies for the particular (common) setting in which the fusion density is believed to be approximately Gaussian. In Section 5 we consider examples of our methodology applied to both light-tailed and heavy-tailed fusion target densities. Finally, in Section 6 we conclude by discussing the exciting new research directions possible using Monte Carlo fusion. Much of the technical detail in the paper is suppressed for ease of reading, but can be found in the appendices.

2. An extended fusion density

Consider $f(\boldsymbol{x})$ and $f_c(\boldsymbol{x})$ as described in (1), where $f(\boldsymbol{x})$ is integrable and we can sample from the density proportional to $f_c(\boldsymbol{x})$.

The following simple observation will form the foundation of our approach. Suppose that $p_c(\boldsymbol{y} \mid \boldsymbol{x})$ is the transition density (with respect to the Lebesgue measure) of a Markov chain on $\mathbb{R}^d$ with invariant density proportional to f_c^2.

Proposition 1. *The* $(C+1)d$*-dimensional (fusion) density proportional to the integrable function*

$$g(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y}) = \prod_{c=1}^{C} \left[f_c^2(\boldsymbol{x}^{(c)}) \cdot p_c(\boldsymbol{y} \mid \boldsymbol{x}^{(c)}) \cdot \frac{1}{f_c(\boldsymbol{y})} \right], \tag{2}$$

admits the marginal density f *for* $\boldsymbol{y}$.

The proof of Proposition 1 is elementary and is omitted. The statistical interpretation of Proposition 1 is simply if it were possible to sample from the $(C+1)d$-dimensional (fusion) density g in (2) then, as a by-product, we would obtain a draw from our fusion target density f in (1). How to directly sample from (2) is not clear, even if it were possible to simulate from $p_c(\cdot \mid \boldsymbol{x})$. Our strategy instead will be to use rejection sampling. Two rejection sampling methods (which have differing efficiencies) are provided in Sections 3 and 4.

3. A fusion rejection sampler using Brownian bridges

3.1. The methodology

Consider the proposal density for the extended fusion target (2) proportional to the function

$$h^{\mathrm{bm}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y}) = \prod_{c=1}^{C} [f_c(\boldsymbol{x}^{(c)})] \exp\left\{ -\frac{C\|\boldsymbol{y} - \bar{\boldsymbol{x}}\|^2}{2T} \right\}, \tag{3}$$

where $\bar{\boldsymbol{x}} = C^{-1} \sum_{c=1}^{C} \boldsymbol{x}^{(c)}$, and T is an arbitrary positive constant.

Simulation from the proposal h^{bm} can be achieved directly. In particular, $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}$ are first drawn independently from $f_1, \dots, f_C$, respectively, and then $\boldsymbol{y}$ is simply a Gaussian random variable centred on $\bar{\boldsymbol{x}}$.

In Proposition 1 we presented a general form of the extended fusion target, in which $p_c(\boldsymbol{y} \mid \boldsymbol{x})$ is the transition density (with respect to the Lebesgue measure) of a Markov chain on $\mathbb{R}^d$ with invariant density proportional to f_c^2. In this paper we set $p_c(\boldsymbol{y} \mid \boldsymbol{x}) := p^{\mathrm{dl}}_{T,c}(\boldsymbol{y} \mid \boldsymbol{x})$, the transition density of a double Langevin diffusion for f_c (i.e. the transition density of a Langevin diffusion for f_c^2) from $\boldsymbol{x}$ to $\boldsymbol{y}$ over a predefined (user-specified) time $T > 0$. To distinguish the resulting extended fusion target from the general case, we further denote the extended fusion target by g^{dl}. In particular, for all $1 \le c \le C$, we consider the d-dimensional (double) Langevin (DL) diffusion processes $\mathfrak{X} = \{\boldsymbol{X}_t^{(c)},\ t \in [0, T],\ c = 1, \dots, C\}$, given by

$$\mathrm{d}\boldsymbol{X}_t^{(c)} = \nabla A_c^{\mathrm{dl}}(\boldsymbol{X}_t^{(c)})\,\mathrm{d}t + \mathrm{d}\boldsymbol{W}_t^{(c)}, \tag{4}$$

where $\boldsymbol{W}_t^{(c)}$ is a d-dimensional Brownian motion, ∇ is the gradient operator over $\boldsymbol{x}$, and

$$A_c^{\mathrm{dl}}(\boldsymbol{x}) := \log f_c(\boldsymbol{x});$$

$\boldsymbol{X}_t^{(c)}$ has invariant distribution $f_c^2(\boldsymbol{x})$ *for any* $t \in [0, T]$ ([14]). We also impose the following standard regularity property (where div denotes the divergence operator)

Condition 1. *Define*

$$\phi_c^{\mathrm{dl}}(\boldsymbol{x}) := \tfrac{1}{2}(\|\nabla A_c^{\mathrm{dl}}(\boldsymbol{x})\|^2 + \operatorname{div} \nabla A_c^{\mathrm{dl}}(\boldsymbol{x})).$$

There exists constant $\Phi_c^{\mathrm{bm}} > -\infty$ *such that, for all* $\boldsymbol{x}$ *and each* $c \in \{1, \ldots, C\}$, $\phi_c^{\mathrm{dl}}(\boldsymbol{x}) \geq \Phi_c^{\mathrm{bm}}$.

Then we have the following proposition which gives a rejection sampling method for $g^{\mathrm{dl}}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y})$.

Proposition 2. *Under Condition 1, we can write*

$$\frac{g^{\mathrm{dl}}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y})}{h^{\mathrm{bm}}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y})} = \left[\frac{\sqrt{C}}{\sqrt{2\pi T}}\right]^C \times \rho^{\mathrm{bm}} \times Q^{\mathrm{bm}} \times \prod_{c=1}^{C} \mathrm{e}^{-T\Phi_c^{\mathrm{bm}}}, \tag{5}$$

where

$$\rho^{\mathrm{bm}} := \rho^{\mathrm{bm}}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}) = \mathrm{e}^{-C\sigma^2/2T}, \qquad \sigma^2 = C^{-1} \sum_{c=1}^{C} \|\boldsymbol{x}^{(c)} - \bar{\boldsymbol{x}}\|^2,$$

and

$$Q^{\mathrm{bm}} = \mathbb{E}_{\overline{\mathbb{W}}}(E^{\mathrm{bm}}),$$

with $\overline{\mathbb{W}}$ *denoting the law of* C *Brownian bridges* $\boldsymbol{x}_t^{(1)}, \ldots, \boldsymbol{x}_t^{(C)}$ *with* $\boldsymbol{x}_0^{(c)} = \boldsymbol{x}^{(c)}$ *and* $\boldsymbol{x}_T^{(c)} = \boldsymbol{y}$ *in the time interval* $[0, T]$ *(noting that these Brownian bridges are independent conditional on the starting and ending points), and*

$$E^{\mathrm{bm}} := \prod_{c=1}^{C} \left[\exp\left\{-\int_0^T (\phi_c^{\mathrm{dl}}(\boldsymbol{x}_t^{(c)}) - \Phi_c^{\mathrm{bm}})\, \mathrm{d}t\right\}\right]. \tag{6}$$

Proof. From the Dacunha-Castelle representation ([8]) we have

$$p_{T,c}^{\mathrm{dl}}(\boldsymbol{y} \mid \boldsymbol{x}^{(c)}) = \frac{f_c(\boldsymbol{y})}{f_c(\boldsymbol{x}^{(c)})} \frac{\sqrt{C}}{\sqrt{2\pi T}} \exp\left\{\frac{-\|\boldsymbol{y} - \boldsymbol{x}_c\|^2}{2T}\right\} \mathbb{E}_{\overline{\mathbb{W}}}\left(\exp\left\{-\int_0^T \phi_c^{\mathrm{dl}}(\boldsymbol{x}_t^{(c)})\, \mathrm{d}t\right\}\right).$$

