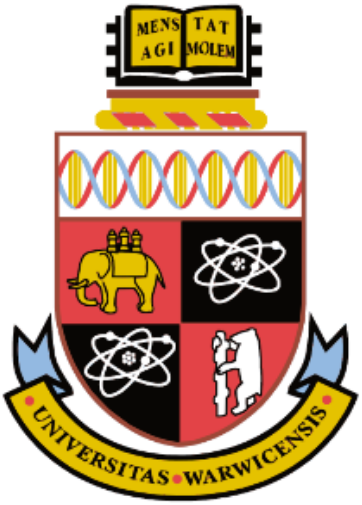


Monte Carlo methods for combining sample approximations of distributions

by

Ryan Sze-Yin Chan

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy in Statistics (Research)

Department of Statistics

October 2022

Contents

List of Algorithms

List of Figures

Acknowledgments

I would like to express my unbounded gratitude to my supervisors, Gareth Roberts and Murray Pollock, for their continued support, encouragement and unfailing patience throughout my PhD. I am extremely grateful for their guidance over the last four years and consider myself lucky to have been given the opportunity to work with them.

I would also like to thank Adam Johansen and Hongsheng Dai for many stimulating discussions around the work in this thesis and their assistance throughout my PhD. I owe a special thanks to Petros Dellaportas for his support early in my PhD. A further thanks to Adam Johansen, Gechun Liang, Krzysztof Łatuszyński for taking the time to read my reports which lead to this thesis.

I have been fortunate enough to have met many wonderful colleagues at both The Alan Turing Institute and within the statistics department at Warwick, and I am thankful to all of them for making these great places to study. For the friends I have made along the way, I am grateful for their friendship and solidarity throughout the PhD experience.

Financial support received from The Alan Turing Institute Doctoral Studentship (EP/N510129/1) is gratefully acknowledged. Further thanks to the Research Engineering Team at The Alan Turing Institute for their support and expertise in parallel computing; the research carried out in this thesis was supported in part through computational resources provided by The Alan Turing Institute.

Much of my PhD was spent working from home and thus I would be remiss not to express a heartfelt thanks to Tommy, Jasper, Jess and Harry, for being amazing housemates. I also extend this thanks to my parents, Samantha and Marvin for the time that I moved back to Wrexham. Thanks to the B.F.C. for their continued support and friendship. Many thanks to them all for keeping me sane over the last four years!

I thank my partner, Giulia, for always being there for me and supporting me along this journey. Finally, a special thank you to my family for always believing in me. It is only with their unconditional support and motivation that any of this has been possible.

Signed: *Ryan Chan*, *October 5, 2022*

Many thanks to Nicolas Chopin and Krzysztof Łatuszyński for taking the time to read and examine my thesis.

Signed: *Ryan Chan*, *February 9, 2023*

Declarations

I hereby declare that this thesis is the result of my own work and research, unless otherwise stated. I confirm that my work has been prepared in accordance with the guidelines set by the University of Warwick on the presentation of a research thesis. This thesis has not been submitted for examination to any institution other than the University of Warwick.

The material in Chapter 6 appears largely in the following:

- Ryan S. Y. Chan, Murray Pollock, Adam M. Johansen, and Gareth O. Roberts. Divide-and-Conquer Monte Carlo Fusion. Statistics e-print 2110.07265, arXiv, 2021.

In this work Chan is the primary author and led on the writing, research and simulations. Pollock & Roberts provided guidance as doctoral supervisors, and Johansen provided additional support on the theory associated with Divide-and-Conquer Sequential Monte Carlo based upon his earlier work in Lindsten et al. [2017].

The material in Chapter 7 will form the entirety of a paper (which has not yet been submitted). Again, Chan is the primary author and led on the writing, research and simulations. Pollock & Roberts provided guidance as doctoral supervisors.

Signed: *Ryan Chan, October 5, 2022*

Abstract

Combining several (sample approximations of) distributions, which we term *sub-posteriors*, into a single distribution proportional to their product, is a common challenge in statistics and data science. For instance, this can occur in distributed 'big data' problems, tempering problems, or when working under multi-party privacy constraints. Many existing approaches resort to approximating the individual sub-posteriors for practical necessity, then finding either an analytical approximation or sample approximation of the resulting (product-pooled) posterior. The quality of the posterior approximation for these approaches is poor when the sub-posteriors fall out-with a narrow range of distributional form, such as being approximately Gaussian. Recently, a *Fusion* approach has been proposed which finds a direct and exact Monte Carlo approximation of the posterior (as opposed to the sub-posteriors), circumventing the drawbacks of approximate approaches. Unfortunately, existing Fusion approaches have a number of computational limitations, particularly when unifying a large number of sub-posteriors or when the sub-posteriors exhibit large correlation. In this thesis, we generalise the theory underpinning existing Fusion approaches, and embed the resulting methodology within a recursive divide-and-conquer sequential Monte Carlo paradigm. This ultimately leads to a competitive Fusion approach, which is appreciably more robust and scalable in a variety of practical settings.

Chapter 1

Introduction

Figure 1.1: The *Fusion Dance* from Dragon Ball Z (1989).

1.1 The Fusion problem

Combining several (sample approximations of) distributions, which we term *sub-posteriors*, into a single distribution proportional to their product, is a common challenge which arises in various settings within statistics. Consider the following d-dimensional (*product-pooled*) target density (which we term the *fusion density*),

$$f(\boldsymbol{x}) \propto f_1(\boldsymbol{x}) \cdots f_C(\boldsymbol{x}) = \prod_{c=1}^{C} f_c(\boldsymbol{x}), \tag{1.1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$, $f_c(\boldsymbol{x})$ for $c = 1, \ldots, C$, represent the individual densities which we wish to unify, and C represents the total *number* of distributions to be unified. Typically, there is no closed form analytical approach to unifying the sub-posterior densities, and so a Monte Carlo approach

is proposed. For convenience, and common to many existing approaches, we typically assume that we have access to independent samples from each sub-posterior. However, in practice, this is often an unrealistic scenario and more commonly we will have access to samples from a Markov chain Monte Carlo (MCMC) algorithm, which will only have a distribution approximately from each sub-posterior. However, we will see that this assumption is not a limiting factor of the methodology developed in this thesis.

The need to unify several (sample approximations of) distributions, over a common parameter space into a single sample approximation of the distribution in the manner of (1.1) is surprisingly common. For instance, it is a problem which classically arises in expert elicitation [Albert et al., 2012; Berger, 1980; Genest and Zidek, 1986], where the distributional views of multiple experts on a topic are pooled into a single view and in meta-analysis [Fleiss, 1993], where the goal is to systematically merge the findings of several independent studies.

The majority of the recent methodological developments for representing or sampling from (1.1) have been focused on tackling distributed *'big data'* problems (see for instance, Scott et al. [2016]; Neiswanger et al. [2014]; Wang and Dunson [2013]; Minsker et al. [2014]; Srivastava et al. [2015]; Entezari et al. [2018]; Rabinovich et al. [2015]; Nemeth and Sherlock [2018]; Rendell et al. [2020]). As the amount of data stored by individuals and organisations grow, statistical models have advanced in complexity and size. This is a particular problem in statistical inference since the computational cost of typical MCMC algorithms for parametric inference, such as Metropolis-Hastings (Metropolis et al. [1953], Hastings [1970]), scale poorly with increasing amounts of data since they usually require access to the full dataset at each iteration Bardenet et al. [2017]. One possible solution to this problem is to use a *fork-and-join* approach, in which the data is split across a number of cores (say C cores) and inference is separately conducted on each core (often using MCMC). The respective methodologies then attempt to unify the sample approximations from each sub-posterior as per (1.1). An advantage of this approach is that inference on each subset of the data can be conducted independently and in parallel. In practice, if one had access to a large cluster of computing cores, then the computational cost of sampling from our target density could be significantly reduced. The main difficulty of these methods is in recombining the individual analyses into a single inference on the full dataset that is both accurate and computationally efficient.

We note here that there are alternative approaches proposed in the literature around improving the scalability of MCMC algorithms for Bayesian inference. For instance there are several *sub-sampling* approaches (e.g. Welling and Teh [2011]; Maclaurin and Adams [2014]; Ma et al. [2015]; Quiroz et al. [2018]; Bouchard-Côté et al. [2018]; Bierkens et al. [2019, 2020]; Pollock et al. [2020]) which aim to reduce the number of data point likelihood evaluations necessary at each iteration of the algorithm. Alternatively, variational inference (VI) [Jordan et al., 1999; Wainwright and Jordan, 2008] is a widely used method from the machine learning literature for approximating Bayesian posterior densities which tends to be faster than MCMC and easier to scale to large datasets [Blei et al.,

2017]. Rather than approximating the target posterior distribution with samples from a Markov chain (whose stationary distribution coincides with the target distribution), VI approximates the posterior with the result of an optimisation algorithm (e.g. stochastic optimisation; see for instance Robbins and Monro [1951]; Hoffman et al. [2013]). As Blei et al. [2017] notes, MCMC methods tend to be more computationally intensive but provides guarantees of producing (asymptotically) exact samples from the target posterior, whereas VI does not enjoy such guarantees. Moreover, Monte Carlo approaches are typically preferable in applications where precise uncertainty quantification is required. Consequently, we focus on developing a Monte Carlo approach in this thesis rather than looking at developing VI methods. Furthermore, in this thesis, we are not solely focused on improving the scalability of methods for Bayesian inference and hence will not be focusing on the sub-sampling MCMC methodologies mentioned above either. We are instead mainly focused on developing robust methodology to tackle the general *fusion problem* outlined by (1.1) which subsequently can be used as an approach for performing Bayesian inference with large datasets.

The fusion problem can also arise in settings where the target distribution exhibits *multi-modality*. In particular, MCMC algorithms typically use localised proposal mechanisms which can exacerbate the difficulties of moving between modes. This localisation can result in the Markov chain becoming trapped in a subset of the state space, despite the Markov chain satisfying all ergodicity properties. With finite computation, the Markov chain can fail to explore all regions of significant probability mass which can lead to biased samples (e.g. see Geyer [1991]; Geyer and Thompson [1995]; Tawn and Roberts [2019]). In such settings, we consider the *power-tempered target distribution* which is the target distribution, π, at some inverse temperature level β, for $\beta \in (0, 1]$, denoted $\pi_\beta(\boldsymbol{x}) \propto [\pi(\boldsymbol{x})]^\beta$. Tempering the target distribution effectively *flattens* the distribution so that MCMC sampling for π_β is simpler. Forming a fusion problem for this problem consists of choosing β such that $\frac{1}{\beta} \in \mathbb{N}^+$ and Markov chain sampling from π_β can mix well across the entire sample space, and by noting

$$\pi(\boldsymbol{x}) = \pi(\boldsymbol{x})^{\frac{1}{\beta}\cdot\beta} \propto \prod_{i=1}^{\frac{1}{\beta}} \pi_\beta(\boldsymbol{x}). \tag{1.2}$$

The fusion problem has also proven to be challenging methodologically in a number of modern settings due to problem specific constraints. These include when dealing with the *privacy constraints* of the individual sources [Yıldırım and Ermiş, 2019], in cases where the sheer *number of sources* is overwhelming, or if the *networking constraints* of the sources are truly *distributed* [Scott et al., 2016]. In particular, fork-and-join methodologies will typically have additional hardware constraints such as minimising or removing communications between computation cores to reduce the effect of *latency* [Scott et al., 2016; Dai et al., 2019]. This in turn has motivated a range of problem specific and pragmatic *approximations*. These approximations are invariably distributional, and typically imposed on each of the sub-posterior distributions (for instance, the sub-posteriors being approximately Gaussian). Such approximations limit the applicability of these methodological approaches

to particular settings, and the unified results can be poorly understood, and even misleading. Alternatively, the *Fusion approach* [Dai et al., 2019, 2021] constructs a direct sample approximation of (1.1), rather than seeking to obtain an ad hoc approximation of f by combining approximations of the sub-posteriors. In this thesis, we focus on developing methodology for an *exact* Monte Carlo approximation of (1.1)—one which provides robust inference in a wide range of practical problems, and yet is amenable to use alongside any problem specific constraints.

The remainder of this introductory chapter is organised as follows: In Section 1.2, we review the existing approaches for the fusion problem of (1.1) and discuss their advantages and disadvantages. In Section 1.3, we provide a summary of the key contributions of this thesis. Lastly, Section 1.4 provides a summary of the structure of this thesis.

1.2 Existing approaches to the fusion problem

As noted above, the majority of methodological developments for tacking the fusion problem has been motivated by performing Bayesian inference with big data. In particular, suppose that we wish to perform inference on a set of parameters $\boldsymbol{x}$ given some conditionally independent data $\boldsymbol{y}$. The fork-and-join approach to performing inference for $\boldsymbol{x}$ begins by splitting the data into C disjoint subsets, $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_C$, and noting that the Bayesian posterior density can be written as

$$p(\boldsymbol{x}|\boldsymbol{y}) \propto \left[\prod_{c=1}^{C} p(\boldsymbol{y}_c|\boldsymbol{x})\right] \cdot p(\boldsymbol{x}), \tag{1.3}$$

where $p(\boldsymbol{y}_c|\boldsymbol{x})$ denotes the likelihood function of $\boldsymbol{y}_c$ given $\boldsymbol{x}$ and the prior distribution is given by $p(\boldsymbol{x})$. In this thesis, we typically have $p(\boldsymbol{x}) = \prod_{c=1}^{C} p(\boldsymbol{x})^{\frac{1}{C}}$ to ensure the amount of total amount prior information in the posterior is preserved [Scott et al., 2016]. This formulation is therefore linked to the general fusion problem given in (1.1) by setting the target density $f(\boldsymbol{x}) \propto p(\boldsymbol{x}|\boldsymbol{y})$ and setting the sub-posteriors as $f_c(\boldsymbol{x}) \propto p(\boldsymbol{y}_c|\boldsymbol{x}) \cdot p(\boldsymbol{x})^{\frac{1}{C}}$. Wang and Dunson [2013, Section 2] provides intuition for splitting the prior distribution in this way by noting:

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{y}) &\propto \left[\prod_{c=1}^{C} p(\boldsymbol{y}_c|\boldsymbol{x})\right] \cdot p(\boldsymbol{x}) \\ &\propto \left[\prod_{c=1}^{C} p(\boldsymbol{x}|\boldsymbol{y}_c)\right] \cdot \left[\frac{p(\boldsymbol{x})}{\prod_{c=1}^{C} p_c(\boldsymbol{x})}\right], \end{aligned} \tag{1.4}$$

where $p(\boldsymbol{x})$ represents the prior distribution for the full dataset and $p_c(\boldsymbol{x})$ represents the prior distribution for subset $c = 1, \ldots, C$. This formulation gives flexibility in the choice of $p_c(\boldsymbol{x})$, but if we require that $p(\boldsymbol{x}) = \prod_{c=1}^{C} p_c(\boldsymbol{x})$, then (1.4) can be written as

$$p(\boldsymbol{x}|\boldsymbol{y}) \propto \prod_{c=1}^{C} p(\boldsymbol{x}|\boldsymbol{y}_c). \tag{1.5}$$

Wang and Dunson [2013] refers to (1.5) as the *independent product equation* (also termed *product density equation (PDE)* in Wang et al. [2015]), which indicates under the independence assumption of the data, the posterior density of the full dataset can be represented by the product of sub-posterior densities if the subsets of the data together form the original dataset. In this section, we will review a number of existing approaches to combining samples from sub-posterior distributions. In particular, we focus on three approximate methodologies for the fusion problem in this thesis: *Consensus Monte Carlo* [Scott et al., 2016; Scott, 2017], a method based on using combining kernel density estimates of the sub-posteriors (which we term *Kernel Density Estimate Monte Carlo*) [Neiswanger et al., 2014] and the *Weierstrass sampler* [Wang and Dunson, 2013]. These methodologies were the earliest methods for tacking the fusion problem and often the most widely used in this literature. We will detail these in Section 1.2.1. We will discuss other approximate approaches in this literature and the weakness of these methods in Section 1.2.2.

1.2.1 Main approximate approaches to the fusion problem

One avenue to recombine sub-posterior samples is to use these samples to first impose an approximation to the sub-posterior distributions, and then combine those approximations. Perhaps the simplest method to combine the sub-posterior draws is to use a Gaussian approximation. In particular, the samples can be used to estimate the mean and variance of each sub-posterior. Since the product of Gaussian distributions is well known, then we can analytically calculate a Gaussian approximation to the fusion target posterior (1.3). This idea was first proposed by Neiswanger et al. [2014, Section 3.1] and the motivation is that as the number of data points grows larger, then Bernstein-von Mises theorem states that the Bayesian posterior will be approximately Gaussian [Le Cam, 1986]. Scott et al. [2016] develops this intuition further with the *Consensus Monte Carlo (CMC)* approach which represents (1.3) by means of a weighted average of sub-posterior samples. In particular, suppose we have N draws from each sub-posterior, $\{\boldsymbol{x}_i^{(c)}\}_{i=1}^N$ where $\boldsymbol{x}_i^{(c)} \sim f_c \propto p(\boldsymbol{y}_c|\boldsymbol{x}) \cdot p(\boldsymbol{x})^{\frac{1}{C}}$, and each sub-posterior is assigned a weight represented by the matrix $\mathbf{W}_c$, then the ith Consensus Monte Carlo posterior draw is obtained by computing

$$\hat{\boldsymbol{x}}_i = \left(\sum_{c=1}^{C} \mathbf{W}_c \right)^{-1} \sum_{c=1}^{C} \mathbf{W}_c \boldsymbol{x}_i^{(c)}, \tag{1.6}$$

for $i = 1, \ldots, N$. CMC is *exact* when each sub-posterior is Gaussian (by setting the weights as $\mathbf{W}_c = \boldsymbol{\Sigma}_c^{-1}$ where $\boldsymbol{\Sigma}_c$ is the covariance matrix for sub-posterior $c = 1, \ldots, C$). In practice, Scott et al. [2016] suggests to use the sample covariance matrix $\hat{\boldsymbol{\Sigma}}_c$ of each sub-posterior c. The central idea of CMC is that even when the sub-posteriors are non-Gaussian, the draw $\hat{\boldsymbol{x}}_i$ will still be a close approximation to the posterior if the sub-posteriors are approximately Gaussian, which is often the case in big data settings [Le Cam, 1986]. Although this approach is simple and computationally efficient, the CMC method has been shown to exhibit large bias in other settings [Wang and Dunson, 2013], (e.g. if skewness or multi-modality are present in the sub-posterior distributions).

Neiswanger et al. [2014] suggests a strategy (which we term the *Kernel Density Estimate Monte Carlo (KDEMC)* approach) based on using *kernel density estimates (KDE)* (see for instance Silverman [1986]; Scott [1992]) to approximate the sub-posterior densities. Given N samples from each sub-posterior $\boldsymbol{x}_i^{(c)} \sim f_c$ for $i = 1, \ldots, N$, then kernel density estimation is a method for providing an estimate $\hat{f}_c$ of f_c, defined as

$$\hat{f}_c(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} K_{\mathbf{H}} \left(\boldsymbol{x} - \boldsymbol{x}_i^{(c)} \right), \tag{1.7}$$

where $K_{\mathbf{H}}(\boldsymbol{x}) = |\mathbf{H}|^{-\frac{1}{2}} K(\mathbf{H}^{-\frac{1}{2}}\boldsymbol{x})$, $\mathbf{H}$ is a $d \times d$ symmetric positive-define matrix known as the *bandwidth* and K is the *kernel* function which is a symmetric d-dimensional multivariate density. While there are several choices for the kernel K, Neiswanger et al. [2014] suggest to use a Gaussian kernel with diagonal bandwidth matrix $h^2\mathbb{I}_d$, where $\mathbb{I}_d$ is the d-dimensional identity matrix. The approximation $\hat{f}_c$ of f_c is then given by

$$\hat{f}_c(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{N}_d \left(\boldsymbol{x} \middle| \boldsymbol{x}_i^{(c)}, h^2\mathbb{I}_d \right), \tag{1.8}$$

where $\mathcal{N}_d(\cdot|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a d-dimensional Gaussian density with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$. While Neiswanger et al. [2014] uses a common variance $h^2\mathbb{I}_d$ for each kernel, there may some performance benefits to consider a diagonal matrix $\boldsymbol{\Lambda}_c$ for each sub-posterior since different parameters may differ considerably in variance for each sub-posterior. Nevertheless, the KDEMC method then approximates the fusion posterior by the product of the KDEs for each sub-posterior,

$$\hat{f}(\boldsymbol{x}) = \prod_{c=1}^{C} \hat{f}_c(\boldsymbol{x}) = \frac{1}{N^C} \prod_{c=1}^{C} \left[\sum_{i=1}^{N} \mathcal{N}_d \left(\boldsymbol{x} \middle| \boldsymbol{x}_i^{(c)}, h^2\mathbb{I}_d \right) \right], \tag{1.9}$$

which is a product of Gaussian mixtures. KDEMC effectively approximates (1.3) by implicitly sampling from the product of non-parametric density estimates. The methodology outlined by Neiswanger et al. [2014, Section 3.2] samples from (1.9) by means of a Metropolis-within-Gibbs approach (see Neiswanger et al. [2014, Algorithm 1]) which we do not detail here. While this approach may improve upon CMC when models move away from the Gaussian setting, kernel density estimation is known to perform poorly in high dimensions, meaning that the performance of KDEMC will degrade as the dimensionality of $\boldsymbol{x}$ increases. Nemeth and Sherlock [2018] further notes that the upper bounds on the mean squared error given in Neiswanger et al. [2014] grows exponentially with the number of sub-posteriors, C, which is a limiting factor in the big data settings where the computational benefit of parallelisation is proportional to the number of available cores.