The result then follows from (2) and (3) by rearrangement and recalling that $\sum_{c=1}^{C} \|\boldsymbol{y} - \boldsymbol{x}^{(c)}\|^2 = \sum_{c=1}^{C} \|\boldsymbol{x}^{(c)} - \bar{\boldsymbol{x}}\|^2 + C\|\boldsymbol{y} - \bar{\boldsymbol{x}}\|^2$. □

Here ρ^{bm} and Q^{bm} are both necessarily bounded by 1, and when interpreted methodologically (see the next section) correspond to separate acceptance steps within our rejection sampling framework. An event of probability ρ^{bm} can be simulated by direct computation, and an event of probability Q^{bm} can be simulated using the extensive efficient methodology on Poisson samplers (using an auxiliary diffusion bridge path-space rejection sampler as developed in, e.g. [3], [4], [5], [6], [7], [9], [20], [21], and [22]). Note that there is a tradeoff involved in the (user-specified) choice of T. For small T, ρ^{bm} will likely be small while Q^{bm} is large, whereas, for large T, the opposite will be true. A small value of T is usually preferred since the computational cost for the diffusion bridge rejection sampling for Q^{bm} is comparatively expensive.

The algorithm for simulating from f (by means of g) therefore proceeds as per Algorithm 1 (which we term *Monte Carlo fusion*).

Algorithm 1. (*Monte Carlo fusion (Brownian bridge approach).*)

```
1.  Initialize a value $T > 0$;
2.  For $c = 1, \dots, C$, simulate $\boldsymbol{x}_c$ from the density $f_c(\boldsymbol{x})$ and calculate $\bar{\boldsymbol{x}}$;
3.  Simulate $\boldsymbol{y}$ from the Gaussian distribution, with density $\exp(-C\|\boldsymbol{y} - \bar{\boldsymbol{x}}\|^2/2T)$;
4.  Generate the standard uniform random variable $U_1$;
5.  if $\log U_1 \le -C\sigma^2/2T$ then
6.      Generate the standard uniform variable $U_2$ and the independent Brownian bridges
        $\boldsymbol{x}_t^{(c)}$, $c = 1, \dots, C$, in $[0, T]$, conditional on the starting point $\boldsymbol{x}_c$ and ending point $\boldsymbol{y}$;
7.      if
            $$U_2 \le E^{\text{bm}} \qquad (7)$$
        then
8.          Accept and output $\boldsymbol{y}$ as a sample from $f(\boldsymbol{x})$;
        // Event (7) can be dealt with via the path-space
        rejection sampling methods in Beskos and Roberts (2005),
        Beskos et al. (2006a), Beskos et al. (2008), and Pollock
        et al. (2016a)
9.      else
10.         Go back to step 2;
11.     end
12. else
13.     Go back to step 2;
14. end
```

3.2. Practical interpretation of the algorithm

Remark 1. (*Adjustment for the simple average of the sample from subdensities.*) In the above algorithm, for all c, $\boldsymbol{x}^{(c)}$ is simulated from $f_c(\boldsymbol{x})$ as in other Monte Carlo fusion algorithms. The proposed combined value $\boldsymbol{y}$, however, is actually generated from a Gaussian distribution with mean $\bar{\boldsymbol{x}}$ and covariance matrix $C^{-1}T\boldsymbol{I}_d$. Therefore, $\boldsymbol{y}$ can be viewed as a simple average of the values $\{\boldsymbol{x}^{(c)},\ c \in C\}$ added to a Gaussian random error term. This algorithm indicates exactly how the simple average of these independent sub-posterior samples can be adjusted as a draw from the target distribution. This adjustment is in the form of the accept/reject step with acceptance probability $\rho^{\text{bm}} \times Q^{\text{bm}}$.

Remark 2. (*Implication on Bayesian group decision theory.*) In step 5 of Algorithm 1, we can also write $\log U_1 \le -C\sigma^2/2T$ as

$$\sigma^2 \le -\frac{2T \log U_1}{C}. \qquad (8)$$

Note that σ^2 is actually the *sample variance* of the simulated starting points, $\boldsymbol{x}^{(c)}$ (strictly speaking, it is the sum of variances for each component of $\boldsymbol{x}^{(c)}$). Thus, in this step, the acceptance condition (8) implies that $\boldsymbol{y}$ will have a reasonable probability of being accepted as a sample from $f(\boldsymbol{x})$ only when the variance of the $\boldsymbol{x}^{(c)}$ is small enough. This coincides with our intuition in group decision making. For example, the small variance of $\{\boldsymbol{x}^{(c)},\ c \in C\}$ (satisfying condition (8)) means that the decisions/results from each group are similar, so that we can combine these decisions/results in step 7, using (7), of Algorithm 1. On the other hand, if the variance of $\{\boldsymbol{x}^{(c)},\ c \in C\}$ (not satisfying condition (8)) is large, then $\{\boldsymbol{x}^{(c)},\ c \in C\}$ provide contradictory evidence and should usually be rejected. Note that, algorithmically, this initial rejection sampling step is efficient since early rejection then avoids the need to carry out the more complicated (and computationally expensive) step 7, via the condition of (7).

Remark 3. (*An extreme case.*) A very interesting extreme case is to choose $T=0$. Then the event (7) will certainly happen, but the event $U_1 \le \exp\{-C\sigma^2/2T\}$ will occur only if $\boldsymbol{x}^{(1)} = \cdots = \boldsymbol{x}^{(C)} = \boldsymbol{y}$. Therefore, in such an extreme case, Algorithm 1 (theoretically) draws a sample $\boldsymbol{y}$ from

$$h^{\mathrm{bm}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y}) \propto \prod_{c=1}^{C} f_c(\boldsymbol{y}),$$

which is the target distribution. In practice, however, we have to choose $T>0$, since the independent sub-posterior samples $\boldsymbol{x}^{(c)}$ have zero probability to be the same.

4. A fusion rejection sampler using Ornstein–Uhlenbeck bridges

4.1. The methodology

In the previous section, the proposal density h^{bm} uses a simple average of the sub-posterior samples $\boldsymbol{x}^{(c)}$ as the mean of the proposal $\boldsymbol{y}$. Another approach is to consider using a weighted average of the sub-posterior samples $\boldsymbol{x}^{(c)}$ as the mean of the proposal $\boldsymbol{y}$. For example, in a typical meta-analysis, a weighted average from different research outputs is typically used as the unification mean, and an individual output with more certainty (or, smaller variance) should consequently have larger weights ([12]).