Wang and Dunson [2013] and Bardenet et al. [2017] notes that the KDEMC method provides a poor approximation to the fusion target density if the supports of the sub-posteriors are almost disjoint. To improve the amount of overlap between the approximations, Wang and Dunson [2013] considers

the product of *Weierstrass transforms* for each sub-posterior, which are simply convolutions of the density with a kernel. While the original Weierstrass transform first introduced in Weierstrass [1885] was stated with the Gaussian kernel, Wang and Dunson [2013] provide a generalisation and approximate each sub-posterior density with

$$\hat{f}_c^W(\boldsymbol{x}) = W_{\mathbf{H}}^K f_c(\boldsymbol{x}) = \int K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{t}) \cdot f_c(\boldsymbol{t}) \, \mathrm{d}\boldsymbol{t}, \tag{1.10}$$

for $c = 1, \ldots, C$. However, the authors do subsequently focus on the Gaussian kernel for convenience. This method approximates (1.3) by the product of Weierstrass transforms, which is sampled by means of a rejection sampler (see Section 2.2) [Wang and Dunson, 2013, Algorithm 1]. Wang and Dunson [2013] call this approach the *Weierstrass Rejection Sampler (WRS)* and claim that using a Weierstrass transform approximation rather than a KDE has a number of better properties such as improvement in performance when the sub-posteriors have little overlapping support, and better scaling with dimensionality. However, these kernel-based approaches can be incredibly sensitive to bandwidth choice as one bandwidth is applied to the whole space [Wang et al., 2015].

In this thesis, we will compare the Fusion methodologies with the three approximate methodologies discussed in this section (CMC [Scott et al., 2016], KDEMC [Neiswanger et al., 2014] and WRS [Wang and Dunson, 2013]). We focus on these three methodologies since these are currently the most well known methods for this problem and we can utilise existing code to implement these methods (see Appendix A for full implementational details of these approaches). We note however that there exists several other approximate methods for the fusion problem outlined in (1.3) which we briefly discuss in the following section.

1.2.2 Alternative approximate approaches to the fusion problem

Whilst the methods discussed in Section 1.2.1 are the most popular and widely used methods for the combination of sub-posterior samples to approximate the full-data posterior density given in (1.3), there has been significant interest in developing alternative approximate approaches for scaling up Monte Carlo sampling for Bayesian inference using a distributed approach. One avenue of research are methods that have been inspired by recent developments in optimal transport theory (see for instance Villani [2009]; Peyré and Cuturi [2019] and references therein). These methods introduce a suitable metric on the space of probability measures (such as the *Wasserstein metric*) and then replace the full posterior density by a geometric combination of the sub-posteriors. Minsker et al. [2014, 2017] focused on creating a method to provide an approach for Bayesian inference which is robust to outliers or corruption in the data. The method targets a 'robust' version of the Bayesian posterior which they define in their paper. In this approach, the authors propose a method based on computing the *geometric median* of a collection of sub-posteriors (which they term the *M-posterior*). However as Bardenet et al. [2017] and Li et al. [2017] note, the robustness of the median posterior estimate advocated here may also be a drawback, as in some

circumstances, valuable information contained in each subset of data may be lost. Srivastava et al. [2015]; Li et al. [2017]; Srivastava et al. [2018] proposed an alternative approach which looked to compute the *Wasserstein barycenter* of sub-posterior distributions (termed the *Wasserstein Posterior (WASP)* by the authors) which can be computed efficiently in practice using techniques developed by Cuturi and Doucet [2014]. A key difference in these approaches is that whereas Scott et al. [2016]; Neiswanger et al. [2014]; Wang and Dunson [2013] proposed to run MCMC over

$$f_c(\boldsymbol{x}) \propto p(\boldsymbol{y}_c|\boldsymbol{x}) \cdot p(\boldsymbol{x})^{\frac{1}{C}}, \tag{1.11}$$

for $c = 1, \ldots, C$, Minsker et al. [2014, 2017]; Srivastava et al. [2015]; Li et al. [2017]; Srivastava et al. [2018] proposed to combine samples from sub-posteriors of the form,

$$f_c(\boldsymbol{x}) \propto p(\boldsymbol{y}_c|\boldsymbol{x})^C \cdot p(\boldsymbol{x}). \tag{1.12}$$

We term these *boosted sub-posteriors* since this corresponds to replicating (or *boosting*) the data in the cth dataset C times to produce pseudo datasets which have the same size as the full dataset and should have the same scale in variance as the full posterior. Indeed, the main difference here is that the likelihood contribution is boosted or inflated by a power C, so that they can be seen as 'noisy' versions of the target full-data posterior, and thus can be treated as a group of estimators of the true posterior. The M-posterior and WASP approaches then consequently attempt to find either the geometric median or the barycenter of these boosted sub-posteriors as an approximation to the target posterior. Whilst these methods can be computationally scalable, the statistical meaning of these median and mean measures is unclear [Bardenet et al., 2017]. Furthermore, these approaches are still fundamentally approximating the target posterior with an alternative measure and thus are not directly sampling from the target posterior.

The *Likelihood Inflating Sampling Algorithm (LISA)* proposed by Entezari et al. [2018] also consider the boosted (or *likelihood inflated*) sub-posterior densities in (1.12) by similarly arguing that these sub-posteriors would be a closer representation of the full-data posterior. Like the Consensus Monte Carlo approach of [Scott et al., 2016], the LISA approach takes a weighted average of sub-posterior samples, and showed that for certain models, it is possible to derive weights that would make their approach lead to a good approximation to the target posterior. For example, it is possible to construct an exact algorithm for Beta-Bernouill models, and to have good approximations for Bayesian linear regression and Bayesian Additive Regression Tree (BART) models. A key drawback of this method is that there currently is no general procedure for combining the sub-posterior samples that will make LISA easy to adapt to a wide variety of models. Indeed, each of these weights for the Beta-Bernoulli, Bayesian linear regression and BART models considered by the authors were derived from the specific model and modified accordingly to give accurate results. Furthermore, as with the CMC approach, taking a simple weighted average of sub-posterior samples will often be too crude of an approximation in many cases.

The *Variational CMC (V-CMC)* approach was introduced by Rabinovich et al. [2015] which views the aggregation of the sub-posterior samples as a variational inference (VI) problem (see for instance Blei et al. [2017] for a review of VI). In contrast, CMC aggregates the sub-posterior samples via weighted averaging. In particular, given N draws from each sub-posterior $\{\boldsymbol{x}_i^{(c)} \sim f_c\}_{i=1}^N$ for $c = 1, \ldots, C$, the ith CMC sample is given by $\hat{\boldsymbol{x}}_i = F(\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_i^{(C)}) = \left(\sum_{c=1}^C \mathbf{W}_c\right)^{-1} \sum_{c=1}^C \mathbf{W}_c \boldsymbol{x}_i^{(c)}$. Rabinovich et al. [2015] notes that the fundamental goal of approximate methods is to choose an aggregation function F such that the induced distribution on $\boldsymbol{x}$ is as close as possible to the target posterior. To this end, V-CMC considers a wider range of aggregation functions and constructs a variational inference problem (i.e. optimisation problem) in order to choose the function with which to aggregate the sub-posterior samples. As with all variational inference methods, the performance of this approach critically depends on choosing a family of distributions which adequately describes the target posterior well but at the same time being simple enough for optimisation to be computationally efficient [Blei et al., 2017; Rabinovich et al., 2015]. In practice, it can be difficult to know how complex your family of distributions needs to be for a given problem.

The *Global CMC (G-CMC)* approach of Rendell et al. [2020] introduces a hierarchical framework to associate an auxiliary parameter with each likelihood contribution for the full-data Bayesian posterior. Given the variable of interest $\boldsymbol{x}$ and C disjoint subsets of data $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_C$, the G-CMC approach introduces C auxiliary variables, $Z_1, \ldots, Z_C$ and defines the probability density function

$$\bar{p}_\lambda(\boldsymbol{x}, z_1, \ldots, z_C) \propto \prod_{c=1}^C \left[K_c^{(\lambda)}(\boldsymbol{x}, z_c) \cdot p(z_c|\boldsymbol{y}_c)\right] \cdot p(\boldsymbol{x}), \tag{1.13}$$

where $\{K_c^{(\lambda)} : \lambda \in \mathbb{R}_+\}$ is a family of Markov transition densities for each $c = 1, \ldots, C$. Defining

$$p^{(\lambda)}(\boldsymbol{x}|\boldsymbol{y}_c) := \int K_c^{(\lambda)}(\boldsymbol{x}, z_c) \cdot p(z_c|\boldsymbol{y}_c) \, \mathrm{d}z_c, \tag{1.14}$$

then the density of the $\boldsymbol{x}$-marginal of $p_\lambda(\boldsymbol{x}, z_1, \ldots, z_C)$ is given by

$$p_\lambda(\boldsymbol{x}) := \int \bar{p}_\lambda(\boldsymbol{x}, z_1, \ldots, z_C) \, \mathrm{d}z_{1:C} \propto \prod_{c=1}^C \left[p^{(\lambda)}(\boldsymbol{x}|\boldsymbol{y}_c)\right] \cdot p(\boldsymbol{x}). \tag{1.15}$$

Rendell et al. [2020] note that if for each $c = 1, \ldots, C$, we have that $p^{(\lambda)}(\boldsymbol{x}|\boldsymbol{y}_c)$ is bounded for all $\lambda > 0$ and $p^{(\lambda)}(\boldsymbol{x}|\boldsymbol{y}_c) \to p(\boldsymbol{x}|\boldsymbol{y}_c)$ pointwise as $\lambda \to 0$, then $p_\lambda \to p$, where p is the full-data posterior given by (1.3), in total variation. Rendell et al. [2020, Section 4] considers a sequential Monte Carlo (see Chapter 3) to approximate a sequence of distributions with densities $\bar{p}_{\lambda_0}, \bar{p}_{\lambda_1}, \ldots$, where $\lambda_0, \ldots, \lambda_n$ is a decreasing sequence since the methodology requires for λ to be sufficiently small such that p_λ is a good approximation of the target posterior p (1.3).

Nemeth and Sherlock [2018] proposed a method which created a Gaussian-process (GP) (see for instance Rasmussen [2003]; Rasmussen and Williams [2006]) approximation for each of the log-sub-

posterior densities and approximating the full log-posterior as a sum of Gaussian processes. In this approach, the authors noted that previous approaches for combining sub-posterior samples have solely relied on the sub-posterior samples outputted from each MCMC algorithm, but have ignored the values of the sub-posterior densities at the samples which are calculated when evaluating the Metropolis-Hastings ratio. The proposed approach places GP priors on the evaluated log-density values of each sub-posterior. The idea here is that the GP on the log-sub-posterior densities provides an estimate of uncertainty in the log-sub-posterior at points where it has not been evaluated at. The resulting approximation (constructed by comining the individual GPs) to the log-posterior density is a sum of GPs which itself is a GP. The success of this method relied on the individual GPs providing a good approximation to the individual log-sub-posterior densities. The method by Nemeth and Sherlock [2018] is therefore an approximate method which avoids directly targeting the target density and combines approximations to the sub-posterior densities.

The *Double Parallel Monte Carlo (DP-MC)* approach of Xue and Liang [2019] first obtains importance weighted samples (see Section 2.3) from the inflated/boosted sub-posteriors given in (1.12). Let $\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(C)}$ denote the (estimated) mean vectors of the respective sub-posterior distributions, and let $\hat{\boldsymbol{\mu}} = \frac{1}{C}\sum_{c=1}^{C} \boldsymbol{\mu}^{(c)}$ be the simple average, then the method proposes to re-centre the sub-posteriors to $\hat{\boldsymbol{\mu}}$ and considers the following mixture to approximate the full-data posterior $p(\boldsymbol{x}|\boldsymbol{y})$:

$$\tilde{p}(\boldsymbol{x}|\boldsymbol{y}) \propto \frac{1}{C}\sum_{c=1}^{C} p\left(\boldsymbol{x} - \hat{\boldsymbol{\mu}} + \boldsymbol{\mu}^{(c)} \middle| \boldsymbol{y}_c\right). \tag{1.16}$$

A drawback with this method is noted by Dai et al. [2021] who states that this methodology relies on the convergence of the posterior to a Gaussian and so performs poorly in scenarios where this is not the case. Further, Dai et al. [2021, Section 5] highlight several example cases (with a logistic regression example) where this method fails to approximate the full-data posterior appropriately.

Other approaches to sampling from (1.3) by combining sub-posterior draws include Wang et al. [2015], which estimates the target posterior by partitioning the space of sub-posterior samples using step functions, and Changye and Robert [2021], which uses random forests to learn approximations to the sub-posteriors and combines them to approximate the full posterior. In the random forests approach of Changye and Robert [2021], they consider combining sub-posteriors of the form:

$$f_c(\boldsymbol{x}) \propto \left[p(\boldsymbol{y}_c|\boldsymbol{x}) \cdot p(\boldsymbol{x})^{1/C}\right]^{\lambda}, \tag{1.17}$$

where λ is not necessarily restricted to 1 (as done in Scott et al. [2016]; Neiswanger et al. [2014]; Wang and Dunson [2013]; Rabinovich et al. [2015]; Nemeth and Sherlock [2018]), or C (as done in Minsker et al. [2014]; Srivastava et al. [2015]; Entezari et al. [2018]). However this method suffers from a curse of dimensionality in the random forest training and consequently needs more sample points to train each random forest learner. Further, Changye and Robert [2021] also note that there currently does not exist any generic method to tune the scale factor λ in the scaled sub-posteriors.

A common theme amongst all these methods discussed so far is that they have been proposed in particular to improve the scalability of Monte Carlo approaches for Bayesian inference and take an approximation approach to combine the sub-posterior samples. Consequently, the theory of these methods are typically asymptotic in the number of data points and usually appeal to the Bernstein-von Mesis theorem. In particular, for these asymptotic regimes, the posterior will tend to a Gaussian distribution (see for instance [Johnson, 1970; Le Cam, 1986]), and therefore is questionable whether many of these approaches offer a significant advantage over simple approaches such as a Laplace approximation to the posterior (as argued by Pollock et al. [2020] and Bardenet et al. [2017]). This therefore motivates a more general approach to the fusion problem as in (1.1) (rather than focusing on the Bayesian inference problem in (1.3)) which avoids any approximation to the sub-posteriors and returns a direct sample approximation to the target fusion density. Indeed, the primary weakness of all these methods discussed thus far is that the recombination of the separately conducted inferences is inexact and involves some approximation of the sub-posteriors.

We end this section by acknowledging that there has been considerable interest in combining machine learning and deep learning models (often referred to as *model fusion* [Claici et al., 2020; Singh and Jaggi, 2020]), whereby the task is to learn from a global machine learning model from a collection of pre-trained local models. This provides an approach to *federated learning (FL)* problems [McMahan et al., 2017; Kairouz et al., 2021; Li et al., 2020] which is a machine learning setting where many clients (e.g. mobile devices or organisations/institutes) collaboratively train a model using a coordinated central server whilst keeping the training data distributed and decentralised. Whilst these methods share the similarity of combining models, we note that these algorithms are tackling a significantly different problem than the one stated here. The fundamental difference being that we are focusing develop Monte Carlo methodology for combining probability distributions (which can represent Bayesian posterior distributions for instance) as per (1.1) using Monte Carlo samples from the sub-posterior distributions. In contrast, these methods look to combine machine learning/deep learning models which are typically trained using stochastic optimisation approaches. For example, McMahan et al. [2017] presented the first of these methods to learn deep neural networks based on model averaging and combining local (stochastic) gradient information to update model parameters. As such, we will not be discussing such methods from the machine learning literature in this thesis.

1.2.3 The Fusion approach

In contrast to these approaches above, the *Fusion approach* [Dai et al., 2019, 2021] constructs a sample approximation of f itself, rather than seeking to obtain an ad hoc approximation to f by combining approximations of the sub-posteriors, $f_1, \ldots, f_C$. Underpinning the Fusion approach is the simple observation that if we sampled (independently) $\boldsymbol{x}^{(c)} \sim f_c$ for $c = 1, \ldots, C$, then conditional on the event that $\boldsymbol{x}^{(1)} = \cdots = \boldsymbol{x}^{(C)}$, we have that $\boldsymbol{x}^{(1)}$ has density f given in (1.1). Clearly the difficulty with exploiting this observation is that we are conditioning on an event of

probability 0. The *Monte Carlo Fusion (MCF)* approach of Dai et al. [2019] provides a framework for practically enforcing this conditioning. This is achieved by initialising C stochastic processes (independently from one another) using a single realisation from each sub-posterior (i.e. $\boldsymbol{X}_0^{(c)} \sim f_c$ for $c = 1, \ldots, C$, where the subscript is a temporal index, noting that $\boldsymbol{X}_0^{(1)} \neq \ldots \neq \boldsymbol{X}_0^{(C)}$), evolving the processes in such a manner that (i) these processes *coalesce* at some fixed future time (i.e. $\boldsymbol{X}_T^{(1)} = \cdots = \boldsymbol{X}_T^{(C)}$); and (ii), the common marginal distribution at the coalescence time, T, is f. In particular, MCF is a rejection sampling approach to sample f by means of sampling from the individual sub-posteriors and a density on an extended space, and so returns independent, identically distributed (i.i.d.) draws from f.

The *Bayesian Fusion (BF)* approach of Dai et al. [2021] re-examined the theoretical underpinnings of MCF by introducing a *stochastic differential equation (SDE)* describing the coalescence of the C stochastic processes, and exploited this theory together with methodology for *sequential Monte Carlo (SMC)* to *gradually* coalesce the stochastic processes. The resulting output of the BF approach is a number of *correlated* and *weighted* draws from f. BF is a far more practical and robust algorithm than MCF. A key advantage of BF over MCF is that it is possible to give considerable user guidance in its implementation.

Since the Fusion approaches consider the wider fusion problem set out by (1.1), we will see that the theory for these approaches do not rely on asymptotic arguments with respect to the number of data points available. We will cover these Fusion methodologies in more detail in Chapter 5. Clearly, the Fusion methods fill a gap in this literature by being the first approach which attempts to provide sample approximations which can directly approximate f (as opposed to sampling from an approximation $\hat{f}$ of f). However, both existing Fusion approaches are computationally expensive to carry out and have key limitations in practice. For instance, the complexity of the methodology is still limited by factors including: (i) the numbers of sub-posteriors being combined; (ii) the level of sub-posterior correlation; (iii) the dimensionality of the sub-posteriors; and (iv) the degree to which the sub-posteriors *conflict*. In this thesis, our main goal will be to develop the Fusion methodology further which consequently is more practical and scalable in practice.

In this thesis, we make two key contributions to address the limitations of MCF and BF: (i) we significantly improve upon the computational efficiency of MCF and BF by allowing the user to incorporate global information about each sub-posterior within the approach, and unify *sub-sets* of the sub-posteriors at any one time; (ii) using the flexibility given by (i) in which sub-posteriors can be partially unified, we embed our improved methodologies within the *divide-and-conquer* paradigm of Lindsten et al. [2017], allowing the user to combine sub-posteriors in *stages* to recover the fusion density f given in (1.1).

1.3 Novel contributions

This thesis will further develop the existing Fusion approaches, so the methodology can be applied to a wider range of applications and problems. We will see that the new Fusion methodologies introduced in this thesis provide greater robustness and scalability in many settings; namely a greater robustness with regards to increasing sub-posterior correlation, increasing number of sub-posteriors, increasing dimensionality, or when combining sub-posteriors which conflict with each other (i.e. sub-posteriors with little overlapping support).

In summary, the main contributions of this thesis are as follows:

- Reformulate the theory underpinning existing Fusion approaches of Dai et al. [2019] and Dai et al. [2021], introducing Generalised Monte Carlo Fusion (GMCF) and Generalised Bayesian Fusion (GBF) approaches in Chapter 6 and Chapter 7 respectively.

- We embed the resulting GMCF and GBF methodologies within a divide-and-conquer paradigm [Lindsten et al., 2017] (leading to the *Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF)* and *Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF)* approaches) by combining the sub-posteriors in stages to recover the fusion density f in (1.1).

- Practical implementational guidance is supplied for our GBF and D&C-GBF approaches to aid practitioners in implementing the methodology.

- We provide extensive simulation studies to illustrate the improvement in the Fusion methodologies in several practical settings.

- Real-data applications are supplied to contrast our methodologies with other competing approaches for combining sub-posterior samples.

1.4 Thesis structure

This thesis in broken into two key parts: Part I is a review of relevant existing literature which is required for the understanding of the methodology developed in this thesis and in Part II, we present our novel contributions to the Fusion methodologies (as previously discussed above).

We begin Part I by reviewing a number of elementary Monte Carlo methods in Chapter 2 which are of particular relevance to the methodology developed in this thesis. Chapter 3 reviews sequential Monte Carlo (SMC) methods and in particular we present the *divide-and-conquer SMC* approach of Lindsten et al. [2017] in Section 3.4. We provide an introductory level overview of methods and theory relating to the path-space simulation of Brownian motion, diffusions and related processes in Chapter 4 as required for Fusion. Lastly, we introduce the existing Fusion methodologies (namely Monte Carlo Fusion [Dai et al., 2019] and Bayesian Fusion [Dai et al., 2021]) in Chapter 5.

Part II consists of two chapters which comprises the main contributions of this thesis. In Chapter 6, we reformulate the theory underpinning the Monte Carlo Fusion approach of Dai et al. [2019] and introduce the *Generalised Monte Carlo Fusion (GMCF)* algorithm. In this chapter, we also embed our GMCF methodology within a *divide-and-conquer* paradigm by combining the sub-posteriors in stages to recover the correct fusion density f, in an approach we term *Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF)*. In Chapter 7, we present the *Generalised Bayesian Fusion (GBF)* and *Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF)* approaches which builds upon these existing Fusion methodologies. In each of these sections, we demonstrate that our newly developed Fusion methodologies offers significant improvements on existing approaches and can be applied in a number of practical settings.

Finally in Chapter 8, we conclude this thesis and provide a summary along with several possible future directions for this work.