Denote by $\hat{\boldsymbol{\mu}}_c$ and $\hat{\boldsymbol{\Lambda}}_c$ the mean and the inverse of the covariance matrix estimates for distribution $f_c(\boldsymbol{x})$, respectively. We consider the more general proposal density

$$h^{\mathrm{ou}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y}) \propto \biggl[\prod_{c=1}^{C} f_c(\boldsymbol{x}^{(c)})\biggr] \operatorname{etr}\biggl[-\frac{1}{2}[\boldsymbol{y}-\tilde{\boldsymbol{x}}]^{\otimes 2}\boldsymbol{D}\biggr],$$

where the notation etr is the exponential trace function, '$\otimes$' is the Kronecker product,

$$\boldsymbol{D} = \sum_{c=1}^{C} \boldsymbol{D}_c, \qquad \boldsymbol{D}_c = \boldsymbol{V}_c^{-1} - \hat{\boldsymbol{\Lambda}}_c,$$

$$\boldsymbol{V}_c := \boldsymbol{V}_c(T) = \operatorname{var}\biggl(\int_0^T \mathrm{e}^{\hat{\boldsymbol{\Lambda}}_c(t-T)}\,\mathrm{d}\boldsymbol{W}_t^{(c)}\biggr) = \frac{\hat{\boldsymbol{\Lambda}}_c^{-1}}{2}(\boldsymbol{I}_d - \mathrm{e}^{-2\hat{\boldsymbol{\Lambda}}_c T}),$$

and

$$\tilde{\boldsymbol{x}} = \boldsymbol{D}^{-1}\biggl\{\sum_{c=1}^{C}(\boldsymbol{V}_c^{-1}\boldsymbol{m}_c - \hat{\boldsymbol{\Lambda}}_c\hat{\boldsymbol{\mu}}_c)\biggr\},$$

$$\boldsymbol{m}_c := \boldsymbol{m}_c(\boldsymbol{x}^{(c)}, T) = \hat{\boldsymbol{\mu}}_c + \mathrm{e}^{-\hat{\boldsymbol{\Lambda}}_c T}(\boldsymbol{x}^{(c)} - \hat{\boldsymbol{\mu}}_c). \tag{9}$$

It is also straightforward to simulate from $h^{\mathrm{ou}}(\cdot)$, since the $\boldsymbol{x}^{(c)}$ are independently drawn from $f_c(\boldsymbol{x})$, and then $\boldsymbol{y}$ is generated from a Gaussian distribution with mean $\tilde{\boldsymbol{x}}$ and covariance matrix $\boldsymbol{D}^{-1}$.

Here the vector $\boldsymbol{m}_c$ and the matrix $\boldsymbol{V}_c$ are actually the mean and covariance matrices, respectively, for the following Ornstein–Uhlenbeck (OU) process at time T conditional on the starting point $\boldsymbol{x}_0^{(c)} = \boldsymbol{x}^{(c)}$ ([17]):

$$\mathrm{d}\boldsymbol{X}_t^{(c)} = \nabla A_c^{\mathrm{ou}}(\boldsymbol{X}_t^{(c)})\,\mathrm{d}t + \mathrm{d}\boldsymbol{W}_t^{(c)}, \qquad \boldsymbol{X}_0^{(c)} \sim f_c^2(\boldsymbol{x}), \tag{10}$$

with

$$A_c^{\mathrm{ou}}(\boldsymbol{x}) = -\frac{(\hat{\boldsymbol{\mu}}_c - \boldsymbol{x})^{tr}\hat{\boldsymbol{\Lambda}}_c(\hat{\boldsymbol{\mu}}_c - \boldsymbol{x})}{2}.$$

We shall also require the following regularity property.

Condition 2. *Define*

$$\phi_c^{\mathrm{ou}}(\boldsymbol{x}) = \tfrac{1}{2}(\|\hat{\boldsymbol{\Lambda}}_c(\hat{\boldsymbol{\mu}}_c - \boldsymbol{x})\|^2 - \mathrm{trace}(\hat{\boldsymbol{\Lambda}}_c)). \tag{11}$$

For any $c \in \{1, \dots, C\}$*, there exists* $\Phi_c^{\mathrm{ou}} > -\infty$ *such that, for all* $\boldsymbol{x}$*,*

$$\phi_c^{\mathrm{dl}}(\boldsymbol{x}) - \phi_c^{\mathrm{ou}}(\boldsymbol{x}) \geq \Phi_c^{\mathrm{ou}}.$$

We now have the following result.

Proposition 3. *Define function*

$$\begin{aligned}\rho^{\mathrm{ou}} &:= \rho^{\mathrm{ou}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}) \\ &= \mathrm{etr}\bigg\{-\frac{1}{2}\bigg[\boldsymbol{H}\boldsymbol{D}^{-1} + \sum_{c=1}^{C}\boldsymbol{M}_{1,c}(\boldsymbol{m}_c + \boldsymbol{M}_{1,c}^{-1}\boldsymbol{M}_{2,c}\boldsymbol{V}_c\hat{\boldsymbol{\Lambda}}_c\hat{\boldsymbol{\mu}}_c)^{\otimes 2}\bigg]\bigg\},\end{aligned} \tag{12}$$

with

$$\boldsymbol{M}_{1,c} = \mathrm{e}^{2\hat{\boldsymbol{\Lambda}}_c T}\hat{\boldsymbol{\Lambda}}_c - \boldsymbol{V}_c^{-1}\bigg(\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\bigg)\boldsymbol{D}^{-1},$$

$$\boldsymbol{M}_{2,c} = \boldsymbol{V}_c^{-1}\bigg(\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\bigg)\boldsymbol{D}^{-1} - 2\hat{\boldsymbol{\Lambda}}_c\mathrm{e}^{2\hat{\boldsymbol{\Lambda}}_c T},$$

$$\boldsymbol{H} = \bigg(\sum_{c=1}^{C}(\boldsymbol{m}_c - \boldsymbol{V}_c\hat{\boldsymbol{\Lambda}}_c\hat{\boldsymbol{\mu}}_c)^{\otimes 2}\boldsymbol{V}_c^{-1}\bigg)\bigg(\sum_{c=1}^{C}\boldsymbol{V}_c^{-1}\bigg) - \bigg\{\sum_{c=1}^{C}\boldsymbol{V}_c^{-1}(\boldsymbol{m}_c - \boldsymbol{V}_c\hat{\boldsymbol{\Lambda}}_c\hat{\boldsymbol{\mu}}_c)\bigg\}^{\otimes 2}.$$

Under Condition 2, we can write

$$\frac{g^{\mathrm{dl}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y})}{h^{\mathrm{ou}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y})} \propto \rho^{\mathrm{ou}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}) \times Q^{\mathrm{ou}} \times \prod_{c=1}^{C}\mathrm{e}^{-T\Phi_c^{\mathrm{ou}}}, \tag{13}$$

where

$$Q^{\mathrm{ou}} = \mathbb{E}_{\overline{\mathbb{O}}}(E^{\mathrm{ou}}),$$

with $\overline{\mathbb{O}}$ denoting the law of C OU bridges $\boldsymbol{x}_t^{(c)}$, $c = 1, \cdots, C$, in time interval $[0, T]$. *These $\boldsymbol{x}_t^{(c)}$ are independent conditional on the starting point $\boldsymbol{x}^{(c)}$ and common ending point $\boldsymbol{y}$, and*

$$E^{\text{ou}} := \prod_{c=1}^{C} \left[\exp\left\{ -\int_0^T (\phi_c^{\text{dl}}(\boldsymbol{x}_t^{(c)}) - \phi_c^{\text{ou}}(\boldsymbol{x}_t^{(c)}) - \Phi_c^{\text{ou}})\, \mathrm{d}t \right\}\right]. \tag{14}$$

Proof. See Appendix A. □

Since ρ^{ou} and Q^{ou} in (13) are always no more than 1, we have the following rejection sampling algorithm.