Part I

Preliminaries

Chapter 2

Monte Carlo Methods

Monte Carlo methods are a class of statistical algorithms which use the simulation of random processes to draw inference on quantities of interest. We note that many practical problems can be reduced to the computation of an integral. While direct approaches to evaluating integrals analytically can be cumbersome and tedious, Monte Carlo methods are able to utilise advances in modern computing power by constructing a *stochastic* algorithm in order to provide consistent estimates of integrals. For instance, consider expectations of the following form

$$\mathbb{E}_\pi[h(X)] := \int_{\mathbb{R}} h(x) \cdot \pi(x) \, \mathrm{d}x, \tag{2.1}$$

where h is some test function, π is some probability density and X denotes a random variable with law π. By applying the *Strong Law of Large Numbers (SLLN)*, if we were able to draw N independent, identically distributed (i.i.d.) samples $X_1, \ldots, X_N$ from π, we can unbiasedly estimate the expectation $\mathbb{E}_\pi[h(X)]$ using the sample average

$$\widehat{\mathbb{E}_\pi[h(X)]} = \frac{1}{N} \sum_{i=1}^{N} h(X_i), \tag{2.2}$$

since we have

$$\lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} h(X_i) = \mathbb{E}_\pi[h(X)]. \tag{2.3}$$

While this is provides an elegant algorithm to estimate integrals with respect to π, it is not entirely clear how to simulate i.i.d. random samples from the density π. This is the key complication in Monte Carlo methods and in this chapter, we review a number of fundamental Monte Carlo methods which are instrumental in the development of the Fusion methodology presented in this thesis. In particular, we discuss some sampling techniques such as rejection sampling, importance

sampling, series sampling, retrospective Bernoulli sampling and methods to simulate from Poisson processes. A detailed account of these methods can be found in a number of texts (see for instance Robert and Casella [2013]; Devroye [1986]; Kingman [1992]).

2.1 Inversion Sampling

One of the simplest methods for generating random samples from a distribution X with density π and *cumulative distribution function (CDF)*, $F_\pi(x) := \mathbb{P}(X \leq x)$, is based on the *inverse* of the CDF. *Inversion sampling* [Devroye, 1986, Part III, Chapter 2] is a method of sampling from a density of π by *inverting* a random sample from a standard uniform random variable, $u \sim \mathcal{U}[0,1]$.

While the CDF is an increasing function, it is not necessarily continuous and hence we define the *generalised inverse CDF*:

$$F_\pi^{-1}(u) := \inf_x \{F_\pi(x) \geq u\}. \tag{2.4}$$

Since $F_\pi(x) \in [0,1]$ for all $x \in \mathbb{R}$, it is possible to draw random samples from π by generating and transforming a uniform random variable $u \in \mathcal{U}[0,1]$. The key idea behind inversion sampling is given by the following identity:

$$\mathbb{P}\left(F_\pi^{-1}(u) \leq x\right) = \mathbb{P}\left(u \leq F_\pi(x)\right) = F_\pi(x), \tag{2.5}$$

and so F_π is the CDF of $X = F_\pi^{-1}(u)$. This argument is summarised in Algorithm 2.1.1.

Algorithm 2.1.1 Inversion sampling to generate N random samples from $\pi(x)$ [Devroye, 1986, Part III, Chapter 3].

1. For i in 1 to N,
   (a) Simulate $u_i \sim \mathcal{U}[0,1]$.
   (b) Set $X_i = F_\pi^{-1}(u_i)$.
2. Return samples $\{X_i\}_{i=1}^N$.

A key drawback of inversion sampling is that few distributions have a CDF whose (generalised) inverse can be evaluated efficiently. However, the generalised inverse of the CDF is just one possible transformation and there exist other transformation methods that yield samples from distributions. For instance, the Box-Muller method [Box and Muller, 1958] is a transformation method for sampling from the standard Normal distribution.

2.2 Rejection Sampling

Rejection sampling [Von Neumann, 1951; Robert and Casella, 2013] is a general Monte Carlo sampling technique for sampling from some target density π by means of an accessible dominating density q. The main idea is to sample from a *proposal distribution*, q, which is easy to sample from and to reject samples that are "unlikely" to have occurred under the target distribution in some

principled way. The choice of q is made such that π is *absolutely continuous* with respect to q with bounded *Radon-Nikodým* derivative. More formally, if we can find a bound M such that

$$\sup_{x\in\mathbb{R}} \frac{\mathrm{d}\pi}{\mathrm{d}q}(x) \leq M < \infty, \tag{2.6}$$

then if we sample $X \sim q$ and accept the sample ($I = 1$) with probability $P_q(X) := \frac{1}{M}\frac{\mathrm{d}\pi}{\mathrm{d}q}(X) \in [0, 1]$, then $(X|I = 1) \sim \pi$. This argument is summarised in Algorithm 2.2.1.

Algorithm 2.2.1 Rejection sampling to generate N random samples from $\pi(x)$ [Von Neumann, 1951].

1. For i in 1 to N,
   (a) Simulate $X_i \sim q$ and $u \in \mathcal{U}[0, 1]$.
   (b) If $u \leq \frac{\pi(X_i)}{M\cdot q(X_i)}$ then accept X_i, else reject and return to Step 1a.
2. Return samples $\{X_i\}_{i=1}^N$.

Rejection sampling is a powerful technique as it allows us to sample from some inaccessible target density π by sampling from an appropriate proposal density q and applying a correction in the form of only accepting a sample X with probability given by $P_q(X)$. This underlying idea given by the following identity:

$$\pi(x) = \int_0^{\pi(x)} 1 \ \mathrm{d}u = \int_0^{\infty} \underbrace{1_{0<u<\pi(x)}}_{=:\pi(x,u)} \ \mathrm{d}u. \tag{2.7}$$

Therefore, $\pi(x)$ can be interpreted as the marginal density of a uniform distribution on the area under the density π, $\{(x, u) : 0 \leq u \leq \pi(x)\}$. Hence we can generate samples from π by sampling the area under the curve. The difficulty is that it is not always clear how to sample uniformly from this area due to the inaccessibility of π. An intuitive way to understand rejection sampling is to consider the univariate setting. From (2.7), the simulation of N random points $\{x_i\}_{i=1}^N$ from π can be thought as the simulation of N bivariate points $\{(x_i, y_i)\}_{i=1}^N$ under the graph π, where we retain only $\{x_i\}_{i=1}^N$. Since π is inaccessible, the idea is to choose another density q to simulate the locations on the x-axis, $X_1, \ldots, X_N \sim q$, and then the location on the y-axis, $u_1, \ldots, u_N \sim \mathcal{U}[0, M \cdot q(X_i)]$. A point X_i for can be retained if u_i lies under π. An example of this is illustrated in Figure 2.1.

Consider the conditional distribution of $\{X \leq x\}$ given $\{U \leq \frac{\pi(X)}{M\cdot q(X)}\}$ where $U \sim \mathcal{U}[0, 1]$, we have

$$\begin{aligned}
\mathbb{P}\left(X \leq x \middle| u \leq \frac{\pi(X)}{M \cdot q(X)}\right) &= \frac{\int_{-\infty}^{x} \int_0^{\frac{\pi(z)}{M\cdot q(z)}} q(z) \ \mathrm{d}u \ \mathrm{d}z}{\int_{-\infty}^{\infty} \int_0^{\frac{\pi(z)}{M\cdot q(z)}} q(z) \ \mathrm{d}u \ \mathrm{d}z} \\
&= \frac{\int_{-\infty}^{x} \frac{\pi(z)}{M\cdot q(z)} \cdot q(z) \ \mathrm{d}z}{\int_{-\infty}^{\infty} \frac{\pi(z)}{M\cdot q(z)} \cdot q(z) \ \mathrm{d}z}
\end{aligned}$$

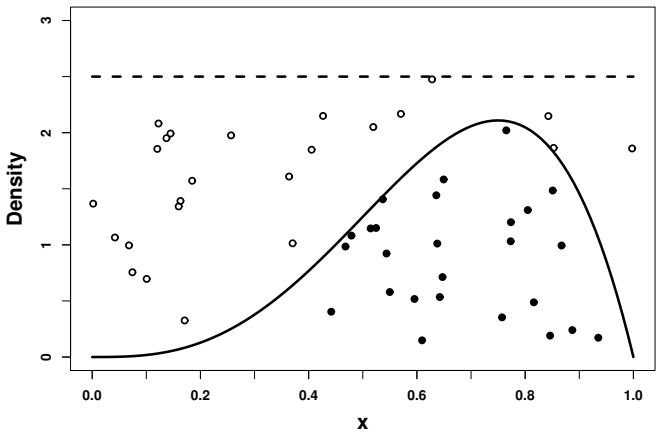


Figure 2.1: An illustration of the simulation of $X \sim \text{Beta}(4, 2)$ (solid line) via rejection sampling using a uniform distribution as the proposal (dotted line) with $M = 2.5$. Empty circles denote rejected samples and filled circles denote accepted proposals.

$$= \frac{\frac{1}{M} \cdot \mathbb{P}_\pi(X \leq x)}{\frac{1}{M} \cdot 1} = F_\pi(x), \tag{2.8}$$

where F_π is the cumulative distribution of π. Furthermore, the probability of acceptance of a proposed sample X is given by

$$\mathbb{E}_q \left[\frac{\pi(X)}{M \cdot q(X)} \right] = \frac{1}{M}. \tag{2.9}$$

This means that the number of draws from q required to obtain a draw from π is Geometrically distributed with mean M. Consequently, rejection sampling can be made computationally more efficient by making M as small as possible by choosing q to be well matched to π.

A key benefit of rejection sampling is that it can still be carried out if we only know π up to a multiplicative constant, i.e. we only know $f(x)$ where $\pi(x) = C \cdot f(x)$, provided that $f(x) < M \cdot q(x)$ for all x, and by accepting a proposal $X \sim q$ with probability

$$P_q(X) = \frac{f(X)}{M \cdot q(X)}.$$

Again considering the conditional distribution of $\{X \leq x\}$ given $\{U \leq \frac{f(X)}{M \cdot q(X)}\}$, we have

$$\begin{aligned} \mathbb{P}\left(X \leq x \middle| u \leq \frac{f(X)}{M \cdot q(X)} \right) &= \frac{\int_{-\infty}^{x} \int_0^{\frac{f(z)}{M \cdot q(z)}} q(z) \, \mathrm{d}u \, \mathrm{d}z}{\int_{-\infty}^{\infty} \int_0^{\frac{f(z)}{M \cdot q(z)}} q(z) \, \mathrm{d}u \, \mathrm{d}z} \\ &= \frac{\int_{-\infty}^{x} \frac{f(z)}{M \cdot q(z)} \cdot q(z) \, \mathrm{d}z}{\int_{-\infty}^{\infty} \frac{f(z)}{M \cdot q(z)} \cdot q(z) \, \mathrm{d}z} \end{aligned}$$

$$= \frac{\int_{-\infty}^{x} \frac{\pi(z)}{C \cdot M} \, \mathrm{d}z}{\int_{-\infty}^{\infty} \frac{\pi(z)}{C \cdot M} \, \mathrm{d}z}$$
$$= \frac{\frac{1}{C \cdot M} \cdot \mathbb{P}_\pi(X \leq x)}{\frac{1}{C \cdot M} \cdot 1} = F_\pi(x). \tag{2.10}$$

2.3 Importance Sampling

In the previous section, we described the rejection sampling approach to sampling from a distribution with density π. However, a rejection sampling algorithm can be wasteful and inefficient in settings where a large number of proposed samples are rejected, which typically occurs when the probability of acceptance is too low. In addition, some useful information about the density is lost upon evaluation whether to accept or reject a particular sample. In rejection sampling, we compensate for the fact that we sampled from a proposal distribution $q(x)$ instead of our target $\pi(x)$ by rejecting some of the proposed values. With *importance sampling* [Kahn, 1949; Goertzel, 1949], we use *weights* to correct for the fact that we sample from the proposal distribution $q(x)$ instead of the target distribution $\pi(x)$. In particular, let $w(X) = \frac{\pi(X)}{q(X)}$ denote the *importance sample weight*, then considering the expectation $\mathbb{E}_\pi[h(X)]$ for some function h, we have

$$\begin{aligned}
\mathbb{E}_\pi[h(X)] &= \int_{-\infty}^{\infty} h(x) \cdot \pi(x) \, \mathrm{d}x \\
&= \int_{-\infty}^{\infty} h(x) \cdot \underbrace{\frac{\pi(x)}{q(x)}}_{=:w(X)} \cdot q(x) \, \mathrm{d}x \\
&= \mathbb{E}_q \left[h(X) \cdot w(X) \right].
\end{aligned} \tag{2.11}$$

This suggests that if we can sample $X_1, \ldots, X_N \sim q$ independently, we can construct a Monte Carlo estimate of $\mathbb{E}_\pi[h(X)]$ by applying the strong law of large numbers using

$$\lim_{N \to N} \frac{1}{N} \sum_{i=1}^{N} h(X_i) \cdot w(X_i) = \mathbb{E}_\pi[h(X)], \tag{2.12}$$

where $X_i \sim q$ for $i = 1, \ldots, N$.

Similarly to the rejection sampling method discussed in Section 2.2, we can construct an asymptotically unbiased estimator of $\mathbb{E}_\pi[h(X)]$ even if we only know π up to a multiplicative constant, i.e. if we only know $\pi(x) = C \cdot f(x)$. In particular, let $w'(X) := f(X)/q(X)$, then we have

$$\frac{\mathbb{E}_q[h(X) \cdot w'(X)]}{\mathbb{E}_q[w'(X)]} := \frac{\int_{-\infty}^{\infty} h(x) \cdot \frac{f(x)}{q(x)} \cdot q(x) \, \mathrm{d}x}{\int_{-\infty}^{\infty} \frac{f(x)}{q(x)} \cdot q(x) \, \mathrm{d}x}$$

$$= \frac{\frac{1}{C}\int_{-\infty}^{\infty} h(x)\cdot\pi(x)\ \mathrm{d}x}{\frac{1}{C}\int_{-\infty}^{\infty}\pi(x)\ \mathrm{d}x}$$
$$= \mathbb{E}_\pi[h(X)]. \tag{2.13}$$

If we can sample $X_1,\ldots,X_N \sim q$ independently, let $w(X_i) := w'(X_i)/\sum_{j=1}^N w'(X_j)$ denote the *normalised* importance weights, then consider the following asymptotically unbiased estimator:

$$\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\frac{h(X_i)\cdot w'(X_i)}{\sum_{j=1}^N w'(X_j)} = \lim_{N\to\infty}\sum_{i=1}^{N} h(X_i)\cdot w(X_i) = \mathbb{E}_\pi[h(X)]. \tag{2.14}$$

This version of importance sampling is called the *self-normalised importance sampling* and the algorithm for this approach is presented in Algorithm 2.3.1.

Algorithm 2.3.1 Importance sampling to generate N random samples to approximate $\pi(x)$ [Kahn, 1949; Goertzel, 1949].

1. For i in 1 to N, simulate $X_i \sim q$ and set $w'(X_i) = \frac{f(X_i)}{q(X_i)}$.
2. For i in 1 to N, set $w(X_i) = w'(X_i)/\sum_{j=1}^N w'(X_j)$.
3. Return weighted samples $\{X_i, w_i\}_{i=1}^N$.

Unlike rejection sampling, importance sampling does not obtain independent samples from the target π. Instead, we obtain a *weighted sample approximation* to approximate the target and integrals with respect to it. An illustration of this technique is shown in Figure 2.2. In this figure, we can see that the proposals that are in regions of low probability density of the target are given very low weight in comparison to the proposals that fall near the mode of the distribution.

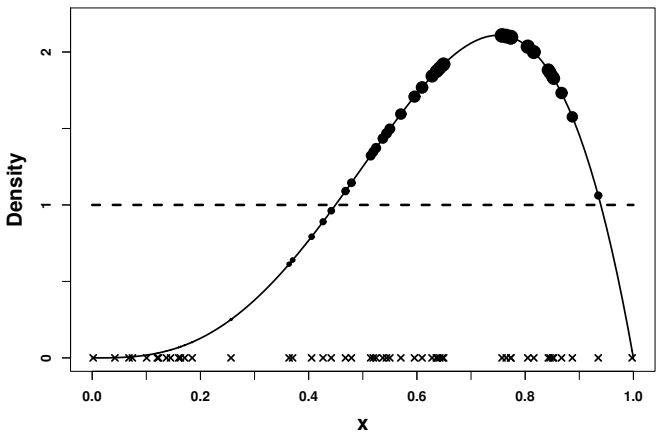


Figure 2.2: An illustration of the simulation of $X \sim \text{Beta}(4,2)$ (solid line) via importance sampling using a uniform distribution as the proposal (dotted line). Crosses denote the proposed samples and the associated weights are plotted with filled dots with their size proportional to their weights.

2.4 Series Sampling

With rejection sampling (see Section 2.2) and importance sampling (see Section 2.3), it is assumed that pointwise evaluations of the target density π are available. However, there are many cases where we are interested in sampling from a density π which cannot be evaluated exactly at any point. *Series sampling* ([Devroye, 1986, Part IV, Chapter 5], [Devroye, 1980]) is a method to draw random samples from a target density π without the need to exactly evaluate π. As with rejection sampling, we assume that there exists an accessible dominating density q. However, we now assume that π can be approximated from above and below by sequences of functions $\pi_n^{\uparrow}$ and $\pi_n^{\downarrow}$ respectively:

1. $\pi(x) \leq M \cdot q(x)$, where M is a constant, for all x,
2. $\lim_{n\to\infty} \pi_n^{\uparrow} = \pi$ and $\lim_{n\to\infty} \pi_n^{\downarrow} = \pi$ such that $\pi_n^{\uparrow} \leq \pi \leq \pi_n^{\downarrow}$ for all n.

Under these conditions, a series sampler can be implemented. The series sampler is similar to rejection sampling in that we first draw a proposal sample from the dominating density, $X \sim q$. However, in this setting, since we cannot evaluate $\pi(X)$ to determine directly whether or not to accept the sample (with probability $P_q(X) := \frac{\pi(X)}{M \cdot q(X)}$), we employ a similar approach as in inversion sampling (see Section 2.1) to simulate *unbiasedly* an event of probability $P_q(X)$. In particular, for any $X \sim q$, we have upper ($\pi_n^{\uparrow}(X)$) and lower ($\pi_n^{\downarrow}(X)$) convergent bounding series that we can iteratively evaluate until a uniform random variable $u \sim \mathcal{U}[0,1]$ lies below $\frac{\pi_n^{\downarrow}(X)}{M \cdot q(X)}$ (hence lies under $P_q(X)$ and we *accept* the sample) or above $\frac{\pi_n^{\uparrow}(X)}{M \cdot q(X)}$ (hence lies above $P_q(X)$ so we *reject* the sample). We can use this to unbiasedly estimate an event of probability $P_q(X)$ in order to determine whether to accept or reject a proposal sample $X \sim q$ as per Algorithm 2.4.1.

Algorithm 2.4.1 Series sampling to generate N random samples from $\pi(x)$ [Devroye, 1986, Part IV, Chapter 5], [Devroye, 1980].

1. For i in 1 to N,
   (a) Simulate $X_i \sim q$ and $u \in \mathcal{U}[0,1]$.
   (b) While $u \in \left(\frac{\pi_n^{\downarrow}(X_i)}{M \cdot q(X_i)}, \frac{\pi_n^{\uparrow}(X_i)}{M \cdot q(X_i)} \right)$, set $n = n + 1$.
   (c) If $u \leq \frac{\pi_n^{\downarrow}(X_i)}{M \cdot q(X_i)}$, accept else return to Step 1a.
2. Return samples $\{X_i\}_{i=1}^N$.

2.5 Retrospective Bernoulli Sampling

Retrospective Bernoulli Sampling [Beskos et al., 2008] is a method to simulate unbiasedly an event of some unknown probability p, where p can be expressed as the limit of an *alternating Cauchy sequence* $(S_k : k \in \mathbb{Z}_{\geq 0})$. In particular, we assume that p can be expressed as the limit of the following series of over and under-estimations:

$$0 < S_2 < S_4 < S_6 < \cdots < p < \cdots < S_5 < S_3 < S_1 \leq 1. \tag{2.15}$$

Without loss of generality, throughout this thesis (unless otherwise stated), we assume that the alternating Cauchy sequence have even terms of the sequence, $(S_{2k} : k \in \mathbb{Z} \geq 0)$, converging from below and the odd terms of the sequence, $(S_{2k+1} : k \in \mathbb{Z} \geq 0)$, converging from above. Since the upper (odd) sub-sequence will be monotonically decreasing and the lower (even) sub-sequence will be monotonically increasing, we can utilise the series sampling (see Section 2.4) and inversion sampling (see Section 2.1) approaches to simulate an event of probability p. In particular, we can draw from a uniform random variable $u \sim \mathcal{U}[0, 1]$ and evaluate the upper and lower sub-sequences until $u \notin (S_{2k}, S_{2k+1})$ and subsequently determine whether u lies above or below p. This argument is summarised in Algorithm 2.5.1.

Algorithm 2.5.1 Retrospective Bernoulli sampling to simulate unbiasedly an event of probability p [Beskos et al., 2008].

1. Simulate $u \sim \mathcal{U}[0, 1]$ and set $k = 1$.
2. While $u \in (S_{2k}, S_{2k+1})$, set $k = k + 1$.
3. If $u \leq S_{2k}$, then return 1, else return 0.

This approach can be extended to cases where p can be represented as the limit of some more general sequences as long as it is possible to find an alternating Cauchy sequence with which to extract upper and lower sub-sequences which monotonically converge to p. In particular, there are instances in this thesis where we require the unbiased simulation of an event of probability p, where p can be represented as a linear transformation of a number of alternating Cauchy sequences. More formally, let $p := f(p_1, \ldots, p_m)$ for some linear function f and $p_1, \ldots, p_m$ can be represented as the limit of alternating Cauchy sequences $(S_k^1, \ldots, S_k^m$, where $k \in \mathbb{Z}_{\geq 0}$ respectively). Retrospective Bernoulli sampling can also be employed in this setting since p can itself be represented as the limit of an alternating Cauchy sequence by aligning the indices of the Cauchy sequences $(S_k^i : i = 1, \ldots, m)$ to ensure that the under and over estimations of p occur on alternating indices. This alignment might require some of the alternating Cauchy sequences to increase their index by 1. Furthermore, it is also possible to extend this approach if it is possible to find a sequence that *eventually* becomes an alternating Cauchy sequence after the inclusion of the first $\hat{k}$ terms, say. In this setting, Algorithm 2.5.1 can be directly applied by modifying Step 1 such that k is initially set to be *greater than* $\hat{k}$.

2.6 Simulating Poisson processes

This section outlines Monte Carlo methods for simulating Poisson processes which are key in the construction of the methodology discussed in this thesis.

Definition 2.6.1. Poisson process. A continuous time stochastic process $\{N(t) : t \geq 0\}$ is a *Poisson process* parametrised with *intensity* (or *rate function*) $\lambda(t)$ if it satisfies the following:

Property 2.6.1. Initial value. $N(0) = 0$.

Property 2.6.2. Poisson distributed number of events. The number of events in a time interval follows a Poisson distribution:

$$N(t+s) - N(t) \sim \text{Poi}\left(\int_t^{t+s} \lambda(u)\ \mathrm{d}u\right). \tag{2.16}$$

Property 2.6.3. Independent increments. The number of events occuring in any two disjoint time intervals are independent: If $r < r+s \leq t < t+s$, then $[N(t+s) - N(t)] \perp\!\!\!\perp [N(r+s) - N(s)]$.

In this section, we focus on how to simulate Poisson process sample paths. A detailed account of Poisson processes can be found in a number of texts (see for instance [Cox and Isham, 1980], [Devroye, 1986, Chapter VI], [Kingman, 1992], [Daley and Vere-Jones, 2003, 2008]).

2.6.1 Time-Homogeneous Poisson process

To simulate sample paths of Poisson processes, it is sufficient to simulate the sample path event times. A Poisson process is *time-homogeneous Poisson process* if it also has the following property:

Property 2.6.4. Independent and identically distributed increments. If $s \leq r < r+s \leq t < t+s$ then $[N(t+s) - N(t)] \perp\!\!\!\perp [N(r+s) - N(r)] \perp\!\!\!\perp N(s) \sim \text{Poi}(\lambda s)$.

In this section, we outline a method for simulating paths of time-homogeneous Poisson processes with constant intensity λ (i.e. $\lambda(t) = \lambda$ for all t). Using Property 2.6.4, for any given sample path, we can directly simulate the number of events that occur in the interval $[0,t]$ since $N(t) \sim \text{Poi}(\lambda t)$. Further, it can be shown that conditional on the number events that occur in $[0,t]$, the event times are uniformly distributed on the interval [Kingman, 1992, Chapter 2.4]: Suppose n is the total number of events that occur in $[0,t]$ ($N(t) = n$), and we are interested in how many of those n events occur in the sub-interval $[0,r] \subset [0,t]$. Since at most $k \leq n$ events could occur in $[0,r]$ and using Property 2.6.4, we have

$$\begin{aligned}
\mathbb{P}(N(r) = k | N(t) = n) &= \frac{\mathbb{P}(N(r) = k) \cdot \mathbb{P}(N(t) - N(r) = n - k)}{\mathbb{P}(N(t) = n)} \\
&= \frac{n!}{(n-k)!k!} \cdot \frac{[\exp(-\lambda r)(\lambda r)^k] \cdot [\exp(-\lambda(t-r))(\lambda(t-r))^{n-k}]}{\exp(-\lambda t)(\lambda t)^n} \\
&= \frac{n!}{(n-k)!k!} \cdot \frac{r^k (t-r)^{n-k}}{t^n} \\
&= \binom{n}{k} p^k (1-p)^{n-k},
\end{aligned} \tag{2.17}$$

where $p = \frac{r}{t}$. Therefore, $\mathbb{P}(N(r) = k | N(t) = n)$ is the probability that k out of n independent $\mathcal{U}[0,t]$ random variables fall in the interval $[0,r]$. This gives a possible algorithm to simulate a time-homogeneous Poisson process which is presented in Algorithm 2.6.1.

Algorithm 2.6.1 Time-homogeneous Poisson process simulation [Kingman, 1992].

1. Simulate $n \sim \text{Poi}(\lambda t)$.
2. If $n \neq 0$,
   (a) Simulate $u_1, \ldots, u_n \overset{\text{iid}}{\sim} \mathcal{U}[0, t]$.
   (b) Set $q_1, \ldots, q_n$ to be the order statistics of the set $\{u_1, \ldots, u_n\}$.
3. Return event times $\{q_i\}_{i=1}^n$.

2.6.2 Time-Inhomogeneous Poisson process

If the intensity of the Poisson process $\lambda(t)$ does depend on time t, then the Poisson process is said to be *time-inhomogeneous*. We can simulate from a time-inhomogeneous Poisson processes with intensity $\lambda(t)$ by simulating a dominating *time-homogeneous* Poisson process with constant intensity Λ (such that for all t, $\lambda(t) \leq \Lambda$) and conduct *Poisson Thinning* (or *colouring* [Kingman, 1992, Chapter 5.1]).

Consider a time-homogeneous Poisson process $\{N(t) : t \geq 0\}$ with intensity Λ where each event is classified either as a 'Type 1' event (with probability p) or a 'Type 2' event (with probability $(1-p)$). Note that the number of 'Type 1' events, $N_1(t)$, given the total number of events, $N(t)$, follows a binomial distribution. Let $N_1(t)$ and $N_2(t)$ denote the number of events classified as 'Type 1' and 'Type 2' respectively and noting $N(t) = N_1(t) + N_2(t)$, we have

$$\begin{aligned}
\mathbb{P}(N_1(t) = n, N_2(t) = m) &= \sum_{k=0}^{\infty} \mathbb{P}(N_1(t) = n, N_2(t) = m | N(t) = k) \cdot \mathbb{P}(N(t) = k) \\
&= \mathbb{P}(N_1(t) = n, N_2(t) = m | N(t) = n + m) \cdot \mathbb{P}(N(t) = n + m) \\
&= \left(\frac{(n+m)!}{n!m!} p^n (1-p)^m \right) \cdot \frac{\exp(-\Lambda t)(\Lambda t)^{n+m}}{(n+m)!} \\
&= \frac{\exp(-\Lambda p t)(\Lambda p t)^n}{n!} \cdot \frac{\exp(-\Lambda(1-p)t)(\Lambda(1-p)t)^m}{m!}, \qquad (2.18)
\end{aligned}$$

and hence $\{N_1(t) : t \geq 0\}$ and $\{N_2(t) : t \geq 0\}$ are two independent Poisson processes. We can view the time-homogeneous Poisson process with intensity Λ as arising from the superposition of a target time-inhomogeneous Poisson process with intensity $\lambda(t)$ and another with intensity $(\Lambda - \lambda(t))$. Any event arising at time q can be assigned to the target Poisson process (with intensity $\lambda(t)$) with probability $\frac{\lambda(q)}{\Lambda}$. This argument is summarised in Algorithm 2.6.2.

Algorithm 2.6.2 Time-inhomogeneous Poisson process simulation [Kingman, 1992].

1. Simulate proposal event times $(p_1, \ldots, p_n)$ of a time-homogeneous Poisson process with intensity Λ as per Algorithm 2.6.1.
2. If $n \neq 0$, set $j = 0$ and for i in 1 to n,
   (a) With probability $\frac{\lambda(p_i)}{\Lambda}$, set $j = j + 1$ and $q_j = p_i$.
3. Return event times $\{q_i\}_{i=1}^n$.

Chapter 3

Sequential Monte Carlo Methods

Sequential Monte Carlo (SMC) methods are a class of Monte Carlo methods that sample *sequentially* from a sequence of target probability densities $\{\pi_n(\boldsymbol{x}_{0:n})\}$ of increasing dimension, where $\boldsymbol{x}_{0:n} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n)$ and each density is defined on a product space $\mathcal{X}^{n+1}$ with

$$\pi_n(\boldsymbol{x}_{0:n}) = \frac{\gamma_n(\boldsymbol{x}_{0:n})}{Z_n}, \tag{3.1}$$

where $\gamma_n : \mathcal{X}^n \to \mathbb{R}^+$ is known pointwise and the *normalising constant*,

$$Z_n = \int \gamma_n(\boldsymbol{x}_{0:n}) \, \mathrm{d}\boldsymbol{x}_{0:n}, \tag{3.2}$$

might be unknown. SMC methodology provides an approximation of $\pi_n(\boldsymbol{x}_{0:n})$ and an estimate of Z_n sequentially for each n, whereby the approach first approximates $\pi_0(\boldsymbol{x}_0)$ and estimates Z_0 at time 0, and then an approximation of $\pi_1(\boldsymbol{x}_{0:1})$ and an estimate of Z_1 in obtained at time 1, and so on. Naturally, this methodology is often used for estimating unknown quantities where observations arrive sequentially in time where it is possible to perform on-line inference by updating the posterior distribution based on incoming data. As such, SMC has a wide variety of applications including signal processing and target tracking [Gordon et al., 1993; Arulampalam et al., 2002], audio enhancement [Godsill et al., 2002; Vermaak et al., 2002], financial modelling [Lopes and Tsay, 2011; Creal, 2012], genetics [Bouchard-Côté et al., 2012] among many others.

In Part II of this thesis, we develop a Fusion methodology for combining sample approximations of distributions that is subsequently embedded within a SMC algorithm. In this chapter, we provide a brief overview of this background material. A number of detailed tutorials and texts of SMC can be found, such as Doucet et al. [2001]; Liu [2001]; Maskell and Gordon [2002]; Chopin [2002]; Doucet and Johansen [2011]; Chopin and Papaspiliopoulos [2020]. In this chapter, we follow the standard SMC notation as used in Doucet and Johansen [2011].

Consider approximating the probability density $\pi_n(\boldsymbol{x}_{0:n})$ for some fixed n. Suppose we have N samples from independent random variables, $X^i_{0:n} \sim \pi_n(\boldsymbol{x}_{0:n})$ for $i = 1, \ldots, N$, then the Monte Carlo method (see Chapter 2) approximates the target distribution (with density $\pi_n(\boldsymbol{x}_{0:n})$) by the *empirical measure*,

$$\hat{\pi}_n(\boldsymbol{x}_{0:n}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X^i_{0:n}}(\boldsymbol{x}_{0:n}), \tag{3.3}$$

where $\delta_{\boldsymbol{y}}(\boldsymbol{x})$ denotes the Dirac delta mass located at $\boldsymbol{y}$. Furthermore, we can approximate any marginal distribution, $\boldsymbol{x}_k$ for $k = 0, \ldots, n$, using

$$\hat{\pi}_n(\boldsymbol{x}_k) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X^i_k}(\boldsymbol{x}_k). \tag{3.4}$$

As discussed in Chapter 2, given N independent samples, $X^i_{0:n} \sim \pi_n(\boldsymbol{x}_{0:n})$ for $i = 1, \ldots, N$, expectations of any test function $h_n : \mathcal{X}^{n+1} \to \mathbb{R}$, given by

$$\begin{aligned} I_n(h_n) &:= \mathbb{E}_{\pi_n}[h_n(X_{0:n})] \\ &= \int h_n(\boldsymbol{x}_{0:n}) \cdot \pi_n(\boldsymbol{x}_{0:n}) \, \mathrm{d}\boldsymbol{x}_{0:n}, \end{aligned} \tag{3.5}$$

can be unbiasedly estimated using the *Monte Carlo estimate*,

$$\begin{aligned} \hat{I}^{MC}_n(h_n) &:= \int h_n(\boldsymbol{x}_{0:n}) \cdot \hat{\pi}_n(\boldsymbol{x}_{0:n}) \, \mathrm{d}\boldsymbol{x}_{0:n} \\ &= \frac{1}{N}\sum_{i=1}^{N} h_n(X^i_{0:n}). \end{aligned} \tag{3.6}$$

The variance of this estimator is given by

$$\mathbb{V}\left[\hat{I}^{MC}_n\right] = \frac{1}{N}\left(\int h^2(\boldsymbol{x}_{0:n}) \cdot \pi_n(\boldsymbol{x}_{0:n}) \, \mathrm{d}\boldsymbol{x}_{0:n} - I^2_n(h_n)\right), \tag{3.7}$$

and hence the variance of the Monte Carlo estimator decreases at a rate of $\mathcal{O}(1/N)$ regardless of the dimension of the space $\mathcal{X}^{n+1}$. However, the method can be difficult to implement if $\pi_n(\boldsymbol{x}_{0:n})$ is high dimensional. Further, the cost of sampling exactly from $\pi_n(\boldsymbol{x}_{0:n})$ sequentially for each value of n is typically at least linear. We can address these problems with importance sampling (see Section 2.3) and *sequential importance sampling* which we discuss in Section 3.1. In Section 3.2, we look at the problem of *importance weight degeneracy* which sequential importance sampling approaches will suffer from, and standard resampling techniques that can be employed to remedy this problem. Having introduced the main ideas within SMC, we present a generic SMC algorithm in Section 3.3. Finally, we briefly overview the Divide-and-Conquer Sequential Monte Carlo (D&C-SMC) framework of Lindsten et al. [2017] which generalises the classical SMC framework from sequences to *trees* (like the ones illustrated in Figures 6.1 and 6.2) in Section 3.4.

3.1 Sequential importance sampling

3.1.1 Importance sampling

We first start this section by discussing how importance sampling (see Section 2.3) can be applied in this setting to approximate $\pi_n(\boldsymbol{x}_{0:n})$ in (3.1). To implement an importance sampler, we first introduce a *joint proposal density*, $q_n(\boldsymbol{x}_{0:n})$, such that $\pi_n(\boldsymbol{x}_{0:n}) > 0 \implies q_n(\boldsymbol{x}_{0:n}) > 0$, so we can write the target as

$$\begin{aligned}\pi_n(\boldsymbol{x}_{0:n}) &= \frac{\gamma_n(\boldsymbol{x}_{0:n})}{Z_n} \\ &= \frac{\frac{\gamma_n(\boldsymbol{x}_{0:n})}{q_n(\boldsymbol{x}_{0:n})} \cdot q_n(\boldsymbol{x}_{0:n})}{Z_n} \\ &= \frac{w'_n(\boldsymbol{x}_{0:n}) \cdot q_n(\boldsymbol{x}_{0:n})}{Z_n},\end{aligned} \tag{3.8}$$

where

$$Z_n = \int w'_n(\boldsymbol{x}_{0:n}) \cdot q_n(\boldsymbol{x}_{0:n}) \, \mathrm{d}\boldsymbol{x}_{0:n}, \tag{3.9}$$

and $w'_n(\boldsymbol{x}_{0:n})$ denotes the *un-normalised importance weight* function,

$$w'_n(\boldsymbol{x}_{0:n}) = \frac{\gamma_n(\boldsymbol{x}_{0:n})}{q_n(\boldsymbol{x}_{0:n})}. \tag{3.10}$$

This suggests that if we can sample $X^{(1)}_{0:n}, \ldots, X^{(1)}_{0:n} \sim q_n(\boldsymbol{x}_{0:n})$ independently and assign the samples *normalised importance weights*, $w^{(i)}_n := w'_n(X^{(i)}_{0:n}) / \sum_{j=1}^N w'_n(X^{(j)}_{0:n})$ for $i = 1, \ldots, N$, then we can construct the following empirical Monte Carlo measure,

$$\hat{\pi}_n(\boldsymbol{x}_{0:n}) = \sum_{i=1}^N w^{(i)}_n \cdot \delta_{X^{(i)}_{0:n}}(\boldsymbol{x}_{0:n}), \tag{3.11}$$

and approximate the normalising constant using

$$\hat{Z}_n = \frac{1}{N} \sum_{i=1}^N w'_n(X^{(i)}_{0:n}). \tag{3.12}$$

Further, if we are interested in estimating $I_n(h_n)$ in (3.5) for some test function h, then by applying the same argument as in (2.13), we can use the *importance sampling estimate*,

$$\begin{aligned}\hat{I}^{IS}_n(h_n) &:= \int h_n(\boldsymbol{x}_{0:n}) \cdot \hat{\pi}_n(\boldsymbol{x}_{0:n}) \, \mathrm{d}\boldsymbol{x}_{0:n} \\ &= \sum_{i=1}^N w^{(i)}_n \cdot h_n(X^{(i)}_{0:n}).\end{aligned} \tag{3.13}$$

Doucet and Johansen [2011, Section 3.2] notes that unlike $\hat{I}_n^{MC}$ in (3.6), $\hat{I}_n^{IS}$ is biased for finite sample size N. However, it is consistent and is asymptotically unbiased. Further, if the normalising constant is known analytically, then it is possible to obtain an unbiased importance sampling estimate of $I_n(h_n)$ but this estimator will generally have higher variance.

3.1.2 Sequential importance sampling

For *sequential importance sampling*, we select a proposal distribution which is chosen such that it allows for recursive updates and has the following structure:

$$\begin{aligned} q_n(\boldsymbol{x}_{0:n}) &= q_{n-1}(\boldsymbol{x}_{0:n-1}) \cdot q_n(\boldsymbol{x}_n|\boldsymbol{x}_{0:n-1}) \\ &= q_0(\boldsymbol{x}_0) \prod_{k=1}^{n} q_k(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}). \end{aligned} \tag{3.14}$$

In practice, we draw a sample particle $X_{0:n} \sim q_n(\boldsymbol{x}_{0:n})$ at time n by first sampling $X_0 \sim q_0(\boldsymbol{x}_0)$ at time 0, and then $X_k \sim q_k(\boldsymbol{x}_k|X_{1:k-1})$ at time k for $k = 1, \ldots, n$. The associated un-normalised importance weights also have a recursive form given by the following decomposition

$$\begin{aligned} w'_n(\boldsymbol{x}_{0:n}) &= \frac{\gamma_n(\boldsymbol{x}_{0:n})}{q_n(\boldsymbol{x}_{0:n})} \\ &= \frac{\gamma_{n-1}(\boldsymbol{x}_{0:n-1})}{q_{n-1}(\boldsymbol{x}_{0:n-1})} \cdot \frac{\gamma_n(\boldsymbol{x}_{0:n})}{\gamma_{n-1}(\boldsymbol{x}_{0:n-1}) \cdot q_n(\boldsymbol{x}_n|\boldsymbol{x}_{0:n-1})} \\ &= w'_{n-1}(\boldsymbol{x}_{0:n-1}) \cdot \alpha(\boldsymbol{x}_{0:n}) \\ &= w'_0(\boldsymbol{x}_0) \prod_{k=1}^{n} \alpha_k(\boldsymbol{x}_{0:k}), \end{aligned} \tag{3.15}$$

where we denote $\alpha_n(\boldsymbol{x}_{0:n})$ as the *incremental importance weight function* given by

$$\alpha_n(\boldsymbol{x}_{0:n}) = \frac{\gamma_n(\boldsymbol{x}_{0:n})}{\gamma_{n-1}(\boldsymbol{x}_{0:n-1}) \cdot q_n(\boldsymbol{x}_n|\boldsymbol{x}_{0:n-1})}. \tag{3.16}$$

Applying a similar argument to Section 2.3, we present a general sequential importance sampling algorithm to approximate from $\pi_n(\boldsymbol{x}_{0:n})$ in Algorithm 3.1.1. At any time, n, we can obtain estimates $\hat{\pi}_n(\boldsymbol{x}_{0:n})$ (using (3.11)) and $\hat{Z}_n$ (using (3.12)) for $\pi_n(\boldsymbol{x}_{0:n})$ and Z_n, respectively.

3.2 Sequential importance resampling

A key drawback of the sequential importance sampling approach detailed in Section 3.1 and Algorithm 3.1.1 is that is suffers from *weight degeneracy*, which is the phenomenon whereby the particle set on average becomes increasingly dominated by heavily weighted particles. The variance of resulting estimates increase exponentially with n [Kong et al., 1994]. To combat this, *resampling* techniques are commonly employed.

Algorithm 3.1.1 Sequential importance sampling to generate N random samples to approximate $\pi_n(\boldsymbol{x}_{0:n})$ [Gordon et al., 1993].

1. **Initialisation Step** ($k = 0$):
   (a) For $i = 1, \ldots, N$,
      i. Simulate $X_0^{(i)} \sim q_0(\boldsymbol{x}_0)$.
      ii. Set $w_0'(X_0^{(i)}) = \pi_0(X_0^{(i)})/q_0(X_0^{(i)})$.
   (b) For $i = 1, \ldots, N$,
      i. Set $w_0^{(i)} := w_0'(X_0^{(i)}) / \sum_{j=1}^N w_0'(X_0^{(j)})$.
2. **Iterative Update Steps** (For $k = 1, \ldots, n$):
   (a) For $i = 1, \ldots, N$,
      i. Simulate $X_k^{(i)} \sim q_k(\boldsymbol{x}_k | X_{1:k-1}^{(i)})$.
      ii. Set $X_{0:k}^{(i)} = \left(X_{0:k-1}^{(i)}, X_k^{(i)}\right)$.
      iii. Set $w_k'(X_{0:k}^{(i)}) = w_{k-1}^{(i)} \cdot \alpha_k(X_{0:k}^{(i)})$ as per (3.16).
   (b) For $i = 1, \ldots, N$,
      i. Set $w_k^{(i)} := w_k'(X_{0:k}^{(i)}) / \sum_{j=1}^N w_k'(X_{0:k}^{(j)})$.
3. Return weighted samples $\left\{X_{0:n}^{(i)}, w_n^{(i)}\right\}_{i=1}^N$.

Recall from Sections 2.3 and 3.1, the importance sampling approximation, $\hat{\pi}_n(\boldsymbol{x}_{0:n})$, of the target distribution, $\pi_n(\boldsymbol{x}_{0:n})$ in (3.11), is constructed by proposing samples from a density $q_n(\boldsymbol{x}_{0:n})$ and weighting those samples appropriately. Consequently, importance sampling (and sequential importance sampling) approaches do not provide samples approximately distributed according to $\pi_n(\boldsymbol{x}_{0:n})$. To obtain approximate samples from $\pi_n(\boldsymbol{x}_{0:n})$, we can sample from the importance sampling approximation $\hat{\pi}_n(\boldsymbol{x}_{0:n})$. This is called *resampling*, where we sample from the empirical distribution $\hat{\pi}_n(\boldsymbol{x}_{0:n})$, which itself was obtained by sampling. As a result of resampling, we obtain equally weighted samples, $\{\check{X}_{0:n}^{(i)}, \frac{1}{N}\}_{i=1}^N$, which can be used to approximate the target distribution using the *resampled empirical measure*, denoted

$$\check{\pi}(\boldsymbol{x}_{0:n}) = \sum_{i=1}^N \frac{N_n^{(i)}}{N} \cdot \delta_{X_{0:n}^{(i)}}(\boldsymbol{x}_{0:n}) = \frac{1}{N} \sum_{i=1}^N \delta_{\check{X}_{0:n}^i}(\boldsymbol{x}_{0:n}), \tag{3.17}$$

where $\{\check{X}_{0:n}^{(i)}\}_{i=1}^N$ denote the resampled particles, and $N_n^{(i)}$ denotes the number of *offspring* of each particle $X_{0:n}^{(i)}$ (i.e. the number of times each particle is resampled) for $i = 1, \ldots, N$.