Algorithm 2. (*Monte Carlo fusion (Ornstein–Uhlenbeck approach).*)

```
1.  Initialise a value $T > 0$ and $\hat{\boldsymbol{\mu}}_c$, $\hat{\boldsymbol{\Lambda}}_c$;
2.  For $c = 1, \ldots, C$, simulate $\boldsymbol{x}^{(c)}$ from the density $f_c(\boldsymbol{x})$ and calculate $\tilde{\boldsymbol{x}}$, $\boldsymbol{D}$;
3.  Simulate $\boldsymbol{y}$ from the Gaussian distribution, with mean $\tilde{\boldsymbol{x}}$ and covariance matrix $\boldsymbol{D}^{-1}$;
4.  Generate the standard uniform random variable $U_1$;
5.  if $U_1 \le \rho^{\text{ou}}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)})$ (given in (12)) then
6.      Generate the standard uniform random variable $U_2$ and independent OU bridges
        $\boldsymbol{x}_t^{(c)}$, $c = 1, \ldots, C$, in $[0, T]$, conditional on the starting point $\boldsymbol{x}_0^{(c)} = \boldsymbol{x}^{(c)}$ and ending point
        $\boldsymbol{y}$;
7.      if
            $$U_2 \le \exp\left\{ -\sum_{c=1}^{C} \int_0^T (\phi_c^{\text{dl}}(\boldsymbol{x}_t^{(c)}) - \phi_c^{\text{ou}}(\boldsymbol{x}_t^{(c)}) - \Phi_c^{\text{ou}})\, \mathrm{d}t \right\} \tag{15}$$
        // Event (15) can be dealt with via the path-space
        rejection sampling methods in Beskos and Roberts (2005),
        Beskos et al. (2006a), Beskos et al. (2008), and Pollock
        et al. (2016a)
8.      then
9.          Accept and output $\boldsymbol{y}$ as a sample from $f(\boldsymbol{x})$;
10.     else
11.         Go back to step 2;
12.     end
13. else
14.     Go back to step 2;
15. end
```

In Algorithm 2, T is a tuning parameter as well. If we choose a small value of T, the acceptance probability $\rho^{\text{ou}}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)})$ will be very low. This is noticed by the facts that $\text{trace}(\boldsymbol{H}\boldsymbol{D}^{-1}) = \infty$ and $\boldsymbol{M}_{1,c}$ and $\boldsymbol{M}_{2,c}$ are finite matrices, if $T = 0$. Therefore, in practice, we should choose a reasonably large value T. However, if we choose a large T, the acceptance probability (15) will be very small. In practice, it is usually preferable to choose a small T since the computational cost for the acceptance/rejection step of (15) is much higher.

4.2. Connection with consensus Monte Carlo

In practice, people may employ an approximate version of Algorithm 2, since we can ignore condition (15) in the rejection step 7, if we choose a small value T. In other words, from (13) of Proposition 3, for small T, the target $g^{\mathrm{dl}}(\cdot)$ is approximately equal to a function proportional to

$$\tilde{h}^{\mathrm{ou}} = h^{\mathrm{ou}} \cdot \rho^{\mathrm{ou}} = \Bigg[\prod_{c=1}^{C} f_c(\boldsymbol{x}^{(c)})\Bigg] \operatorname{etr}\Big[-\frac{1}{2}[\boldsymbol{y}-\tilde{\boldsymbol{x}}]^{\otimes 2}\boldsymbol{D}\Big] \cdot \rho^{\mathrm{ou}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}).$$

Such an approximation will be very good since Q^{ou}, the acceptance probability (15), will be very close to 1 with a small T. In other words, we only simulate $\boldsymbol{y}$ from $\tilde{h}^{\mathrm{ou}}(\cdot)$ and accept $\boldsymbol{y}$ as a sample approximately from the target distribution $f(\cdot)$.

We draw $\boldsymbol{x}_c$, $c = 1, \dots, C$, independently from each $f_c(\boldsymbol{x})$, respectively. Another naive approach to combine these draws is to use the following linear combination:

$$\boldsymbol{y} = \Bigg(\sum_{c=1}^{C} \hat{\boldsymbol{\Lambda}}_c\Bigg)^{-1} \Bigg[\sum_{c=1}^{C} \hat{\boldsymbol{\Lambda}}_c \boldsymbol{x}_c\Bigg]. \tag{16}$$

In practice, $\hat{\boldsymbol{\Lambda}}_c^{-1}$ can be obtained via preliminary analysis. Such a $\boldsymbol{y}$ can be viewed as a sample approximately from $f(\boldsymbol{x})$ as well. This is named as consensus Monte Carlo in [23].

The following lemma tells us how the simulation of $\boldsymbol{y}$ from $\tilde{h}^{\mathrm{ou}}(\cdot)$ is related to the consensus Monte Carlo sample (16).

Lemma 1. *If $f_c(\boldsymbol{x})$ is a Gaussian distribution and if we choose $T = \infty$, then simulating $\boldsymbol{y}$ from $\tilde{h}^{\mathrm{ou}}(\cdot)$ will be the same as the consensus Monte Carlo (CMC) method. Both draw samples exactly from the target distribution.*

Proof. See Appendix B. □

This lemma tells us why CMC does not provide good results. It is because CMC simulates $\boldsymbol{y}$ from $\tilde{h}^{\mathrm{ou}}(\cdot)$ with $T = \infty$; however, T should be chosen as a small value to achieve better approximation or even use the exact Algorithm 2 with the diffusion path rejection sampler.

5. Simulation studies

5.1. Distribution with light tails

We consider the target distribution $f(x) \propto \mathrm{e}^{-x^4/2}$. We choose $C = 4$ and $f_c(x) = \mathrm{e}^{-x^4/2C}$, $c = 1, \cdots, C$. It is easy to check that $\phi^{\mathrm{dl}}(x) = \frac{1}{2}(4x^6/C^2 - 6x^2/C)$ satisfies Condition 1. On the other hand, for any chosen values $\hat{\boldsymbol{\mu}}_c$ and $\hat{\boldsymbol{\Lambda}}_c$, $\phi_c^{\mathrm{ou}}(\boldsymbol{x})$ in (11) satisfies Condition 2 as well. Therefore, both Algorithm 1 and Algorithm 2 can be applied to simulate from $f(x)$.

We compare the following Monte Carlo (MC) methods for the estimation of the density function of $f(x)$.

1. Simulating MC samples directly from $f(x)$ via a simple rejection sampling with standard Gaussian distribution as the proposal.

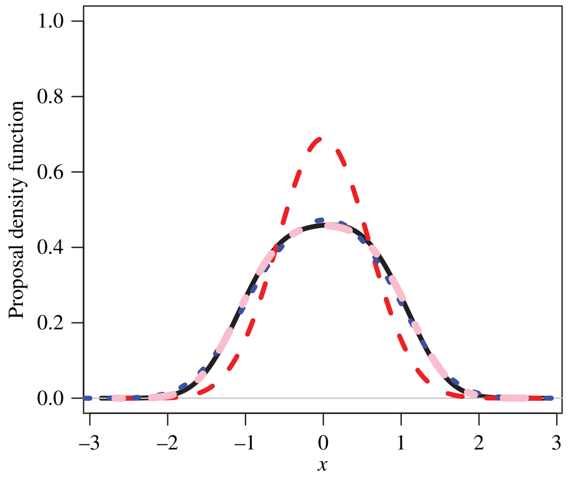

FIGURE 1: Kernel density fitting with bandwidth 0.25 for the density proportional to $e^{-x^4/2}$, based on different MC methods: the standard exact MC (simulation 1, *solid curve*); Algorithm 1 (simulation 2, *dash–dot curve*); Algorithm 2 (simulation 3, *dotted curve*); the CMC algorithm (simulation 4, *dashed curve*).

TABLE 1: System running times in seconds for simulating 10 000 realisations.

	Algorithm		
	CMC	Algorithm 1	Algorithm 2
Running time (s)	0.05	0.36	4.05

2. Simulating MC samples based on the exact simulation method, Algorithm 1 with $T = 1$.

3. Simulating MC samples based on the exact simulation method, Algorithm 2 with $T = 1$.

4. Simulating MC samples based on the consensus method of [23].

The density curve estimation results are summarised in Figure 1. Note that all results are based on 10 000 realisations. The solid curve (simulation 1, the true fitted density curve), the dotted curve (simulation 2), and the dash–dot curve (simulation 3) are all exact algorithms and they are almost identical. The consensus method (simulation 4, the dashed curve) has very large biases. Note that both the CMC algorithm and Algorithm 2 use the same values of $\hat{\boldsymbol{\mu}}_c$ and $\hat{\boldsymbol{\Lambda}}_c$ based on preliminary analysis.