The simplest resampling method is *multinomial resampling* (first introduced by Gordon et al. [1993]), where we select the ith particle, $X_{0:n}^i$, with probability $w_n^{(i)}$ (the *normalised* weight for particle i). The importance weights are then reset after to have equal weighting. If we wanted to obtain N samples, we would simply resample N times from $\hat{\pi}_n(\boldsymbol{x}_{0:n})$ according to the normalised weights $\{w_n^{(i)}\}_{i=1}^n$. Consider the number of offspring, $N_n^{(i)}$, of each particle $X_{0:n}^{(i)}$, then with multinomial resampling, $(N_n^{(1)}, \ldots, N_n^{(N)})$ follows a multinomial distribution parameterised with N trials with probabilities given by the weights, $(w_n^{(i)}, \ldots, w_n^{(N)})$. Other popular resampling methods include

systematic resampling [Carpenter et al., 1999; Fearnhead, 1998], *stratified resampling* [Kitagawa, 1996] and *residual resampling* [Higuchi, 1997; Liu and Chen, 1998; Whitley, 1994]. More details on resampling methods, and comparisons between them, can be found in a number of texts, such as Doucet et al. [2001]; Douc et al. [2005]; Gerber et al. [2019]. With any resampling method, we wish to ensure that it is unbiased (i.e. $\mathbb{E}[N_n^{(i)}|w_n^{(i)}] = N \cdot w_n^{(i)}$ for all $i = 1, \ldots, N$) while minimising the additional variance, $\mathbb{V}[N_n^{(i)}|w_n^{(i)}]$, which is consequently introduced as a result of resampling.

By resampling, we are able to obtain samples distributed approximately according to $\pi_n(\boldsymbol{x}_{0:n})$. However, if we are interested in estimating integrals with respect to $\pi_n(\boldsymbol{x}_{0:n})$, for instance $I_n(h_n)$ in (3.13), we obtain an estimate with lower variance using $\hat{\pi}_n(\boldsymbol{x}_{0:n})$ than that which we would obtain by using the resampled empirical measure, $\check{\pi}_n(\boldsymbol{x}_{0:n})$, since resampling introduces additional variance [Chopin, 2004]. The key advantage of resampling is that we are able to remove particles with low weights with a high probability. In a sequential framework, this can be have computational benefits since we do not carry forward particles with low weights and instead focus our computational efforts on regions of high probability mass. Intuitively, resampling can be seen to be a tool which provides stability of the algorithm in the future at the cost of increase in immediate Monte Carlo variance. Incorporating resampling methodologies within Algorithm 3.1.1 leads to the *sequential importance resampling (SIR)* (sometimes referred to as *sequential importance sampling / resampling (SISR)*) algorithm which is presented in Algorithm 3.2.1.

3.3 A generic sequential Monte Carlo algorithm

Resampling has the effect of removing particles with low weights and multiplying particles with higher weights. However, as noted in the previous section, this comes with the cost of immediately adding variance. In practice, it is more sensible to only resample when we observe weight degeneracy as resampling may be unnecessary otherwise. As proposed by Kong et al. [1994], we can monitor weight degeneracy by computing the *effective sample size (ESS)*. The ESS of a particle set is defined as the equivalent number of independent samples generated directly form the target distribution, which yields the same efficiency in the estimation obtained by importance sampling. Kong [1992] provides a possible mathematical definition which considers the ESS as a function proportional to the ratio between the variance of the Monte Carlo estimator in (3.6) (obtained by drawing samples directly from the target) over the variance of the importance sampling estimate in (3.13) (obtained using any importance sampling approach including sequential algorithms). This heuristic cannot be readily evaluated, however, Kong et al. [1994] proposed an approximation given by

$$\widehat{\mathrm{ESS}} := \left(\sum_{i=1}^{N} \left(w_n^{(i)} \right)^2 \right)^{-1} \in [1, N]. \tag{3.18}$$

The interpretation is that inference based on the N weighted samples is approximately equivalent (in terms of estimator variance) to inference based on ESS direct samples from the target distribution.

Algorithm 3.2.1 Sequential importance resampling to generate N random samples to approximate $\pi_n(\boldsymbol{x}_{0:n})$ [Gordon et al., 1993].

1. **Initialisation Step**:
   (a) For $i = 1, \ldots, N$,
      i. Simulate $X_0^{(i)} \sim q_0(\boldsymbol{x}_0)$.
      ii. Set $w_0'(X_0^{(i)}) = \pi_0(X_0^{(i)})/q_0(X_0^{(i)})$.
   (b) For $i = 1, \ldots, N$,
      i. Set $w_0^{(i)} := w_0'(X_0^{(i)}) / \sum_{j=1}^N w_0'(X_0^{(j)})$.
   (c) For $i = 1, \ldots, N$,
      i. Resample $\check{X}_0^{(i)} \sim \hat{\pi}_0(\boldsymbol{x}_0)$ and set $w_0^{(i)} = \frac{1}{N}$ to obtain weights to obtain N equally weighted particles, $\left\{\check{X}_0^{(i)}, \frac{1}{N}\right\}_{i=1}^N$.
2. **Iterative Update Steps** (For $k = 1, \ldots, n$):
   (a) For $i = 1, \ldots, N$,
      i. Simulate $X_k^{(i)} \sim q_k(\boldsymbol{x}_k | \check{X}_{1:k-1}^{(i)})$.
      ii. Set $X_{0:k}^{(i)} = \left(\check{X}_{0:k-1}^{(i)}, X_k^{(i)}\right)$.
      iii. Set $w_k'(X_{0:k}^{(i)}) = w_{k-1}^{(i)} \cdot \alpha_k(X_{0:k}^{(i)})$ as per (3.16).
   (b) For $i = 1, \ldots, N$,
      i. Set $w_k^{(i)} := w_k'(X_{0:k}^{(i)}) / \sum_{j=1}^N w_k'(X_{0:k}^{(j)})$.
   (c) For $i = 1, \ldots, N$,
      i. Resample $\check{X}_{0:k}^{(i)} \sim \hat{\pi}_0(\boldsymbol{x}_{0:k})$ and set $w_0^{(i)} = \frac{1}{N}$ to obtain weights to obtain N equally weighted particles, $\left\{\check{X}_{0:k}^{(i)}, \frac{1}{N}\right\}_{i=1}^N$.
3. Return weighted samples $\left\{X_{0:n}^{(i)}, w_n^{(i)}\right\}_{i=1}^N$ *or* resampled particles $\left\{\check{X}_{0:n}^{(i)}, \frac{1}{N}\right\}_{i=1}^N$.

In practice, we use (3.18) to determine if resampling is necessary and resample only when $\widehat{\mathrm{ESS}}$ falls below some user-specified threshold, N_{thr}; typically $N_{\mathrm{thr}} = \frac{N}{2}$. This leads to the extension of the SIR algorithm (Algorithm 3.2.1) and gives us a generic SMC algorithm which incorporates adaptive resampling. We summarise these arguments in Algorithm 3.3.1.

Although resampling mitigates some of the effects of weight degeneracy, it can cause *sample degeneracy* which is the phenomenon whereby after sufficiently many time steps, every resampling step reduces the number of unique values representing the variables at the start of the sequence, i.e. $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, etc. For this reason, any SMC algorithm that relies on the distribution of full paths, $\boldsymbol{x}_{0:n}$, will fail for large enough n for any finite sample size. However, as noted by Doucet and Johansen [2011], this problem is a manifestation of a deeper problem which resampling actually mitigates. It is inherently impossible to accurately represent a distribution on a space of arbitrarily high dimension with a sample of fixed, finite sample size. *Sample impoverishment* is a term which is used to describe the situation in which very few different particles have significant weight. This problem can occur if resampling is not utilised since taking the product of many incremental importance weights over many time steps will ultimately lead to high variance in the importance weights.

Algorithm 3.3.1 Sequential Monte Carlo (with adaptive resampling) to generate N random samples to approximate $\pi_n(\boldsymbol{x}_{0:n})$ [Gordon et al., 1993; Kong, 1992; Kong et al., 1994].

1. **Initialisation Step**:
   (a) For $i = 1, \ldots, N$,
      i. Simulate $X_0^{(i)} \sim q_0(\boldsymbol{x}_0)$.
      ii. Set $w_0'(X_0^{(i)}) = \pi_0(X_0^{(i)})/q_0(X_0^{(i)})$.
   (b) For $i = 1, \ldots, N$,
      i. Set $w_0^{(i)} := w_0'(X_0^{(i)}) / \sum_{j=1}^N w_0'(X_0^{(j)})$.
   (c) Compute effective sample size $\widehat{\text{ESS}}$ as per 3.18
      i. If $\widehat{\text{ESS}} \leq N_{\text{thr}}$, for $i = 1, \ldots, N$, resample $\bar{X}_0^{(i)} \sim \hat{\pi}_0(\boldsymbol{x}_0)$ and set $\bar{w}_0^{(i)} = \frac{1}{N}$.
      ii. If $\widehat{\text{ESS}} > N_{\text{thr}}$, for $i = 1, \ldots, N$, let $\bar{X}_0^{(i)} = X_0^{(i)}$ and $\bar{w}_0^{(i)} = w_0^{(i)}$.
2. **Iterative Update Steps** (For $k = 1, \ldots, n$):
   (a) For $i = 1, \ldots, N$,
      i. Simulate $X_k^{(i)} \sim q_k(\boldsymbol{x}_k | \bar{X}_{1:k-1}^{(i)})$.
      ii. Set $X_{0:k}^{(i)} = \left(\bar{X}_{0:k-1}^{(i)}, X_k^{(i)}\right)$.
      iii. Set $w_k'(X_{0:k}^{(i)}) = w_{k-1}^{(i)} \cdot \alpha_k(X_{0:k}^{(i)})$ as per (3.16).
   (b) For $i = 1, \ldots, N$,
      i. Set $w_k^{(i)} := w_k'(X_{0:k}^{(i)}) / \sum_{j=1}^N w_k'(X_{0:k}^{(j)})$.
   (c) Compute effective sample size $\widehat{\text{ESS}}$ as per 3.18
      i. If $\widehat{\text{ESS}} \leq N_{\text{thr}}$, for $i = 1, \ldots, N$, resample $\bar{X}_{0:k}^{(i)} \sim \hat{\pi}_{0:k}(\boldsymbol{x}_{0:k})$ and set $\bar{w}_{0:k}^{(i)} = \frac{1}{N}$.
      ii. If $\widehat{\text{ESS}} > N_{\text{thr}}$, for $i = 1, \ldots, N$, let $\bar{X}_{0:k}^{(i)} = X_{0:k}^{(i)}$ and $\bar{w}_{0:k}^{(i)} = w_{0:k}^{(i)}$.
3. Return weighted samples $\left\{X_{0:n}^{(i)}, w_n^{(i)}\right\}_{i=1}^N$ *or* resampled particles $\left\{\check{X}_{0:n}^{(i)}, \frac{1}{N}\right\}_{i=1}^N$.

It is not possible to circumvent these problems by increasing the number of samples at every iteration to maintain a constant effective sample size, as this would lead to an exponential growth in the number of samples required. Doucet and Johansen [2011] views resampling as 'resetting the system', as its representation of final time marginals remain well behaved at the expense of further diminishing the quality of the path samples. By focusing on fixed-dimensional time marginals, we can circumvent the problem of increasing dimensionality.

3.4 Divide-and-Conquer Sequential Monte Carlo

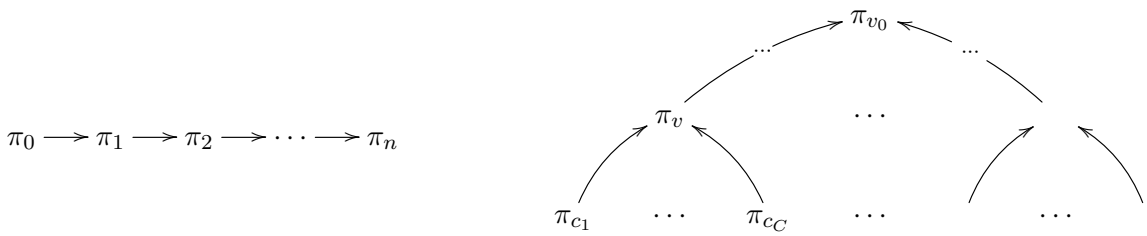
The SMC algorithms discussed so far approximate some sequence of probability distributions of interest, $\{\pi_k(\boldsymbol{x}_k) : k = 0, \ldots, n\}$. This is achieved by simulating a collection of N *normalised* weighted particles $\{X_k^{(i)}, w_k^{(i)}\}_{i=1}^N$ such that the kth marginal distribution can be approximated by the weighted empirical distribution

$$\hat{\pi}_k(\boldsymbol{x}_k) = \sum_{i=1}^N w_k^{(i)} \delta_{X_k^{(i)}}(\boldsymbol{x}_k). \tag{3.19}$$

For SMC methods, the weighted particles are generated sequentially whereby the particles simulated at iteration k depends on the particles generated up to iteration $(k-1)$. Lindsten et al. [2017] notes that for many statistical models, a sequential decomposition (illustrated in Figure 3.1a) might not be the most natural, nor computationally efficient, way of approaching a particular inference problem. Instead, Lindsten et al. [2017] introduced a recursive *divide-and-conquer* approach based upon an auxiliary *tree-structured* decomposition of the inference problem, in which multiple independent populations of weighted particles are resampled, merged and propagated as the method progresses.

In the *Divide-and-Conquer Sequential Monte Carlo (D&C-SMC)* approach [Lindsten et al., 2017], inference on a multivariate distribution is performed by first splitting the collection of variables into disjoint sets and defining suitable auxiliary target distributions for each of these sets. This can have computational benefits since sampling from these distributions is typically easier than sampling from the original distribution and can be done in parallel. The resulting samples are then merged to provide an approximation to the original multivariate distribution of interest. For example, one model class which Lindsten et al. [2017] suggest D&C-SMC can potentially be useful are Bayesian hierarchical models. Essentially, D&C-SMC splits the overall inferential task into a collection of simpler distributions to sample from. At any intermediate iteration of the D&C-SMC algorithm, multiple independent sets of weighted particles are obtained to approximate the intermediate auxiliary target distributions. These are then subsequently merged and propagated as the algorithm progresses towards obtaining a weighted particle set to approximate the original distribution of interest. Standard SMC methodology is employed to propagate the weighted particles at each step of the algorithm. In this section, we describe the D&C-SMC approach outlined in Lindsten et al. [2017].

The D&C-SMC methodology generalises the classical SMC framework from sequences/chains to trees. To illustrate this difference, graph notation is used to describe the execution flow of the algorithm. In Figure 3.1a, a sequence of distributions are organised along a chain, where subsequent distribution are associated with neighbouring nodes on a chain. Each node corresponds to a sequential importance sampling step (which can include resampling steps if necessary), which are labelled by the corresponding target distribution at that step. Arrows illustrate the recursive dependencies of the SMC algorithm. In contrast, a general divide-and-conquer approach, the distributions are organised by a *tree* denoted $\mathbb{T} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$ and (directed) edge set $\mathcal{E}$. We assume that we have a collection of auxiliary distributions $\{\pi_k : k \in \mathcal{V}\}$. Whilst a classical SMC algorithm would have a chain of distributions and $\mathcal{V} = \{0, 1, \ldots, n\}$, we now generalise $\mathcal{V} = \{v_0, v_1, \ldots\}$ to be nodes in a tree. Let $v_0 = \text{Root}(\mathbb{T})$ denote the root of the tree (which represents the distribution or density of interest, i.e. π_{v_0} corresponds to the target density), $\text{Leaf}(\mathbb{T})$ denote the leaves of the tree and $\text{Ch}(v)$ denote the children of vertex $v \in \mathcal{V}$ where $\text{Ch}(t) = \emptyset$ if t is a leaf. The directed edges in the tree are used to illustrate the computational flow of D&C-SMC and an example is provided in Figure 3.1b. Here, each node corresponds to a target distribution $\{\pi_v : v \in \mathcal{V}\}$, which are sampled by merging and propagating samples from the children of each vertex, $\text{Ch}(v) = (c_1, \ldots, c_C)$.



(a) Classical SMC sampler.

(b) D&C-SMC [Lindsten et al., 2017].

Figure 3.1: Illustrative comparison of classical SMC sampler and a D&C-SMC sampler.

In D&C-SMC, weighted sample approximations of auxiliary densities obtained at a particular vertex of a tree are constructed by merging and propagating sample approximations from the children of the vertex. The spaces on which the vertex distributions are defined are constructed recursively,

$$X_v = \left(\otimes_{u \in \mathrm{Ch}(v)} X_c\right) \times \tilde{X}_v, \tag{3.20}$$

where the *incremental* set $\tilde{X}_v$ can be chosen arbitrarily. In the case where $\tilde{X}_v = \emptyset$, X_v is simply the joint distribution of the variables defined by the children of vertex v. SMC methodology is iteratively applied to work through the vertices of the tree from the leaves of the tree to the root, using at each stage the output of one step as the input for the subsequent steps in the algorithm. The D&C-SMC approach takes a "bottom-up" approach where auxiliary target distributions defined by the vertices of the tree are approximated by weighted samples by repeated application of SMC methodology. The algorithm is described by Lindsten et al. [2017] by specifying the operations that are carried out at each vertex of the tree which leads to a recursive definition of the method. For each vertex $v \in \mathcal{V}$, a procedure `dc_smc(v)` (as given in Algorithm 3.4.1) which returns a weighted particle population $\{X_v^{(i)}, w_v^{(i)}\}_{i=1}^N$ to approximate π_v, the normalising constant Z_v (such that $\pi_v(\boldsymbol{x}_v) = \gamma_v(\boldsymbol{x}_v)/Z_v$) and any expectations with respect to π_v.

D&C-SMC begins by obtaining a particle approximation of π_c for each child note $u \in \mathrm{Ch}(v)$ by a recursive call. Jointly these particle populations provide an approximation of the product measure

$$\otimes_{u \in \mathrm{Ch}(v)} \pi_c(\mathrm{d}\boldsymbol{x}_c) \approx \otimes_{u \in \mathrm{Ch}(v)} \hat{\pi}_c^N(\mathrm{d}\boldsymbol{x}_c). \tag{3.22}$$

This point-mass approximation has support on $N^{|\mathrm{Ch}(v)|}$ points, although these points are implicitly given by the $N \cdot |\mathrm{Ch}(v)|$ unique particles (assuming no duplicates among the particles in the individual child populations). Composing all possible permutations of the samples of the children distributions can be performed at the expense of a computational cost of $O(N^{|\mathrm{Ch}(v)|})$. Lindsten et al. [2017] termed this approach *mixture resampling* but noted that in order to obtain a more computationally manageable approximation of the product measure, we can alternatively generate N samples from the approximation in (3.22). This is equivalent to resampling each child particle

Algorithm 3.4.1 Divide-and-Conquer SMC [Lindsten et al., 2017, Algorithm 2]: `dc_smc(v)`.

1. For $u \in \mathrm{Ch}(v)$,
   (a) $\left\{X_u^{(i)}, w_u^{(i)}\right\}_{i=1}^N \leftarrow$ `dc_smc(c)`.
   (b) Resample $\left\{X_u^{(i)}, w_u^{(i)}\right\}_{i=1}^N$ to obtain an equally weighted particle population $\left\{\check{X}_u^{(i)}, 1\right\}_{i=1}^N$.
2. For $i = 1, \ldots, N$,
   (a) If $\tilde{X}_v \neq \emptyset$, simulate $\tilde{X}_v^{(i)} \sim q_v\left(\cdot \middle| \check{X}_{c_1}^{(i)}, \ldots, \check{X}_{c_C}^{(i)}\right)$ where $\mathrm{Ch}(v) := (c_1, c_2, \ldots, c_C)$. Otherwise, set $\tilde{X}_v^{(i)} \leftarrow \emptyset$.
   (b) Set $X_v^{(i)} = \left(\check{X}_{c_1}^{(i)}, \ldots, \check{X}_{c_C}^{(i)}, \tilde{X}_v^{(i)}\right)$.
   (c) Set
$$w_v'\left(X_v^{(i)}\right) = \frac{\gamma_v\left(X_v^{(i)}\right)}{\prod_{u \in \mathrm{Ch}(v)} \gamma_u\left(\check{X}_u^{(i)}\right)} \cdot \frac{1}{q_v\left(\tilde{X}_v^{(i)} \middle| \check{X}_{c_1}^{(i)}, \ldots, \check{X}_{c_C}^{(i)}\right)}. \tag{3.21}$$
3. For $i = 1, \ldots, N$, set $w_v^{(i)} := w_v'(X_v^{(i)}) / \sum_{j=1}^N w_k'(X_v^{(i)})$.
4. Return weighted samples $\left\{X_v^{(i)}, w_v^{(i)}\right\}_{i=1}^N$.

population and creating N equally weighted tuples $\{(\check{X}_{c_1}^i, \ldots, \check{X}_{c_C}^i), 1\}_{i=1}^N$. We would expect this basic merging strategy to perform worse than mixture resampling but this approach can be done in $O(N)$ cost. Alternatively, Lindsten et al. [2017, Section 4.1] also detailed a *lightweight mixture resampling* strategy in which more than one permutation, but not all possible permutations, are used and found it to work well; as noted by Kuntz et al. [2021a] such a strategy can be connected directly with the theory of incomplete U-statistics and consequently one might hope to realise much of the benefit of mixture resampling at a much reduced cost (see e.g. Kong and Zheng [2021]).

The proposal sampling in Step 2a is based on user-provided proposal densities $q_v(\cdot|\check{X}_{c_1}^{(i)}, \ldots, \check{X}_{c_C}^{(i)})$ and has access to the state of all the children $\mathrm{Ch}(v) := (c_1, c_2, \ldots, c_C)$ of vertex v. For each $i = 1, \ldots, N$, the particle tuple $(\check{X}_{c_1}^i, \ldots, \check{X}_{c_C}^i)$ is generated in the resampling stage, and we then sample a incremental variables for vertex v successor state $\tilde{X}_v^i \sim q_v(\cdot|\check{X}_{c_1}^{(i)}, \ldots, \check{X}_{c_C}^{(i)})$. In some cases, parts of the tree structured decomposition do not require this proposal sampling step namely when $\tilde{X}_v = \emptyset$. The ith sample at node v of the tree is then constructed in Step 2b by concatenating the tuple of the resampled child particles $(\check{X}_{c_1}^i, \ldots, \check{X}_{c_C}^i)$ and the proposed incremental state $\tilde{X}_v^i$ (if it is non-empty). These samples are then importance weighted according to (3.21), where the weights are given by the ratio of the un-normalised target densities divided by the proposal density to account for the discrepancy between the target and the proposal with the convention that $\prod_{c \in \emptyset}(\cdot) = 1$ to allow for the importance sampling steps at the leaves of the tree.

Lindsten et al. [2017, Section 3.3] and Kuntz et al. [2021b] provide theoretical results on D&C-SMC which include the unbiasedness of normalising constants estimates (which are inherited from standard SMC algorithms [Moral, 2004, Proposition 7.4.1]), strong law of large numbers, finite sample L_p errors bounds as well as a $\sqrt{N}$-central limit theorem under mild conditions.

Chapter 4

Path-space simulation of Brownian motion and diffusions

This chapter reviews several Monte Carlo methods for simulating sample path trajectories of diffusion processes which are relevant to the methodology developed in this thesis. In particular, the Fusion methodologies discussed in this thesis rely on the computation of unbiased estimators which arise in the simulation of finite dimensional subsets of diffusion sample paths. We begin the chapter with a brief introduction to Brownian motion and related processes, along with algorithms to simulate sample path trajectories of those processes in Section 4.1 and Section 4.2. In Section 4.3, we provide a discussion of diffusions and elements of stochastic calculus which are useful for this thesis. This is followed by an introduction to algorithms for simulating sample path trajectories of a class of diffusions in Section 4.4.

4.1 Simulating Brownian Motion and related processes

Brownian Motion (or a *Wiener Process*) is a continuous time stochastic process which forms the key building block for simulating sample path trajectories for diffusion processes. The study of Brownian motion dates back to 1827 when Scottish botanist Robert Brown used a microscope to observe grains of pollen in water and found that the floating particles were behaving in an erratic fashion. Although Brown did not provide theory for this phenomenon, it became known as "Brownian motion". In 1905, Einstein provided physical theory of Brownian Motion [Einstein, 1905]. A detailed mathematical account of Brownian motion can be found in a number of texts (see for instance Kloeden and Platen [1992, Section 1.8], Øksendal [2007, Section 2.2], Karatzas and Shreve [1991, Chapter 2], Revuz and Yor [1991, Chapter 1]).

Definition 4.1.1. Brownian motion. A stochastic process $W := \{W_t : t \geq 0\}$ is called *(standard) Brownian Motion* (or a *Wiener Process*) if it satisfies the following properties:

Property 4.1.1. Initial Value. $W_0 = 0$.

Property 4.1.2. Independent increments. $W_t - W_s$ and $W_u - W_v$ are independent for $t > s \geq u > v$.

Property 4.1.3. Normally distributed increments. $W_t - W_s \sim \mathcal{N}_1(0, |t-s|)$ for $s < t$.

Property 4.1.4. Continuous paths. W_t is a *continuous function* of t (with probability 1).

In addition, Brownian motion satisfies several *self-similarity* properties which mean some transformations of the process leaves its properties invariant:

Property 4.1.5. Self-similarity Properties. Let W_t be Brownian motion, then the following processes are also Brownian motion processes:

1. **Scaling.** $B_t = \frac{1}{c} W_{c^2 t}$ where $c > 0$ is a constant.
2. **Symmetry.** $B_t = -W_t$.
3. **Increments.** $B_t = W_{t+s} - W_s$ where s is a fixed constant.
4. **Time Inversion.** $B_t = tW_{1/t}$ where $B_0 = 0$.

4.1.1 Simulating Brownian motion paths

Brownian motion sample paths are *continuous* and are infinite-dimensional random variables, and hence it is not possible to simulate and store entire sample path trajectories. However, we can simulate Brownian motion at any *finite* number of time points exactly by direct application of Definition 4.1.1. In particular, it is not possible to simulate $W \sim \mathbb{W}^x_{s,t}$ where $\mathbb{W}^x_{s,t}$ denotes the law of Brownian motion over time $[s,t]$ with $W_s = x$ (sometimes referred to as the *Wiener measure*). Instead, we can attempt to simulate a *finite* dimensional subset of the sample path (i.e. a *skeleton*) which we can use to characterise a Brownian motion sample path. In particular, since W has independent and normally distributed increments (due to Properties 4.1.2 and 4.1.3), the transition density of a Brownian motion process is known over any finite interval. We can use these properties to simulate a sample path of this process over any finite collection of time points, $\mathcal{T} := \{q_1, \ldots, q_n\}$, via Algorithm 4.1.1. In Figure 4.1, we present an illustration of Brownian motion sample paths which are simulated on a fine time mesh given by a collection of time points.

Algorithm 4.1.1 Brownian motion simulation at times $\mathcal{T} := \{q_1, \ldots, q_n\}$.

1. For i in 1 to n,
   (a) Simulate $W_{q_i} \sim \mathcal{N}_1(W_{q_{i-1}}, q_i - q_{i-1})$.
2. Return $\{W_{q_i}\}_{i=1}^n$.

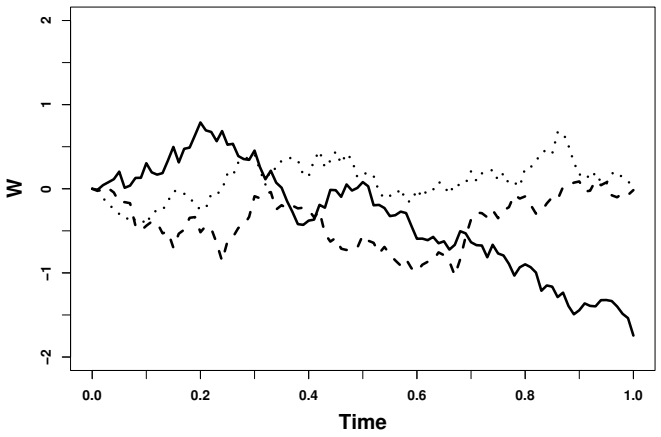


Figure 4.1: Brownian motion sample path trajectories, $W \sim \mathbb{W}_{0,1}^0$, simulated as per Algorithm 4.1.1 on a fine time mesh $\mathcal{T} = \{0, 0.01, \ldots, 0.98, 0.99, 1\}$.

4.1.2 Simulating Brownian bridges paths

Whilst Algorithm 4.1.1 allows us to simulate a Brownian motion sample path at a finite collection of points, it does not tell us how to further simulate the path between any two consecutive points. However, as a consequence of Properties 4.1.2, 4.1.3 and the Markov property, the law of the process between any two consecutive points are conditionally independent of the other simulated points.

Definition 4.1.2. Brownian Bridge. A *Brownian Bridge* is a Brownian motion conditioned to have a start point $(s, W_s = x)$ and end point $(t, W_t = y)$, and its law is denoted by $\mathbb{W}_{s,t}^{x,y}$.

Suppose we are interested in simulating the position of Brownian motion at an intermediate point $q \in (s, t)$ given the positions W_s and W_t at times s and t respectively. By the Markov property,

$$\begin{aligned}
\mathbb{P}(W_q = w | W_s = x, W_t = y) &\propto \mathbb{P}(W_t = y | W_q = w, W_s = x) \cdot \mathbb{P}(W_q = w | W_s = x) \\
&= \mathbb{P}(W_t = y | W_q = w) \cdot \mathbb{P}(W_q = w | W_s = x) \\
&\propto \exp\left\{-\frac{1}{2}\frac{(y-w)^2}{(t-q)}\right\} \cdot \exp\left\{-\frac{1}{2}\frac{(w-x)^2}{(q-s)}\right\} \\
&\sim \mathcal{N}_1\left(\frac{(t-q)x + (q-s)y}{(t-s)}, \frac{(t-q)(q-s)}{(t-s)}\right). \qquad (4.1)
\end{aligned}$$

As mentioned in Section 4.1.1, whilst it is not possible to simulate and store entire Brownian motion sample path trajectories, we can form *skeletons* of a Brownian motion sample path and can characterise the sample path using a only finite dimensional subset of the sample path. By using (4.1), we can now simulate Brownian motion at any desired time point even if the sample path has already been partially simulated via Algorithm 4.1.2.

Algorithm 4.1.2 Brownian bridge simulation at times $\{q_1, \ldots, q_n\}$ given the process at times $\{s, q_1, \ldots, q_n, t\}$.

1. Set $\mathcal{S} := \{(s, X_s), (q_i, X_{q_i})_{i=1}^n, (t, X_t)\}$.
2. For i in 1 to n,
   (a) Set $l := \sup\{\mathcal{S} : \mathcal{S} \leq q_i\}$ and $r := \inf\{\mathcal{S} : \mathcal{S} \geq q_i\}$.
   (b) Simulate $W_{q_i} \sim \mathcal{N}_1\left(W_l + \frac{(q_i - l)(W_r - W_l)}{r - l}, \frac{(r - q_i)(q_i - l)}{r - l}\right)$.
   (c) Set $\mathcal{S} := S \cup \{(q_i, W_{q_i})\}$.
3. Return $\{W_{q_i}\}_{i=1}^n$.

By using just the values of sample path at a finite collection of time points, we are able to bypass the computational and mathematical restrictions that come with the task of simulating entire Brownian motion sample path trajectories (i.e. required computation is finite and skeleton simulation at any required time point is exact). This idea of being able to characterise a sample path through a finite dimensional subset of the path is fundamental to simulating diffusion sample paths later in Section 4.3 and Section 4.4. We will provide a formal definition of a *skeleton* of a diffusion sample path later in Section 4.3 (see Definition 4.3.2), but for now this means being able to characterise a sample path by only using a finite dimensional subset of the path which can be simulated without error. Figure 4.2 illustrates several Brownian bridge sample path trajectories.

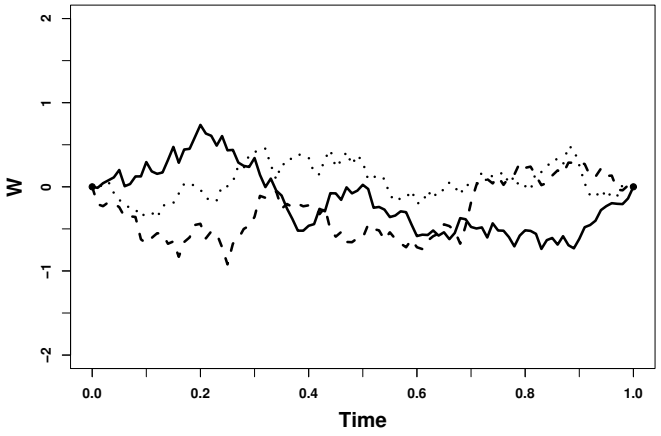


Figure 4.2: Brownian bridge sample path trajectories, $W \sim \mathbb{W}_{0,1}^{0,0}$, simulated as per Algorithm 4.1.2 on a fine time mesh $\mathcal{T} = \{0, 0.01, \ldots, 0.98, 0.99, 1\}$.

4.1.3 Simulating minimum and maximum points of a Brownian bridge

The joint distribution of the minimum value of a Brownian bridge, $\hat{m} := \inf\{W_q : q \in [s, t]\}$, and the time at which is attained, $\tau := \sup\{q \in [s, t] : W_q = \hat{m}\}$ is given in [Karatzas and Shreve, 1991]:

$$\mathbb{P}(\hat{m} \in \mathrm{d}w, \tau \in \mathrm{d}q | W_s = x, W_t = y)$$

$$\propto \frac{(w-x)(w-y)}{\sqrt{(t-q)^3(q-s)^3}} \exp\left\{-\frac{(w-x)^2}{2(q-s)} - \frac{(w-y)^2}{2(t-q)}\right\} \, \mathrm{d}w \, \mathrm{d}q. \tag{4.2}$$

Beskos et al. [2006a, Section 3.1] showed that it is possible to simulate $(\tau, \hat{\mu})$ from (4.2) by first simulating $u_1, u_2 \sim \mathcal{U}[0,1]$ and setting

$$\hat{m} = x - \frac{1}{2}\left[\sqrt{(y-x)^2 - 2(t-s)\log(u_1)} - (y-x)\right], \tag{4.3}$$

and

$$V = \begin{cases} \xi_1, \text{ where } \xi_1 \sim \mathrm{IG}\left(\frac{y-\hat{m}}{x-\hat{m}}, \frac{(y-\hat{m})^2}{t-s}\right) \text{ if } u_2 < \frac{x-\hat{m}}{x+y-2\hat{m}}, \\ \frac{1}{\xi_2}, \text{ where } \xi_2 \sim \mathrm{IG}\left(\frac{x-\hat{m}}{y-\hat{m}}, \frac{(x-\hat{m})^2}{t-s}\right) \text{ if } u_2 \geq \frac{x-\hat{m}}{x+y-2\hat{m}}, \end{cases} \tag{4.4}$$

where $\mathrm{IG}(\mu, \lambda)$ denotes the inverse Gaussian distribution with density

$$\mathrm{IG}(u; \mu, \lambda) = \sqrt{\frac{\lambda}{2\pi u^3}} \exp\left\{-\frac{\lambda(u-\mu)^2}{2\mu^2 u}\right\}, \qquad u > 0. \tag{4.5}$$

Furthermore, it is also possible to simulate the Brownian bridge sample path minimum conditional on it occuring within a particular interval $\hat{m} \in [a_1, a_2]$ where $a_1 < a_2 \leq (x \wedge y)$. For instance, we can simply achieve this via rejection sampling (see Section 2.2) by simulating sample path minima until one falls within the interval $[a_1, a_2]$ [Beskos et al., 2006a]. However, as noted in [Pollock et al., 2016], it is more computationally efficient to simulate the minimum by inversion sampling (see Section 2.1). In this setting, we modify how the uniform random variable u_1 is simulated:

$$u_1 \sim \mathcal{U}[M(a_1), M(a_2)], \text{ where } M(a) := \exp\left\{-\frac{2(a-x)(a-y)}{(t-s)}\right\}. \tag{4.6}$$

Algorithm 4.1.3 provides a summary of the above arguments.

Algorithm 4.1.3 Brownian bridge simulation at its minimum point (constrained to the interval $[a_1, a_2]$, where $a_1 < a_2 \leq (x \wedge y)$ and conditional on $W_s = x$ and $W_t = y$) [Pollock et al., 2016, Algorithm 12].

---

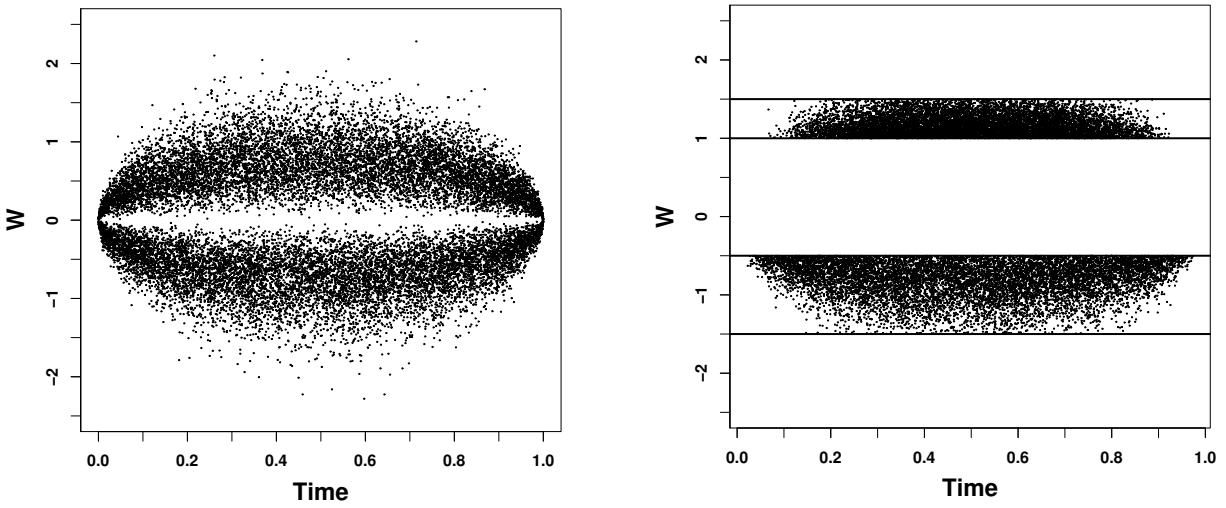
1. Simulate $u_1 \sim U[M(a_1), M(a_2)]$, where $M(a) := \exp\left\{-\frac{2(a-x)(a-y)}{(t-s)}\right\}$ and $u_2 \sim U[0,1]$.
2. Set $\hat{m} = x - \frac{1}{2}\left[\sqrt{(y-x)^2 - 2(t-s)\log(u_1)} - (y-x)\right]$.
3. Set $V = \begin{cases} \xi_1, \text{ where } \xi_1 \sim \mathrm{IG}\left(\frac{y-\hat{m}}{x-\hat{m}}, \frac{(y-\hat{m})^2}{t-s}\right) \text{ if } u_2 < \frac{x-\hat{m}}{x+y-2\hat{m}}, \\ \frac{1}{\xi_2}, \text{ where } \xi_2 \sim \mathrm{IG}\left(\frac{x-\hat{m}}{y-\hat{m}}, \frac{(x-\hat{m})^2}{t-s}\right) \text{ if } u_2 \geq \frac{x-\hat{m}}{x+y-2\hat{m}}. \end{cases}$
4. Set $\tau := \frac{sV+t}{1+V}$.
5. Return $(\tau, \hat{m})$.

We can similarly simulate the maximum of a Brownian bridge sample path, $(\tau, \check{m})$ where $\check{m} := \sup\{W_q : q \in [s,t]\}$, by a reflection argument. In particular, by the self-similarity properties of

Brownian motion (Property 4.1.5), we know that if W is Brownian motion then so is $B_t = -W_t$. As a consequence, if we were to reflect the Brownian bridge, $W' \sim \mathbb{W}_{s,t}^{-x,-y}$ and simulate the minimum point of this reflected Brownian bridge using Algorithm 4.1.3, then the reflection of the minimum point is required simulated maximum of $W \sim \mathbb{W}_{s,t}^{x,y}$. An algorithm to simulate the maximum point of a Brownian bridge sample path trajectory is given in Algorithm 4.1.4. Figure 4.3 provides an illustration of simulated minima and maxima of a Brownian bridge sample path.

Algorithm 4.1.4 Brownian bridge simulation at its maximum point (constrained to the interval $[a_1, a_2]$, where $(x \wedge y) \leq a_1 < a_2$ and conditional on $W_s = x$ and $W_t = y$).

1. Simulate a minimum point $(\hat{m}, \tau)$ in the interval $[-a_2, -a_1]$ conditional on $W_s = -x$ and $W_t = -y$, as per Algorithm 4.1.3.
2. Set $\check{m} = -\hat{m}$.
3. Return $(\tau, \check{m})$.



(a) Minimum and maximum point simulation of $W \sim \mathbb{W}_{0,0}^{0,1}$ without restriction.

(b) Minimum point simulation of $W \sim \mathbb{W}_{0,1}^{0,0} | \hat{m} \in [-1.5, -0.5]$ and maximum point simulation of $W \sim \mathbb{W}_{0,1}^{0,0} | \check{m} \in [1.0, 1.5]$.

Figure 4.3: An illustration of 10000 minimum and maximum points of Brownian bridge sample path trajectories simulated as per Algorithm 4.1.3 and Algorithm 4.1.4.

4.1.4 Simulating Bessel bridges

It transpires later in this thesis that it is necessary to simulate a value of a Brownian bridge path at some time q conditional on a minimum, $(\tau, \hat{m})$, or maximum, $(\tau, \check{m})$, point of the path. The law of the remainder of the trajectory of a Brownian bridge path given its minimum or maximum point is called a *Bessel Bridge*. Asmussen et al. [1995, Proposition 2] proved that a Bessel bridge can be constructed using a three-dimensional Brownian bridge of unit length conditioned to start and end at zero.

First consider the case that we are interested in simulating the Bessel bridge sample path at some intermediate time $q \in (\tau, t)$. We are effectively simulating the value of a Brownian bridge path at time q, conditioned to some start point $(\tau, \hat{m})$, end point (t, y) and for the path to remain above $\hat{m}$ (since it is the minimum of the sample path). By re-scaling location and time and using the self-similarity properties of Brownian motion (Property 4.1.5), this is equivalent to simulating the value of the unit length Brownian bridge sample path at time $q' := \frac{q-\tau}{t-\tau}$ with start point $(0, 0)$ and end point $(1, y' = y - \hat{m})$ conditioned to remain above 0. The simulated value must then also be appropriately re-scaled again, so $w = W_{q'} + \hat{m}$. Beskos et al. [2006a, Theorem 2] used this as a means of simulating intermediate points of a Bessel bridge by first simulating three independent realisations of a Brownian bridge of unit length conditioned on the start and end points of zero at time q', denoted $\{b_1, b_2, b_3\}$. We then re-scale the simulated point to obtain

$$w = \hat{m} + \sqrt{(t-\tau)\left[\frac{(y-\hat{m})(q-\tau)}{(t-\tau)(t-\tau)^{\frac{1}{2}}} + b_1\right]^2 + (t-\tau)b_2^2 + (t-\tau)b_3^2}. \tag{4.7}$$

If we are interested in simulating the sample path at time $q \in (s, \tau)$, then we can simply reverse time and apply similar arguments: consider a Brownian bridge path at a time q with start point (s, x) and end at $(\tau, \hat{m})$ conditioned to remain above $\hat{m}$. This is equivalent to simulating at time $q' = \frac{q-s}{\tau-s}$ with start point at $(0, 0)$ and to end at $(1, \hat{m} - x)$ and conditioned to remain above 0. Similarly, we must re-scale the simulated value, $w = W_{q'} + x$.

$$w = \hat{m} + \sqrt{(\tau-s)\left[\frac{(x-\hat{m})(\tau-q)}{(\tau-s)(\tau-s)^{\frac{1}{2}}} + b_1\right]^2 + (\tau-s)b_2^2 + (\tau-s)b_3^2} \tag{4.8}$$

To summarise, the algorithm to simulate a Bessel Bridge at some time $q \in (s, t)$ given the minimum point is given in Algorithm 4.1.5. We can use this algorithm repeatedly to obtain a skeleton to characterise the sample path.