The running time of the algorithms are presented in Table 1. It seems that Algorithm 2 uses the most system running time. This is because Algorithm 2 has a smaller acceptance probability (about 0.011 for the path-space rejection sampling (15)) than that of Algorithm 1 (about 0.139 for the path-space rejection sampling (7)).

Note that Condition 1 is usually satisfied in most applications; however, Condition 2 only holds when the target $f(x)$ has lighter tails than Gaussian distributions. We present an example in the following section, where only Condition 1 holds.

5.2. Beta distribution

Consider the target distribution as the beta distribution with density $\pi(u) \propto u^4(1-u),\ u \in [0, 1]$, i.e. Beta(5, 2). To use the proposed algorithms, the support of the target distribution should be in the whole real axis. Therefore, we need to use the variable transformation $x = \log(u/(1-u))$ and consider the target distribution as

$$f(x) \propto \left[\frac{\exp(x)}{1+\exp(x)}\right]^5 \left[\frac{1}{1+\exp(x)}\right]^2.$$

We decompose $\pi(x)$ into $C = 5$ components,

$$\begin{aligned} f(x) &\propto f_1(x) \cdots f_C(x) \\ f_c(x) &= \left[\frac{\exp(x)}{1+\exp(x)}\right]\left[\frac{1}{1+\exp(x)}\right]^{0.4}. \end{aligned} \tag{17}$$

Note that, for this simple example, Condition 1 is satisfied but Condition 2 is not satisfied. Therefore, we compare the following MC methods for the estimation of the density function of Beta(5, 2).

(a) Simulating MC samples directly from Beta(5, 2) via the simple R command, *rbeta*.

(b) Simulating MC samples based on the exact simulation method, Algorithm 1 with $T = 3$.

(c) Simulating MC samples based on the consensus method of [23], with variable transformation and with the decomposition in (17).

For simulation (c), $\hat{\boldsymbol{\mu}}_c$ and $\hat{\boldsymbol{\Lambda}}_c$ (also known as the weight of each consensus sample in the CMC algorithm) are respectively chosen as the estimated mean and inverse of the variance of $f_c(\cdot)$, as suggested by [23].

The density curve estimation results are summarized in Figure 2. Note that all results are based on 10 000 realisations. The solid curve (simulation (a)) and the dotted curve (simulation (b)) are almost identical, since both of them are based on exact simulation methods. Again, the CMC algorithm gives very biased results.

Remark 4. (*Tuning parameters and the density decomposition.*) We may choose any value T in the proposed algorithms. However, as we mentioned before, value T is a tuning parameter for the efficiency of Algorithm 1, which is indeed shown by our simulation results. The CPU running time for simulating 10 000 realisations based on Algorithm 1 for the beta example is summarized in Figure 3. The optimum value is about $T = 2$.

We here use Algorithm 1 to provide two empirical approaches to find a good value T in practice.

Approach 1. Although it is not easy to calculate an explicit value for the overall acceptance probability

$$\underbrace{\mathrm{e}^{-C\sigma^2/2T}}_{\rho^{\mathrm{bm}}} \underbrace{\mathbb{E}_{\overline{\mathbb{W}}}\left(\prod_{c=1}^{C}\left[\exp\left\{-\int_0^T (\phi_c^{\mathrm{dl}}(\boldsymbol{x}_t^{(c)}) - \Phi_c^{\mathrm{bm}})\,\mathrm{d}t\right\}\right]\right)}_{Q^{\mathrm{bm}}}, \tag{18}$$

where σ^2 is the sample variance of the distributed MC samples, it can be estimated easily using a small number of preliminary simulation studies. Therefore, in practice, we choose a sequence

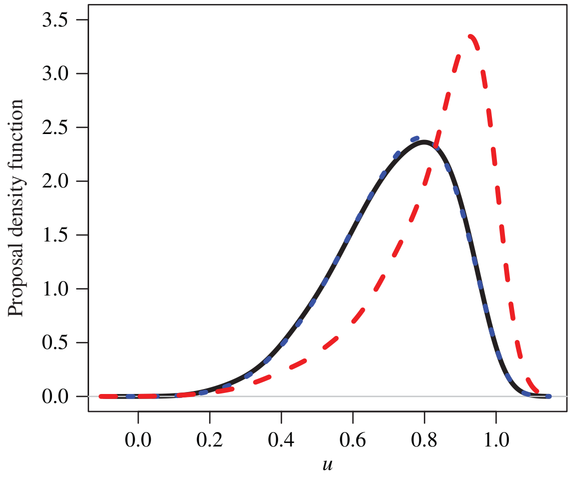

FIGURE 2: Kernel density fitting with bandwidth 0.04 for Beta(5, 2), based on different MC methods: the standard exact MC (simulation (a), *solid curve*); Algorithm 1 (simulation (b), *dotted curve*); the CMC algorithm (simulation (c), *dashed curve*).

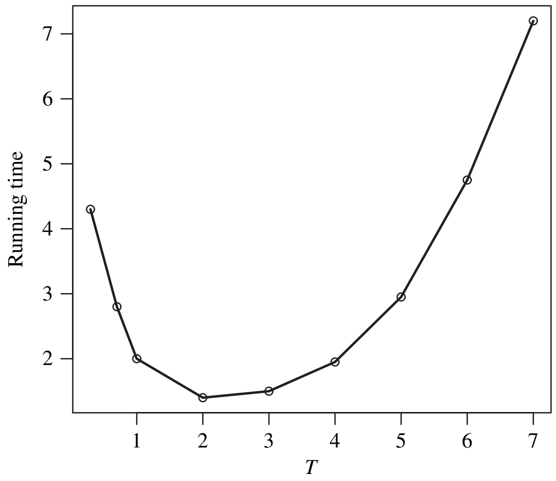

FIGURE 3: CPU time (in seconds) for Algorithm 1, based on different T.

of different values of T, say $T_1, T_2, \ldots, T_q$, and, for each T_i, we carry out a small number of preliminary simulations to estimate the above probability. Then the value T_i which gives the largest acceptance probability is chosen.

Approach 2. When dealing with Q^{bm} via path-space rejection sampling, we need upper bounds Ψ_c, $\phi_c^{\mathrm{dl}}(\boldsymbol{x}_c) \le \Psi_c$ for all $\boldsymbol{x}_c$, and we want $(\Psi_c - \Phi_c^{\mathrm{bm}})T$ to be as small as possible to give efficient Poisson thinning steps in path-space rejection sampling for diffusion bridges. Therefore, if Ψ_c (or an approximate value) is available, it makes sense to consider choosing a value of T to give a large value for the following probability (replacing $\phi_c^{\mathrm{dl}}(\boldsymbol{x}_t^{(c)})$ by Ψ_c in (18)):

$$\mathrm{e}^{-C\sigma^2/2T}\prod_{c=1}^{C}\bigg[\exp\bigg\{-\int_0^T(\Psi_c-\Phi_c^{\mathrm{bm}})\,\mathrm{d}t\bigg\}\bigg] = \exp\bigg\{-\frac{\sum_{c=1}^{C}\|\boldsymbol{x}^{(c)}-\bar{\boldsymbol{x}}\|^2}{2T}-\sum_{c=1}^{C}(\Psi_c-\Phi_c^{\mathrm{bm}})T\bigg\}.$$

Each preliminary simulation study simulates values $\boldsymbol{x}^{(c)}$ from each $f_c(\boldsymbol{x})$ and thus gives $\bar{\boldsymbol{x}}$. Therefore, we can find the value T^* which can maximize the above formula in each preliminary simulation study. The averaged optimal T^* values corresponding to each preliminary study may also be used as the value T.