Algorithm 4.1.5 (Minimum) Bessel bridge simulation at time $q \in (s, t)$ conditioned on $W_s = x, W_t = y, W_\tau = \hat{m}$ [Asmussen et al., 1995; Beskos et al., 2006a], [Pollock et al., 2016, Algorithm 13].

---

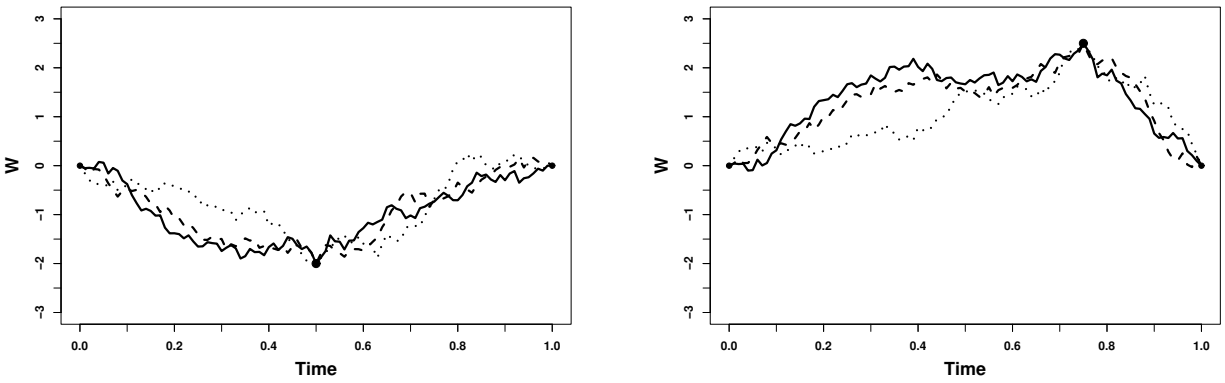
1. If $q < \tau$, then $r = s$, else $r = t$.
2. Simulate $b_1, b_2, b_3 \sim \mathcal{N}_1\left(0, \frac{|\tau-q|\cdot|q-r|}{(\tau-r)^2}\right)$.
3. Return $W_q := \hat{m} + \sqrt{|\tau - r|} \cdot \sqrt{\left(\frac{(W_r-\hat{m})\cdot|\tau-q|}{|\tau-r|^{3/2}} + b_1\right)^2 + b_2^2 + b_3^2}$.

In order to simulate a Bessel bridge given a maximum point $\check{m}$, by the Property 4.1.5, we can note that the maximum of $W \sim \mathbb{W}_{s,t}^{x,y}$ is simply minus the minimum of $-W \sim \mathbb{W}_{s,t}^{-x,-y}$, and hence we can just apply a reflection argument. In particular, we can apply Algorithm 4.1.5 to simulate an intermediate point of a reflected Bessel bridge conditional on $W'_s = -x$, $W'_t = -y$ with minimum

$\hat{m}' = -\check{m}$ at time τ and then reflect the resulting simulated point along the x-axis. An algorithm to simulate a Bessel bridge at an intermediate time $q \in (s,t)$ given a sample path maximum $(\check{m}, \tau)$ is given in Algorithm 4.1.6. Figure 4.4 illustrates several Bessel bridge sample path trajectories.

Algorithm 4.1.6 (Maximum) Bessel bridge simulation at time $q \in (s,t)$ conditioned on $W_s = x, W_t = y, W_\tau = \check{m}$.

1. Simulate intermediate point, W'_q, of a Bessel bridge at time q given $W'_s = -x$, $W'_t = -y$ and minimum point $W'_\tau = -\check{m}$.
2. Set $W_q := -W'_q$.
3. Return W_q.



(a) $W \sim \mathbb{W}_{0,1}^{0,0}|(\tau = 0.5, \hat{m} = -2)$.

(b) $W \sim \mathbb{W}_{0,1}^{0,0}|(\tau = 0.75, \check{m} = 2.5)$.

Figure 4.4: Bessel bridge sample path trajectories simulated as per Algorithm 4.1.5 and Algorithm 4.1.6 on a fine time mesh $\mathcal{T} = \{0, 0.01, \ldots, 0.98, 0.99, 1\}$.

4.2 Brownian bridge path-space constructions

In the subsequent methodology discussed in this thesis, we require the ability to simulate *layered Brownian bridge* sample path skeletons, which are Brownian bridge sample paths with the restriction that they are constrained within a particular interval. In this section, we review methodology for simulating quantities relating to various Brownian bridge path-space constructions. In Section 4.2.1, we discuss methods for simulating the probability that Brownian and Bessel bridge sample paths are constrained within particular intervals. Next, in Section 4.2.2, we introduce algorithms for simulating layered Brownian bridge sample path (in particular, simulating finite dimensional sample paths, or *skeletons*, of layered Brownian bridges). These algorithms are detailed in a number of texts including (but not limited to) Beskos et al. [2008], Beskos et al. [2012], Wang and Pötzelberger [1997], Pötzelberger and Wang [2001], Chen and Huang [2013], Giesecke and Smelov [2013], Pollock [2013, Chapter 6] and Pollock et al. [2016, Section 6 and 7]. Here, we follow the notation of Pollock et al. [2016, Section 6 and 7] to introduce this methodology.

4.2.1 Simulating Brownian bridge path-space probabilties

The task of simulating events with the probability that a Brownian or Bessel bridge sample path is constrained within a particular interval is difficult since these probabilities typically can only be represented as the limit of alternating Cauchy sequences (see Section 2.5) [Wang and Pötzelberger, 1997; Pollock et al., 2016]. We cannot naively approximate these probabilities by truncating the alternating Cauchy sequences since this ultimately results in bias. However, Beskos et al. [2008] noted that we can use the Cauchy sequence representations and employ a retrospective Bernoulli sampling approach (see Section 2.5) to simulate unbiasedly events of these unknown probabilities. In this section, we outline some results from Pötzelberger and Wang [2001]; Beskos et al. [2008]; Pollock [2013]; Pollock et al. [2016] and methods to (unbiasedly) simulate events with probabilities that a Brownian or Bessel bridge sample path is contained within a particular interval. Proofs are not reproduced here but can be found in Pollock [2013, Chapter 6] and references therein.

We begin by considering the probability that $\{W_u : s \leq u \leq t\} \in [l, v]$ for some sample path $W \sim \mathbb{W}_{s,t}^{x,y}$. Following the notation of Pollock et al. [2016], we slightly abuse notation and write $\{W \in [l, v]\}$ to mean $\{W_u : s \leq u \leq t\} \subseteq [l, v]$. We denote the probability that a Brownian bridge sample path $W \sim \mathbb{W}_{s,t}^{x,y}$ remains in the interval $[l, v]$ as $\gamma_{s,t}^{l,v}(x, y)$.

Theorem 4.2.1. ([Pötzelberger and Wang, 2001, Theorem 3], [Pollock et al., 2016, Theorem 3]). *The probability that a Brownian bridge sample path $W \sim \mathbb{W}_{s,t}^{x,y}$ remains in the interval $[l, v]$, i.e. $W_u \in [l, v]$ for all $u \in [s, t]$, can be represented by the following infinite series*

$$\begin{aligned}\gamma_{s,t}^{l,v}(x, y) &\coloneqq \mathbb{P}\left(W \in [l, v]\right) \\ &= 1 - \sum_{j=1}^{\infty} \left\{ \varsigma_{s,t}^{l,v}(j; x, y) - \varphi_{s,t}^{l,v}(j; x, y) \right\}, \end{aligned} \tag{4.9}$$

where

$$\varsigma_{s,t}^{l,v}(j; x, y) \coloneqq \bar{\varsigma}_{s,t}^{l,v}(j; x, y) + \bar{\varsigma}_{s,t}^{-l,-v}(j; -x, -y), \tag{4.10}$$

$$\varphi_{s,t}^{l,v}(j; x, y) \coloneqq \bar{\varphi}_{s,t}^{l,v}(j; x, y) + \bar{\varphi}_{s,t}^{-l,-v}(j; -x, -y), \tag{4.11}$$

and

$$\bar{\varsigma}_{s,t}^{l,v}(j; x, y) \coloneqq \exp\left\{ -\frac{2}{t-s} \left(|v-l|j + (l \wedge v) - x\right) \cdot \left(|v-l|j + (l \wedge v) - y\right) \right\}, \tag{4.12}$$

$$\bar{\varphi}_{s,t}^{l,v}(j; x, y) \coloneqq \exp\left\{ -\frac{2j}{t-s} \left(|v-l|^2 j + |v-l|(x-y) \right) \right\}. \tag{4.13}$$

Corollary 4.2.1. ([Beskos et al., 2008, Proposition 2], [Pollock et al., 2016, Corollary 3]) *Events of probability $\gamma_{s,t}^{l,v}(x, y)$ can be simulated by retrospective Bernoulli sampling with the following al-*

ternating Cauchy sequence,

$$S^{\gamma}_{2k} \coloneqq 1 - \sum_{j=1}^{k} \left\{ \varsigma^{l,v}_{s,t}(j;x,y) - \varphi^{l,v}_{s,t}(j;x,y) \right\}, \tag{4.14}$$

$$S^{\gamma}_{2k+1} \coloneqq S_{2k} - \varsigma^{l,v}_{s,t}(k+1;x,y). \tag{4.15}$$

As a consequence, events of probability $\gamma^{l,v}_{s,t}(x,y)$ can be simulated unbiasedly by retrospective Bernoulli sampling as per Algorithm 4.2.1.

Algorithm 4.2.1 Simulating an event of probability $\gamma^{l,v}_{s,t}(x,y)$ [Beskos et al., 2008], [Pollock et al., 2016].

1. Simulate $u \sim U[0,1]$ and set $k = 1$.
2. While $u \in (S^{\gamma}_{2k+1}, S^{\gamma}_{2k})$, where $S^{\gamma}_{2k} \coloneqq 1 - \sum_{j=1}^{k}\{\varsigma^{l,v}_{s,t}(j;x,y) - \varphi^{l,v}_{s,t}(j;x,y)\}$ and $S^{\gamma}_{2k+1} \coloneqq S^{\gamma}_{2k} - \varsigma^{l,v}_{s,t}(k+1;x,y)$, then $k = k+1$.
3. If $u \leq S^{\gamma}_{2k+1}$, then $u < p$ so return to 1, or return 0.

We can consider simulating events of probability that a Bessel bridge sample path with known minimum (or maximum) stays within a particular interval. We begin by focusing on Bessel bridges with known sample path *minimum* $\hat{m}$ and note that we can consider the probability that a Bessel bridge sample path given the sample path *maximum* $\check{m}$ can be computed by a reflection argument. Let $\delta^{\hat{m},v}_{s,t}(x,y)$ to denote the probability that a Bessel bridge sample path $W \sim \mathbb{W}^{x,y}_{s,t}|\hat{m}$ with minimum $\hat{m}$ remains in the interval $[\hat{m}, v]$. We consider the two possible cases: (i) neither of the end points are equal to the minimum; and (ii) either one of the end points of the sample path is equal to the sample path minimum,

$$\delta^{\hat{m},v}_{s,t}(1;x,y) \coloneqq \mathbb{P}\left(W \in [\hat{m},v] | W \geq \hat{m}, (x \wedge y) > \hat{m}\right), \tag{4.16}$$

$$\delta^{\hat{m},v}_{s,t}(2;x,y) \coloneqq \mathbb{P}\left(W \in [\hat{m},v] | W \geq \hat{m}, (x \wedge y) = \hat{m}\right), \tag{4.17}$$

and note that $\delta^{\hat{m},v}_{s,t}(x,y) = \mathbb{1}_{\{(x\wedge y)>\hat{m}\}} \cdot \delta^{\hat{m},v}_{s,t}(1;x,y) + \mathbb{1}_{\{\hat{m}=(x\wedge y)\}} \cdot \delta^{\hat{m},v}_{s,t}(2;x,y)$. Firstly, consider case (i) where neither end points of the sample path attains the Bessel bridge minimum.

Theorem 4.2.2. [Beskos et al., 2008, Proposition 3], [Pollock et al., 2016, Theorem 4]. *The probability that a Bessel bridge sample path $W \sim \mathbb{W}^{x,y}_{s,t}|\hat{m}$ with minimum $\hat{m} < (x \wedge y)$ remains in the interval $[\hat{m}, v]$, i.e. $W_u \in [\hat{m}, v]$, for all $u \in [s,t]$, can be represented as the following infinite series*

$$\begin{aligned} \delta^{\hat{m},v}_{s,t}(1;x,y) &\coloneqq \mathbb{P}(W \in [\hat{m},v] | W \geq \hat{m}, (x \wedge y) > \hat{m}) \\ &= \frac{\gamma^{\hat{m},v}_{s,t}(x,y)}{1 - \exp\left\{-2\frac{(x-\hat{m})\cdot(y-\hat{m})}{t-s}\right\}}. \end{aligned} \tag{4.18}$$

Corollary 4.2.2. [Beskos et al., 2008, Proposition 3], [Pollock et al., 2016, Corollary 4]. *Events of probability* $\delta_{s,t}^{\hat{m},v}(1;x,y)$ *can be represented as the limit as* $k \to \infty$ *of the following alternating Cauchy sequence,*

$$S_k^{\delta,1} \coloneqq \frac{S_k^{\gamma}}{1-\exp\left\{-2\frac{(x-\hat{m})(y-\hat{m})}{t-s}\right\}}, \tag{4.19}$$

i.e. $\lim_{k\to\infty} S_k^{\delta,1} = \delta_{s,t}^{\hat{m},v}(1;x,y)$.

Since S_k^{γ} is an alternating Cauchy sequence and $\left(1-\exp\left\{-2\frac{(x-\hat{m})(y-\hat{m})}{t-s}\right\}\right)$ is just a constant, then $S_k^{\delta,1}$ is a linear transformation of an alternating Cauchy sequence and therefore $S_k^{\delta,1}$ is an alternating Cauchy sequence itself. This can be used to unbiasedly simulate events of probability $\delta_{s,t}^{\hat{m},v}(1;x,y)$. Now consider case (ii) where either one of the end points attains the Bessel bridge minimum.

Theorem 4.2.3. [Beskos et al., 2008, Proposition 3], [Pollock et al., 2016, Theorem 5]. *The probability that a Bessel bridge sample path* $W \sim \mathbb{W}_{s,t}^{x,y}|\hat{m}$ *with minimum* $\hat{m} = x < y$ *remains in the interval* $[\hat{m}, v]$, *i.e.* $W_u \in [\hat{m}, v]$, *for all* $u \in [s,t]$, *can be represented as the following infinite series*

$$\begin{aligned}\delta_{s,t}^{\hat{m},v}(2;x,y) &\coloneqq \mathbb{P}(W \in [\hat{m},v] | W \geq \hat{m}, (x \wedge y) = \hat{m}) \\ &= 1 - \frac{1}{(y-\hat{m})}\sum_{j=1}^{\infty}\left\{\psi_{s,t}^{\hat{m},v}(j;y) - \chi_{s,t}^{\hat{m},v}(j;y)\right\},\end{aligned} \tag{4.20}$$

where we denote

$$\psi_{s,t}^{\hat{m},v}(j;y) \coloneqq (2|v-\hat{m}|j - (y-\hat{m}))\exp\left\{-\frac{2|v-\hat{m}|j}{t-s}(|v-\hat{m}|j - (y-\hat{m}))\right\}, \tag{4.21}$$

$$\chi_{s,t}^{\hat{m},v}(j;y) \coloneqq (2|v-\hat{m}|j + (y-\hat{m}))\exp\left\{-\frac{2|v-\hat{m}|j}{t-s}(|v-\hat{m}|j + (y-\hat{m}))\right\}. \tag{4.22}$$

Corollary 4.2.3. [Pollock et al., 2016, Corollary 5]. *After inclusion of the first* $\hat{k} \coloneqq \frac{\sqrt{(t-s)+|v-\hat{m}|^2}}{2|v-\hat{m}|}$ *terms,* $\delta_{s,t}^{\hat{m},y}(2;x,y)$ *can be represented as the limit as* $k \to \infty$ *of the following alternating Cauchy sequence (where* $k \in \mathbb{N}$ *such that* $k \geq \hat{k}$*),*

$$S_{2k}^{\delta,2} \coloneqq 1 - \frac{1}{|x-y|}\sum_{j=1}^{k}\left\{\psi_{s,t}^{\hat{m},v}(j;(x\vee y)) - \chi_{s,t}^{\hat{m},v}(j;(x\vee y))\right\}, \tag{4.23}$$

$$S_{2k+1}^{\delta,2} \coloneqq S_{2k}^{\delta,2} - \frac{1}{|x-y|}\psi_{s,t}^{\hat{m},v}(k+1;(x\vee y)). \tag{4.24}$$

Since we can unbiasedly simulate events of probability $\delta_{s,t}^{\hat{m},v}(1;x,y)$ and $\delta_{s,t}^{\hat{m},v}(2;x,y)$, then events of probability $\delta_{s,t}^{\hat{m},v}(x,y)$ can be simulated as per Algorithm 4.2.2. Further, events of probability $\delta_{s,t}^{l,\hat{m}}(x,y)$ can be simulated unbiasedly by applying a reflection argument as per Algorithm 4.2.3.

Algorithm 4.2.2 Simulating an event of probability $\delta_{s,t}^{\hat{m},v}(x,y)$ [Pollock et al., 2016].

1. Simulate $u \sim \mathcal{U}[0,1]$.
2. If $(x \wedge y) > \hat{m}$:
   (a) Set $k = 1$.
   (b) While $u \in (S_{2k+1}^{\delta,1}, S_{2k}^{\delta,1})$, where $S_k^{\delta,1}$ is defined in (4.19), then $k = k+1$.
   (c) If $u \leq S_{2k+1}^{\delta,1}$, then $u < p$ so return to 1, else $u > p$ so return 0.
3. If $(x \wedge y) = \hat{m}$:
   (a) Set $k = \frac{\sqrt{(t-s)+|v-\hat{m}|^2}}{2|v-\hat{m}|}$.
   (b) While $u \in (S_{2k+1}^{\delta,2}, S_{2k}^{\delta,2})$, where $S_{2k}^{\delta,2}$ and $S_{2k+1}^{\delta,2}$ are defined in (4.23) and (4.24) respectively, then $k = k+1$.
   (c) If $u \leq S_{2k+1}^{\delta,2}$, then $u < p$ so return 1, else $u > p$ so return 0.

Algorithm 4.2.3 Simulating an event of probability $\delta_{s,t}^{l,\check{m}}(x,y)$.

1. Set $x' = -x$, $y' = -y$, $\hat{m}' = -\check{m}$ and $v' = -l$.
2. Simulate event probability $\delta_{s,t}^{\hat{m}',v'}(x',y')$ as per Algorithm 4.2.2.

4.2.2 Layered Brownian bridge constructions

We now focus on how to construct and simulate finite dimensional skeletons of *layered Brownian bridges* which will be instrumental in the development of the methodology in the later sections. Although there are several different ways to construct *layer information* of a Brownian bridge sample path, we focus on the *Bessel layer approach* [Beskos et al., 2008], [Pollock, 2013, Section 6.2.1], [Pollock et al., 2016, Section 7.1] in this thesis. Alternative approaches such as the *localised approach* [Chen and Huang, 2013; Giesecke and Smelov, 2013; Pollock et al., 2016] or the *intermediate layer approach* [Pollock et al., 2016, Section 8] are not discussed here.

The key idea of the *Bessel approach* to construct layer information for Brownian bridge sample paths is that finite dimensional subsets of Brownian bridge sample paths can be simulated jointly with information regarding the interval in which its constrained by partitioning the path-space with an arbitrary increasing sequence $\{a_i\}_{i\geq 0}, a_0 = 0$ which radiates outwards from the interval $[(x \wedge y), (x \vee y)]$. Specifically, the *ith Bessel layer* is defined as

$$\mathcal{I}_i = [(x \wedge y) - a_i, (x \vee y) + a_i]\,. \tag{4.25}$$

Let $S_k^{\gamma}(s,t,x,y,l,v)$ denote the alternating Cauchy sequence whose limit as $k \to \infty$ is $\gamma_{s,t}^{l,v}(x,y)$, then the smallest Bessel layer, $\mathcal{I} = l$, in which a particular Brownian bridge sample path is constrained can be simulated unbiasedly by retrospective Bernoulli sampling as per Algorithm 4.2.4. Intuitively, in Algorithm 4.2.4, we keep extending the Bessel layer and pushing it outwards until an event of probability $\gamma_{s,t}^{l,v}(x,y)$ occurs (which we simulate through retrospective Bernoulli sampling). In Step 3, we are essentially checking if this event occurs and return the current layer if it does, or extend the Bessel layer if it does not. An example simulation of a Bessel layer is provided in Figure 4.5.

Algorithm 4.2.4 Brownian bridge Bessel layer simulation [Pollock et al., 2016, Algorithm 14].

1. Simulate $u \sim U[0,1]$ and set $l = 1, k = 0$.
2. While $u \in \left(S^{\gamma}_{2k+1}(s,t,x,y,(x \wedge y) - a_l, (x \vee y) + a_l), S^{\gamma}_{2k}(s,t,x,y,(x \wedge y) - a_l, (x \vee y) + a_l)\right)$, then $k = k + 1$.
3. If $u \geq S^{\gamma}_{2k}$, set $l = l + 1$ and return to Step 2, else set $\mathcal{I} = l$ and end.