In addition, when we split the target f into $f \propto f_1\cdots f_C$, we actually chose $f_1=\cdots=f_C=f^{1/C}$, since such a decomposition will always give the smallest variation for $\boldsymbol{x}^{(c)}$, $c=1,\dots,C$, as suggested in [10].

5.3. Normal distribution: comparison of the two algorithms

We have seen in previous sections that Algorithm 2 may have limited applications due to its complexity and the stronger condition requirement, Condition 2. However, Algorithm 2 can gain great advantage if the target distribution is very close to a Gaussian distribution, which is usually true for Bayesian posterior densities under the big data framework. We here use a simple Gaussian example to compare the two algorithms.

We consider the target distribution as standard normal $f(x)\propto \mathrm{e}^{-x^2/2}$ and $\phi_c^{\mathrm{dl}}(x)=\frac{1}{2}(x^2/C^2-1/C)$. We choose the OU process with $\hat{\boldsymbol{\mu}}_c=0$ and $\hat{\boldsymbol{\Lambda}}_c=1/2C$, which gives $\phi_c^{\mathrm{ou}}(\boldsymbol{x})=\frac{1}{2}(x^2/4C^2-1/2C)$. The normalising constant $\Phi_c^{\mathrm{ou}}=-1/4C$ guarantees that $\phi_c^{\mathrm{dl}}(\boldsymbol{x})-\phi_c^{\mathrm{ou}}(\boldsymbol{x})$ is bounded below. Under these settings, the acceptance probability E^{ou} in (14) becomes

$$E^{\mathrm{ou}} := \prod_{c=1}^{C}\bigg[\exp\bigg\{-\int_0^T\frac{3\boldsymbol{x}_t^{(c)}}{8C^2}\,\mathrm{d}t\bigg\}\bigg]$$

and, furthermore, $Q^{\mathrm{ou}}=\mathbb{E}_{\overline{\mathbb{O}}}(E^{\mathrm{ou}})$. On the other hand, if we use Algorithm 1, we can choose $\Phi_c^{\mathrm{bm}}=-1/2C$ and the acceptance probability E^{bm} defined in (6) becomes

$$E^{\mathrm{bm}} := \prod_{c=1}^{C}\bigg[\exp\bigg\{-\int_0^T\frac{\boldsymbol{x}_t^{(c)}}{2C^2}\,\mathrm{d}t\bigg\}\bigg]$$

and $Q^{\mathrm{bm}}=\mathbb{E}_{\overline{\mathbb{W}}}(E^{\mathrm{bm}})$. Note that $\overline{\mathbb{W}}$ and $\overline{\mathbb{O}}$ are the probability measures induced by Brownian bridges and OU bridges, respectively, conditional on the same starting and ending points. It is clear that, for the same $\boldsymbol{x}_t^{(c)}$, $E^{\mathrm{ou}}\geq E^{\mathrm{bm}}$; thus, Q^{ou} should be greater than Q^{bm}, if the value T is not too large. Therefore, Algorithm 2 should have higher acceptance probabilities for the complicated path-space rejection sampling for diffusions. In addition, we also found that ρ^{ou} is higher than ρ^{bm} for this toy Gaussian example via 10 000 simulations.

We chose $T=3$ for both algorithms, which is about the most efficient tuning parameter value for both. For Algorithm 2, the overall acceptance probability is about 0.1 with $\rho^{\mathrm{ou}}\approx 0.16$ and $Q^{\mathrm{ou}}\approx 0.63$. The overall acceptance probability for Algorithm 1 is about 0.07 with $\rho^{\mathrm{bm}}\approx 0.14$ and $Q^{\mathrm{bm}}\approx 0.51$. Indeed, Algorithm 2 gains advantages when the target is an (approximate) Gaussian distribution. In fact, when using Algorithm 2, if we choose an OU process such that $\phi_c^{\mathrm{dl}}(\boldsymbol{x}) \approx \phi_c^{\mathrm{ou}}(\boldsymbol{x})$, the acceptance probability Q^{ou} will be almost close to 1, which is much better than Algorithm 1.

6. Conclusion

In this paper we have introduced a novel theoretical framework, and direct methodological implementation (Monte Carlo fusion), to address the common (but challenging) 'fusion' problem—unifying distributed analyses and inferences on shared parameters from multiple sources into a single coherent inference—by viewing it as a simple rejection sampler on an extended space (Section 2), and developing an appropriate sampling mechanism (Section 3).

Our fusion approach in this paper is not only the first to answer in a principled manner how to combine samples from multiple sources, but (as shown in Section 4) also provides a principled approach to understand the errors that arise in other existing unification schemes (such as those in big data divide-and-conquer approaches). The errors in existing unification schemes can be considerable, even for simple one-dimensional unification targets (such as those we consider in Section 5).

Characterising the error in existing unification schemes is possible by setting the (proposal) sampling mechanisms of those schemes within our framework, and finding a representation for the remaining error (which we could remove by further acceptance or rejection). This opens interesting avenues of research in which existing unification schemes are adapted within our framework into efficient proposal mechanisms for our extended fusion target density (2).

A number of avenues to apply our work directly to interesting applications are possible. In addition to those discussed in Section 1 (namely expert elicitation, multiview learning, and meta-analysis), other application areas include Bayesian group decision theory (see Remark 2) and Bayesian sensitivity analysis. In the case of Bayesian sensitivity analysis we need to assess a large number of prior distributions, but existing methods address this by using approximations ([27]).

A key avenue for (on-going) future research is to fully explore how to use the Monte Carlo fusion framework we introduce to modify, and remove approximation, from existing Monte Carlo methods (particularly within the big data setting) that use the divide-and-conquer (fork-and-join) strategies described in Section 1. In the particular setting of big data the unification of distributed inferences is only part of the problem—additional constraints are imposed due to practical computational and hardware concerns (such as avoiding as far as possible communication between computing cores). As such, key future research will focus on how to implement Monte Carlo fusion with this particular set of constraints (as direct implementation is not possible).

Another interesting big data direction would be to blend the multi-core approach of divide-and-conquer strategies with state-of-the-art single-core approaches for big data (see, for instance, [22]).

Appendix A. Proof of Proposition 3

To prove the proposition, we first introduce a lemma about a density proportional to the following function

$$\tilde{g}^{\mathrm{ou}}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y}) = \prod_{c=1}^{C} \big[f_c(\boldsymbol{x}^{(c)}) p_{T,c}^{\mathrm{ou}}(\boldsymbol{y} \mid \boldsymbol{x}^{(c)}) \exp\big(A_c^{\mathrm{ou}}(\boldsymbol{x}^{(c)}) - A_c^{\mathrm{ou}}(\boldsymbol{y})\big)\big], \tag{19}$$

where $p^{\mathrm{ou}}_{T,c}(\boldsymbol{y} \mid \boldsymbol{x}^{(c)})$ is the transition density from $\boldsymbol{x}^{(c)}$ at time 0 to $\boldsymbol{y}$ at time T for the OU process given by (10).

Lemma 2. (The expression of $\tilde{g}^{\mathrm{ou}}$.) *The formula in (19) can be rewritten as*

$$\tilde{g}^{\mathrm{ou}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y}) \propto \bigg[\prod_{c=1}^{C} f_c(\boldsymbol{x}^{(c)})\bigg] \operatorname{etr}\bigg[-\frac{1}{2}[\boldsymbol{y}_T - \tilde{\boldsymbol{x}}]^{\otimes 2}\boldsymbol{D}\bigg]\rho^{\mathrm{ou}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}),$$

where $\tilde{\boldsymbol{x}}$ and ρ^{ou} are given by (9) and (12), respectively.

Proof. See the supplementary file ([11]). □

Proof of Proposition 3. If we denote $\overline{\mathbb{DL}}$ as the law of C d-dimensional Langevin diffusion bridges given in (4), with starting points $\boldsymbol{x}^{(c)}$ and common ending point $\boldsymbol{y}$, the result of Proposition 2 can be written as

$$\frac{g^{\mathrm{dl}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y})}{h^{\mathrm{bm}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y})} \times \mathrm{d}\overline{\mathbb{DL}}(\vec{\boldsymbol{x}}) \propto \rho^{\mathrm{bm}} \times E^{\mathrm{bm}} \times \prod_{c=1}^{C} \mathrm{e}^{-T\Phi_c} \times \mathrm{d}\overline{\mathbb{W}}(\vec{\boldsymbol{x}}),$$

where $\vec{\boldsymbol{x}} = \{\boldsymbol{x}_t^{(c)},\ c = 1, \cdots, C,\ t \in [0, T]\}$ are typical diffusion bridge paths, with starting points $\boldsymbol{x}^{(c)}$ and common ending point $\boldsymbol{y}$.

If we consider a very special case where each $f_c(\boldsymbol{x})$ is a Gaussian density $\exp(A_c^{\mathrm{ou}}(\boldsymbol{x}))$, we immediately find that the target g^{dl} becomes

$$g^{\mathrm{ou}} = \prod_{c=1}^{C} \bigg[\mathrm{e}^{2A_c^{\mathrm{ou}}(\boldsymbol{x}^{(c)})} p^{\mathrm{ou}}_{T,c}(\boldsymbol{y} \mid \boldsymbol{x}^{(c)}) \frac{1}{\mathrm{e}^{A_c^{\mathrm{ou}}(\boldsymbol{y})}}\bigg].$$

In addition, the proposal density h^{bm} in Proposition 2 becomes

$$\hbar^{\mathrm{bm}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y}) = \prod_{c=1}^{C} [\mathrm{e}^{A_c^{\mathrm{ou}}(\boldsymbol{x}^{(c)})}]\mathrm{e}^{-\|\boldsymbol{y}-\bar{\boldsymbol{x}}\|^2/2T},$$

and, furthermore, the rejection sampling ratio in Proposition 2 becomes

$$\frac{g^{\mathrm{ou}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y})}{\hbar^{\mathrm{bm}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y})} \times \mathrm{d}\overline{\mathbb{O}}(\vec{\boldsymbol{x}}) \propto \rho^{\mathrm{bm}} \times E_*^{\mathrm{ou}} \times \mathrm{d}\overline{\mathbb{W}}(\vec{\boldsymbol{x}}),$$

$$E_*^{\mathrm{ou}} := \prod_{c=1}^{C} \bigg[\exp\bigg\{-\int_0^T \phi_c^{\mathrm{ou}}(\boldsymbol{x}_t^{(c)})\ \mathrm{d}t\bigg\}\bigg].$$

Now we have

$$\begin{aligned}
\frac{g^{\mathrm{dl}}}{h^{\mathrm{ou}}} \frac{\mathrm{d}\overline{\mathbb{DL}}}{\mathrm{d}\overline{\mathbb{O}}} &= \frac{g^{\mathrm{dl}}}{h^{\mathrm{bm}}} \frac{\mathrm{d}\overline{\mathbb{DL}}}{\mathrm{d}\overline{\mathbb{W}}} \frac{\hbar^{\mathrm{bm}}}{g^{\mathrm{ou}}} \frac{\mathrm{d}\overline{\mathbb{W}}}{\mathrm{d}\overline{\mathbb{O}}} \frac{g^{\mathrm{ou}}}{h^{\mathrm{ou}}} \frac{h^{\mathrm{bm}}}{\hbar^{\mathrm{bm}}} \\
&\propto \frac{E^{\mathrm{bm}}}{E_*^{\mathrm{ou}}} \frac{\prod_{c=1}^{C}\big[e^{2A_c^{\mathrm{ou}}(\boldsymbol{x}^{(c)})} p^{\mathrm{ou}}_{T,c}(\boldsymbol{y} \mid \boldsymbol{x})/\mathrm{e}^{A_c^{\mathrm{ou}}(\boldsymbol{y})}\big]}{\big[\prod_{c=1}^{C} f_c(\boldsymbol{x}^{(c)})\big] \operatorname{etr}[-[\boldsymbol{y}-\tilde{\boldsymbol{x}}]^{\otimes 2}\boldsymbol{D}/2]} \frac{\big[\prod_{c=1}^{C} f_c(\boldsymbol{x}^{(c)})\big]}{\big[\prod_{c=1}^{C} \mathrm{e}^{A_c^{\mathrm{ou}}(\boldsymbol{x}^{(c)})}\big]} \\
&= \frac{E^{\mathrm{bm}}}{E_*^{\mathrm{ou}}} \frac{\tilde{g}^{\mathrm{ou}}(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y})}{\big[\prod_{c=1}^{C} f_c(\boldsymbol{x}^{(c)})\big] \operatorname{etr}[-[\boldsymbol{y}-\tilde{\boldsymbol{x}}]^{\otimes 2}\boldsymbol{D}/2]}.
\end{aligned}$$

Lemma 2 and the expressions for E^{bm} and E_*^{ou} immediately give

$$\frac{g^{\mathrm{dl}}}{h^{\mathrm{ou}}}=\frac{E^{\mathrm{bm}}}{E_*^{\mathrm{ou}}}\cdot\rho^{\mathrm{ou}}\propto E^{\mathrm{ou}}\cdot\rho^{\mathrm{ou}},$$

where E^{ou} is given in (14). □

Appendix B. Proof of Lemma 1

We consider the formula for ρ^{ou} in (12). If $T=\infty$, we have

$$\boldsymbol{m}_c=\hat{\boldsymbol{\mu}}_c,\qquad \boldsymbol{V}_c=\frac{\hat{\boldsymbol{\Lambda}}_c^{-1}}{2},$$

and

$$\boldsymbol{D}=\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c,\qquad \boldsymbol{M}_{1,c}=\mathrm{e}^{2\hat{\boldsymbol{\Lambda}}_c T}\hat{\boldsymbol{\Lambda}}_c-2\hat{\boldsymbol{\Lambda}}_c,$$
$$\boldsymbol{M}_{2,c}=4\hat{\boldsymbol{\Lambda}}_c-2\mathrm{e}^{2\hat{\boldsymbol{\Lambda}}_c T}\hat{\boldsymbol{\Lambda}}_c=-2\boldsymbol{M}_{1,c}.$$

Therefore,

$$\begin{aligned}\lim_{T\to\infty}\boldsymbol{M}_{1,c}(\boldsymbol{m}_c+\boldsymbol{M}_{1,c}^{-1}\boldsymbol{M}_{2,c}\boldsymbol{V}_c\hat{\boldsymbol{\Lambda}}_c\hat{\boldsymbol{\mu}}_c)^{\otimes 2}&=\lim_{T\to\infty}\Big(\boldsymbol{M}_{1,c}^{1/2}\hat{\boldsymbol{\mu}}_c+\boldsymbol{M}_{1,c}^{-1/2}\boldsymbol{M}_{2,c}\frac{\hat{\boldsymbol{\mu}}_c}{2}\Big)^{\otimes 2}\\&=\boldsymbol{0},\end{aligned}$$

and, furthermore, $\rho^{\mathrm{ou}}(\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(C)})$ becomes a value not depending on $\boldsymbol{X}_0^{(1:C)}$ at all as $T\to\infty$. Therefore, with $T=\infty$, the density function $\tilde{h}^{\mathrm{ou}}(\cdot)$ becomes

$$\tilde{h}^{\mathrm{ou}}\cdot(\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(C)},\boldsymbol{y})\propto\Bigg[\prod_{c=1}^{C}f_c(\boldsymbol{x}^{(c)})\Bigg]\operatorname{etr}\Bigg[-\frac{1}{2}[\boldsymbol{y}-\tilde{\boldsymbol{x}}]^{\otimes 2}\Big(\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\Big)\Bigg],$$

with

$$\tilde{\boldsymbol{x}}=\Big(\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\Big)^{-1}\Big\{\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\hat{\boldsymbol{\mu}}_c\Big\}.$$

Then we can generate $\boldsymbol{y}$ as, with some standard Gaussian random errors $\boldsymbol{\varepsilon}$,

$$\begin{aligned}\boldsymbol{y}&=\tilde{\boldsymbol{x}}+\Big(\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\Big)^{-1/2}\cdot\boldsymbol{\varepsilon}\\&=\Big(\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\Big)^{-1}\Big\{\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\hat{\boldsymbol{\mu}}_c+\Big(\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\Big)^{1/2}\boldsymbol{\varepsilon}\Big\}\\&\overset{\mathrm{D}}{=}\Big(\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\Big)^{-1}\Big\{\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\hat{\boldsymbol{\mu}}_c+\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c^{1/2}\boldsymbol{\varepsilon}_c\Big\}\\&=\Big(\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c\Big)^{-1}\Big\{\sum_{c=1}^{C}\hat{\boldsymbol{\Lambda}}_c(\hat{\boldsymbol{\mu}}_c+\hat{\boldsymbol{\Lambda}}_c^{-1/2}\boldsymbol{\varepsilon}_c)\Big\}\end{aligned}$$

where '$\overset{\mathrm{D}}{=}$' denotes equality in distribution, and $\boldsymbol{\varepsilon}_c$, $c = 1, \ldots, C$, means C independent standard normal vectors.

By noting that $\hat{\boldsymbol{\mu}}_c + \hat{\boldsymbol{\Lambda}}_c^{-1/2}\boldsymbol{\varepsilon}_c$ has the same distribution as $\boldsymbol{x}^{(c)}$ if $f_c(\boldsymbol{x})$ is a Gaussian distribution, the lemma is proved.

Acknowledgements

The authors would like to thank the Isaac Newton Institute for Mathematical Sciences for support and hospitality during the programme 'Scalable inference; statistical, algorithmic, computational aspects (SIN)' when work on this paper was undertaken. MP and GOR were supported by the EPSRC under grant EP/K014463/1. GOR was additionally supported by the EPSRC under grants EP/K034154/1 and EP/D002060/1.

References

[1] Agarwal, A. and Duchi, J. C. (2012). Distributed delayed stochastic optimization. In *51st IEEE Conference on Decision and Control*, pp. 5451–5452.

[2] Berger, O. J. (1980). *Statistical Decision Theory and Bayesian Analysis*. Springer, New York.

[3] Beskos, A. and Roberts, G. O. (2005). Exact simulation of diffusions. *Ann. Appl. Prob.* **15,** 2422–2444.

[4] Beskos, A., Papaspiliopoulos, O. and Roberts, G.O. (2006b), Retrospective exact simulation of diffusion sample paths with applications, *Bernoulli* **12,** 1077–1098.

[5] Beskos A., Papaspiliopoulos O. and Roberts G. O. (2008). A factorisation of diffusion measure and finite sample path constructions. *Methodology Comput. Appl. Prob.* **10,** 85–104.

[6] Beskos A., Papaspiliopoulos O., Roberts G. O. and Fearnhead P. (2006a). Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes (with discussion). *J. R. Statist. Soc. B* **68,** 333–382.

[7] Chen, N. and Huang, Z. (2013). Localisation and exact simulation of Brownian motion-driven stochastic differential equations. *Math. Operat. Res.* **38,** 591–616.

[8] Dacunha-Castelle, D. and Florens-Zmirou, D. (1986). Estimation of the coefficients of a diffusion from discrete observations. *Stochastics* **19,** 263–284.

[9] Dai H. (2014). Exact simulation for diffusion bridges: an adaptive approach. *J. Appl. Prob.* **51,** 346–358.

[10] Dai H. (2017). A new rejection sampling method without using hat function. *Bernoulli* **23,** 2434–2465.

[11] Dai, H., Pollock, M. and Roberts, G. (2019). Monte Carlo fusion. Supplementary material. Available at http://doi.org/10.1017/jpr.2019.12.

[12] Fleiss J. L. (1993). Statistical basis of meta-analysis. *Statist. Methods Med. Res.* **2,** 121–145.

[13] Genest, C. and Zidek, J. V. (1986). Combining probability distributions: a critique and an annotated bibliography. *Statist. Sci.* **1,** 114–148.

[14] Hansen, N. R. (2003). Geometric ergodicity of discrete-time approximations to multivariate diffusions. *Bernoulli* **9,** 725–743.

[15] Li, C., Srivastava, S. and Dunson, D. B. (2017). Simple, scalable and accurate posterior interval estimation. *Biometrika* **104,** 665–680.

[16] Li, Y., Yang M. and Zhang Z. (2015). Multi-view representation learning: A survey from shallow methods to deep methods. *J. Latex Class Files* **14,** 20pp.

[17] Masuda H. (2004). On multidimensional Ornstein-Uhlenbeck processes driven by a general Lévy process. *Bernoulli* **10,** 97–120.

[18] Minsker, S., Srivastava, S., Lin, L. and Dunson, D. B. (2014). Scalable and robust Bayesian inference via the median posterior. In *Proc. 31st Internat. Conference on Machine Learning*, pp. 1656–1664.

[19] Neiswanger W., Wang C. and Xing E. P. (2014). Asymptotically exact, embarrassingly parallel MCMC. In *Proc. 13th Conference on Uncertainty In Artificial Intelligence*, pp. 623–632.

[20] Pollock, M. (2013). Some Monte Carlo methods for jump diffusions. Doctoral Thesis, University of Warwick.

[21] Pollock, M., Johansen, A. M. and Roberts, G. O. (2016b). On the exact and ε-strong simulation of (jump) diffusions. *Bernoulli* **22,** 794–856.

[22] Pollock M, Fearnhead P., Johansen A. M. and Roberts G. O. (2016a). The scalable Langevin exact algorithm: bayesian inference for big data. Submitted to *J. R. Statist. Soc. B*.

[23] Scott, S. L. *et al.* (2016). Bayes and big data: the consensus Monte Carlo algorithm. *Internat. J. Manag. Sci. Eng. Manag.* **11,** 78–88.

[24] Smith, T. C., Spiegelhalter, D. J. and Thomas, A. (1995). Bayesian approaches to random-effects meta-analysis: a comparative study *Statist. Med.* **14,** 2685–2699.

[25] Srivastava, S., Cevher, V., Dinh, Q. and Dunson, D. (2016). WASP: scalable Bayes via barycenters of subset posteriors. In *Proc. 18th Internat. Conference on Artificial Intelligence and Statistics*, pp. 912–920.

[26] Stamatakis A. and Aberer A. J. (2013). Novel parallelization schemes for large-scale likelihood-based phylogenetic inference. In *Proc. 2013 IEEE 27th Internat. Symposium on Parallel and Distributed Processing*, pp. 1195-1204.

[27] Tan, A., Doss, H. and Hobert, J. P. (2015). Honest importance sampling with multiple Markov chains. *J. Comput. Graph. Statist.* **24,** 792–826.

[28] Wang X. and Dunson, D. B. (2014). Parallelizing MCMC via Weierstrass sampler. Preprint. Available at https://arxiv.org/abs/1312.4605v2.

[29] Zhao J., Xie X., Xu X. and Sun S. (2017). Multi-view learning overview: recent progress and new challenges. *Inf. Fusion* **38,** 43–54.