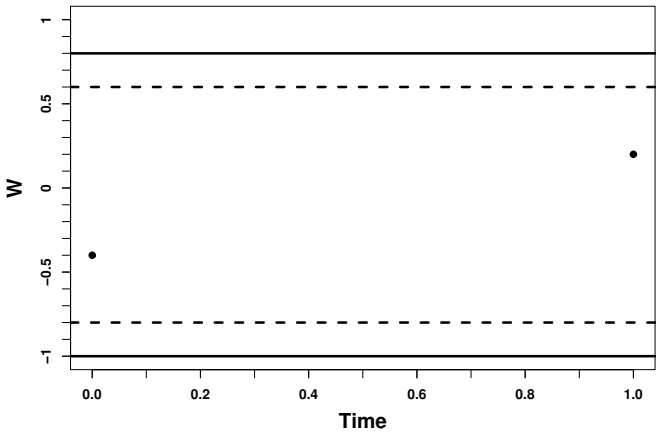


Figure 4.5: An illustration of a simulated Bessel layer for a path $W \sim \mathbb{W}^{0,1}_{-0.4,0.2}$, where $\{a_i\}_{i \geq 0} = \{0, 0.2, 0.4, \dots\}$. The solid lines denotes the interval which constrains the path entirely. The dashed lines indicate the interval which does not constrain the path, as the path at some point will fall outside these dotted lines.

After simulating the Bessel layer, we require a method of simulating intermediate points from the Brownian bridge sample path that is restricted to remain within the Bessel layer simulated. Let D_l be the set of sample paths which are contained in the lth Bessel layer, we have

$$D_l = L_l \cup U_l, \tag{4.26}$$

where

$$L_l \coloneqq \{W : \hat{m}_{s,t} \in [(x \wedge y) - a_l, (x \wedge y) - a_{l-1})\} \bigcap \{W : \check{m}_{s,t} \in [(x \vee y), (x \vee y) + a_l]\}, \tag{4.27}$$

$$U_l \coloneqq \{W : \hat{m}_{s,t} \in [(x \wedge y) - a_l, (x \wedge y)]\} \bigcap \{W : \check{m}_{s,t} \in ((x \vee y) + a_{l-1}, (x \vee y) + a_l]\}. \tag{4.28}$$

The derivation of this result can be found in Pollock [2013, Section 6.2.1]. Intuitively, L_l is the set of sample paths where the minimum is on the 'edge' of the Bessel layer, and U_l is the set of sample paths where the maximum is on the 'edge' of the Bessel layer. Directly simulating intermediate points from a sample path with law D_l, denoted $\mathbb{D}_l$, is not possible. However, if we denote

$$\hat{M}_l = \{W : \hat{m}_{s,t} \in [(x \wedge y) - a_l, (x \wedge y) - a_{l-1}]\},$$
$$\check{M}_l = \{W : \check{m}_{s,t} \in [(x \vee y) + a_{l-1}, (x \vee y) + a_l]\},$$

we can propose sample paths from the mixture measure $\mathbb{B}_l \coloneqq \frac{\hat{\mathbb{M}}_l}{2} + \frac{\check{\mathbb{M}}_l}{2}$, where $\hat{\mathbb{M}}_l$ and $\check{\mathbb{M}}_l$ are the law induced by the restricting $\mathbb{W}_{s,t}^{x,y}$ to $\hat{M}_l$ and $\check{M}_l$, respectively, and accept them with probability given by the Radon-Nikodým derivative of $\mathbb{D}_l$ with respect to $\mathbb{B}_l$ given by Beskos et al. [2008]:

$$\frac{\mathrm{d}\mathbb{D}_l}{\mathrm{d}\mathbb{B}_l}(x) \propto \frac{\mathbb{1}(W \in D_l)}{1 + \mathbb{1}(W \in (\hat{M}_l \cap \check{M}_l))}. \tag{4.29}$$

Therefore, sample paths can be drawn from $\mathbb{D}_l$ by proposing from $\mathbb{B}_l \coloneqq \frac{\hat{\mathbb{M}}_l}{2} + \frac{\check{\mathbb{M}}_l}{2}$ and then accepting the path with probability (4.29). In particular, with probability $1/2$, we sample from $\hat{\mathbb{M}}_l$ (or $\check{\mathbb{M}}_l$) and accept with probability 1 if the sample path is contained within the $(l-1)$th Bessel layer since,

$$\frac{\mathrm{d}\mathbb{D}_l}{\mathrm{d}\mathbb{B}_l}(x) \propto \frac{\mathbb{1}(W \in D_l)}{1 + \mathbb{1}(W \in (\hat{M}_l \cap \check{M}_l))} = \frac{1}{1+0} = 1, \tag{4.30}$$

or we accept the sample path with probability $1/2$ if the maximum is contained between the $(l-1)$th and lth Bessel layer since

$$\frac{\mathrm{d}\mathbb{D}_l}{\mathrm{d}\mathbb{B}_l}(x) \propto \frac{\mathbb{1}(W \in D_l)}{1 + \mathbb{1}(W \in (\hat{M}_l \cap \check{M}_l))} = \frac{1}{1+1} = \frac{1}{2}, \tag{4.31}$$

and 0 otherwise and reject the sample path.

As noted by Beskos et al. [2008] and Pollock et al. [2016], we are not able to directly evaluate (4.29). However, we are able to obtain an unbiased estimate using results stated in Section 4.2.1. In particular, with probability $1/2$, we simulate the sample path minimum $W_\tau \coloneqq \hat{m}_{s,t}$ as per Algorithm 4.1.3. We can then simulate any required intermediate points $\xi_1, \ldots, \xi_\kappa$ from the Bessel bridge conditional on the minimum as per Algorithm 4.1.5. Let $\chi_1, \ldots, \chi_{\kappa+3}$ be the order statistics of $\{\xi_1, \ldots, \xi_\kappa, s, \tau, t\}$, then

$$\begin{aligned}\mathbb{P}_{\hat{\mathbb{M}}_l}(X \in D_l) &= \mathbb{P}\left(X \in [(x \wedge y) - a_l, (x \vee y) + a_l) \middle| X_{\chi_1}, \ldots X_{\chi_{\kappa+3}}\right) \\ &= \prod_{i=1}^{\kappa+2} \delta_{\chi_i,\chi_{i+1}}^{\hat{m},(x\vee y)+a_l}\left(X_{\chi_i}, X_{\chi_{i+1}}\right),\end{aligned} \tag{4.32}$$

and

$$\mathbb{P}_{\hat{\mathbb{M}}_l}\left(X \in (\hat{M}_l \cap \check{M}_l)\right) = \mathbb{P}_{\hat{\mathbb{M}}_l}(X \in D_l) - \prod_{i=1}^{\kappa+2} \delta_{\chi_i,\chi_{i+1}}^{\hat{m},(x\vee y)+a_{l-1}}\left(X_{\chi_i}, X_{\chi_{i+1}}\right). \tag{4.33}$$

Since these probabilities in (4.32) and (4.33) can be represented as a linear function of δ probabilities (recalling from Section 4.2, $\delta_{s,t}^{\hat{m},v}(x,y)$ denotes the probability that a Bessel bridge with minimum $\hat{m} < (x \wedge y)$ remains in the interval $[\hat{m}, v]$), then events of this probability can be simulated unbiasedly by retrospective Bernoulli sampling (see Section 2.5). In the case that we sample $\check{M}_l$, we can simply amend the arguments above. The algorithm for simulating a layered Brownian bridge conditional on a Bessel layer is given in Algorithm 4.2.5.

Algorithm 4.2.5 Layered Brownian bridge simulation (Bessel approach): Sampling X at times $\xi_1, \ldots, \xi_\kappa$ [Pollock et al., 2016, Algorithm 15].

1. Simulate $u_1, u_2, \sim U[0,1]$ and set $j = k = 0$.
2. Simulate auxiliary information, conditional on Bessel bridge, $I = l$,
   (a) If $u_1 < 1/2$, simulate minimum point $(\tau, \hat{m}_{s,t})$ and set: $l_1 = l_2 = \hat{m}_{s,t}$; $v_1 = (x \vee y) + a_{l-1}$; $v_2 = (x \vee y) + a_l$.
   (b) If $u_1 > 1/2$, simulate maximum point $(\tau, \check{m}_{s,t})$ and set $l_1 = (x \wedge y) - a_{l-1}$; $l_2 = (x \wedge y) - a_l$; $v_1 = v_2 = \check{m}_{s,t}$.
3. Simulate intermediate times $X_{\xi_1}, \ldots, X_{\xi_\kappa}$ from a Bessel bridge conditional on X_τ.
4. While $u_2 \in (\prod_{i=1}^{\kappa+2} S^\delta_{2j+1}(l_1, v_1), \prod_{i=1}^{\kappa+2} S^\delta_{2j}(l_1, v_1))$, set $j = j + 1$.
   (a) If $u_2 \leq \prod_{i=1}^{\kappa+2} S^\delta_{2j+1}(l_1, v_1)$, then accept sample path.
   (b) If $u_2 \geq \prod_{i=1}^{\kappa+2} S^\delta_{2j}(l_1, v_1)$, while $u_2 \in (\prod_{i=1}^{\kappa+2} S^\delta_{2k+1}(l_2, v_2), \prod_{i=1}^{\kappa+2} S^\delta_{2k}(l_2, v_2))$, $k = k + 1$.
      i. If $u_2 \leq \prod_{i=1}^{\kappa+2} S^\delta_{2k+1}(l_2, v_2)$, then with probability $1/2$, accept sample path, else return to Step 1.
      ii. If $u_2 \geq \prod_{i=1}^{\kappa+2} S^\delta_{2k+1}(l_2, v_2)$, then reject sample path and return to Step 1.

After accepting a proposed sample path skeleton, simulating the process at further intermediate times conditional on the sample path skeleton is difficult since the information regarding the sample path minimum and maximum induces a dependency between the sub-intervals in which we want to simulate an intermediate point and all other sub-intervals. Further, we know precisely the minimum (or maximum) of the sample path, so the law we need to simulate further points from is a Bessel bridge, contained in some interval. In other words, if we wanted to simulate the process at further times conditional on the accepted sample path, there is a dependency on whether or not the other points in other sub-intervals have achieved this minimum or maximum yet. To remove the induced dependency between the sub-intervals of time, an interval of path-space in which the sample path minimum and maximum is constrained is constructed by *dissecting an intersection layer*. Further points can be simulated later by layered Brownian bridge simulated via *intersection layer approach* (see Pollock [2013, Section 6.3] and Pollock et al. [2016, Section 8]). As the methodology that we develop in this thesis does not require us to simulate the layered Brownian bridge path at further intermediate points, we will not discuss intersection layer constructions here.

4.3 Diffusion Processes

Diffusion processes are widely used across a number of application areas across the natural and social sciences. In finance, they are employed to model stock prices, options, exchange rates, interest rates and many other financial instruments [Black and Scholes, 1973; Merton, 1973, 1976; Chan et al., 1992; Karatzas and Shreve, 1998]. Other applications can be found within biology [McAdams and Arkin, 1997; Golightly and Wilkinson, 2006], genetics [Kimura and Ohta, 1971] and chemistry [Gillespie, 1976, 1977], to name a few. We define a (one-dimensional) diffusion process as:

Definition 4.3.1. Diffusion process. A *diffusion process* $X : \mathbb{R} \to \mathbb{R}$ is a Markov process which satisfies the stochastic differential equation (SDE) of the form:

$$\mathrm{d}V_t = \beta(V_t)\ \mathrm{d}t + \sigma(V_t)\ \mathrm{d}W_t \quad \text{with } V_0 = v \in \mathbb{R},\ t \in [0, T], \tag{4.34}$$

where $\beta : \mathbb{R} \to \mathbb{R}$ and $\sigma : \mathbb{R} \to \mathbb{R}_+$ denote the *drift* and *diffusion* coefficients respectively, and W_t is a standard Brownian motion (see Definition 4.1.1).

Regularity conditions are assumed to ensure the existence of the solution of the SDE in (4.34) (see for instance Yamada and Watanabe [1971]; Kloeden and Platen [1992, 1995], Øksendal [2007, Chapter 5] Rogers and Williams [2000, Chapter V, Section 6]). Several works such as Beskos and Roberts [2005]; Beskos et al. [2006b,a, 2008] have proposed novel methods for the exact simulation of diffusion bridges driven by a class of SDEs with the following conditions:

Condition 4.3.1. Solutions. *The coefficients $\beta(x)$ and $\sigma(x)$ of (4.34) are sufficiently regular to ensure the existence of a unique, non-explosive, weak solution.*

Condition 4.3.2. Continuity. *The drift coefficient $\beta \in C^1$ and the volatility coefficient $\sigma \in C^2$ and is strictly positive.*

Condition 4.3.3. Growth Bound. *There exists $K > 0$ such that $|\beta(x)|^2 + |\sigma(x)|^2 \leq K(1 + |x|^2)$ for all $x \in \mathbb{R}$.*

We are interested in the measure $\mathbb{T}^V_{0,T}$ of V on the sample path induced by (4.34). The problem here is that $\mathbb{T}^V_{0,T}$ is typically unknown and hence a Monte Carlo estimator can be used to estimate expected values $\mathbb{E}_{\mathbb{T}^V_{0,T}}[h(V)]$ for test functions $h : \mathbb{R} \to \mathbb{R}$. Following the Monte Carlo method discussed in Chapter 2, if N independent draws $V^{(1)}, \ldots, V^{(N)} \sim \mathbb{T}^V_{0,T}$ could be obtained, then by the strong law of large numbers, we can construct an estimator for the expectation,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} h\left(V^{(i)}\right) = \mathbb{E}_{\mathbb{T}^V_{0,T}}[h(V)]. \tag{4.35}$$

As with Brownian motion (see Section 4.1) and layered Brownian bridges (see Section 4.2.2), diffusion sample paths are infinite dimensional random variables and hence it is not possible to draw entire sample paths from $\mathbb{T}^V_{0,T}$. Alternatively, we can simulate a finite dimensional subset of the sample path (a *skeleton*). Given these constraints, we must consider the form of h so that it may be evaluated given a finite dimensional subset of a sample path and that any numerical approximation will impact the unbiasedness and convergence of the resulting Monte Carlo estimator.

Diffusion sample paths can be approximately simulated at a finite collection of time points by noting that Brownian motion has a Gaussian transition density (by Property 4.1.3) and thus the transition density of (4.34) can be approximated by that of an SDE with fixed coefficients over

short intervals. This approximation approach is known as *discretisation* [Jacod and Protter, 2012; Kloeden and Platen, 1992]. The most common discretisation scheme for simulating diffusions is the *Euler-Maruyama scheme* [Maruyama, 1955], where a sample path is approximated at each point in time on a fine mesh (of size Δt) by means of the following recursion,

$$V_{t+\Delta t} = V_t + \beta(V_t)\Delta t + \sigma(V_t)\xi, \quad \text{where } \xi \sim \mathcal{N}_1(0, \Delta t). \tag{4.36}$$

Discretisation methods have the property that the approximation error can be minimised by decreasing the size of the fine mesh (i.e. as $\Delta t \to 0$) but this comes at the expense of increased computational cost. Although useful in visualising diffusion sample paths, discretisation techniques are fundamentally approximations and hence result in a loss of unbiasedness of the Monte Carlo estimator in (4.35). Furthermore, for some test functions h, mesh based discretisation schemes do not sufficiently characterise simulated sample paths for the evaluation of h. An example of such case noted in Pollock et al. [2016, Section 1] by considering the case where we are interested whether a simulate sample path $V \sim \mathbb{T}^V_{0,T}$ crosses some barrier (i.e. for some set A, we have $h(V) := \mathbb{1}(V \in A)$). We will also see later in this thesis that we require methods to simulation diffusion bridge paths (i.e. simulating diffusions conditional on an end point $V_T = y$), which numerical methods, such as the Euler-Maruyama scheme, are not suitable for.

To circumvent these drawbacks, there has been significant development of methodologies, referred to as *path-space rejection sampling* or *Exact Algorithms*, which do not introduce approximation error when simulating sample paths at a finite collection of time points [Beskos and Roberts, 2005; Beskos et al., 2006b,a, 2008]. These are rejection sampling (see Section 2.2) based methods on a diffusion path-space, where sample paths are drawn from a proposal measure $\mathbb{P}^V_{0,T}$ which are then accepted (or rejected) with a probability proportional to the Radon-Nikodým derivative of $\mathbb{T}^V_{0,T}$ with respect to $\mathbb{P}^V_{0,T}$. Central to this approach is the notion of simulating *skeletons*.

Definition 4.3.2. Skeleton (of a diffusion sample path). [Pollock et al., 2016, Defintion 1]. A *skeleton*, $\mathcal{S}$, is a finite dimensional representation of a diffusion sample path, $V \sim \mathbb{T}^V_{0,T}$, that can be simulated without any approximation error by means of a proposal sample path drawn from an equivalent proposal measure, $\mathbb{P}^V_{0,T}$, and accepted with probability proportional to $\frac{\mathrm{d}\mathbb{T}^V_{0,T}}{\mathrm{d}\mathbb{P}^V_{0,T}}(V)$, which is sufficient to restore the sample path at any finite collection of time points exactly with finite computation where $V|\mathcal{S} \sim \mathbb{P}^V_{0,T}|\mathcal{S}$. A skeleton typically comprises information regarding the sample path at a finite collection of time points and path-space information which ensures the sample path is almost surely constrained to some compact interval.

To introduce the methodology to simulate sample paths from (4.34), we first review some elements of stochastic calculus and state some key results which are of particular relevance to this methodology. A deeper investigation of stochastic calculus theory can be found in a number of texts: Karatzas and Shreve [1991]; Revuz and Yor [1991]; Kloeden and Platen [1992]; Rogers and Williams [2000]; Øksendal [2007]; Cohen and Elliott [2015]. The remainder of this section is organised as follows:

in Section 4.3.1, we introduce Itô's calculus and Itô's integral before introducing the Lamperti Transformation and Cameron-Martin-Girsanov's Theorem in Sections 4.3.2 and 4.3.3, respectively. Lastly, we look at a representation of the transition density for a class of diffusions in Section 4.3.4.

4.3.1 Itô calculus

The SDE in (4.34) can be interpreted in integrated form,

$$V_T = V_0 + \int_0^T \beta(V_t)\ \mathrm{d}t + \int_0^T \sigma(V_t)\ \mathrm{d}W_t. \tag{4.37}$$

where W_t is standard Brownian motion (see Definition 4.1.1). In this section, we look to provide a definition for integrals of the following form,

$$\mathcal{I}[f] := \int_0^T f(t)\ \mathrm{d}W_t, \tag{4.38}$$

for some suitable function f (which may be parameterised by some other stochastic process). First, we provide an informal proof of the existence of such integrals in the style of Øksendal [2007, Chapter 3.1]: In general, to evaluate an integral of a function $f : \mathbb{R} \to \mathbb{R}$ with respect to another function $g : \mathbb{R} \to \mathbb{R}$ over an interval $[0, T]$, we can partition the interval $[0, T]$ into a fine mesh $\mathcal{T} := \{t_i := iT/N : i = 1, \ldots, N\}$, provided that g has bounded variation on compact time intervals (which is a necessary condition [Banach and Steinhaus, 1927]),

$$\mathbb{V}_{[0,T]}(g) := \lim_{N\to\infty} \sum_{i=1}^{N} |g(t_i) - g(t_{i-1})| < \infty, \tag{4.39}$$

then we can define the integral in the Riemann–Stieltjes sense [Stieltjes, 1894],

$$\int_0^T f(t)\ \mathrm{d}g(t) := \lim_{N\to\infty} \sum_{i=1}^{N} f(s_i) \cdot [g(t_i) - g(t_{i-1})], \quad \text{where } s_i \in [t_{i-1}, t_i]. \tag{4.40}$$

However, as Øksendal [2007, Chapter 3.1] notes, the variations of the paths of Brownian motion are too large for us to define the integral (4.38) in the Riemann-Stieltjes sense. In particular, the total variation of Brownian motion is infinite (i.e. $\mathbb{V}_{[0,T]}(W) = \infty$). Further, unlike the Riemann-Stieltjes integral, evaluating the function f in the integral (4.38) at different points will produce different answers. For example, consider the expectation of (4.38) with respect to the measure $\mathbb{W}^0_{0,T}$ in the case where $f(t) := W_t$ and $s_i := t_{i-1}$ (evaluating at the left end point),

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{N} W_{t_i-1} \cdot [W_{t_i} - W_{t_{i-1}}]\right] &= \sum_{i=1}^{N} \mathbb{E}\left[W_{t_{i-1}} \cdot [W_{t_i} - W_{t_{i-1}}]\right] \\ &= \sum_{i=1}^{N} \mathbb{E}[W_{t_i-1}] \cdot \mathbb{E}[W_{t_i} - W_{t_{i-1}}] = 0.\end{aligned} \tag{4.41}$$

Consider now we choose a different evaluation point, $s_i = t_i$ (evaluating at the right end point),

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{N} W_{t_i} \cdot [W_{t_i} - W_{t_{i-1}}]\right] &= \sum_{i=1}^{N} \mathbb{E}\left[W_{t_i}^2 - [W_{t_i} \cdot W_{t_{i-1}}]\right] \\ &= \sum_{i=1}^{N} (t_i - t_{i-1}) = T. \end{aligned} \tag{4.42}$$

The choice of evaluating at the left end points (i.e. the approximation in (4.41) to (4.40)) leads to the *Itô integral* which we consider in the remainder of this thesis, as this preserves the martingale property [Øksendal, 2007, Section 3.1]. By using the approximation in (4.41), we can make use of the property of Brownian motion that it has independent increments (Property 4.1.2). Evaluating at the mid points, $s_i := (t_{i-1} + t_i)/2$, and approximating (4.40) as above leads to the *Stratonovich integral* Øksendal [2007, Example 3.1.1] which we do not cover in this thesis (see Øksendal [2007, Section 3.3] for a comparison of Itô and Stratonovich integrals).

Continuing to follow the approach outlined in Øksendal [2007, Chapter 3.1], we can construct the Itô integral by considering the integration of *elementary functions* of the following form,

$$\phi_N(t) := \sum_{i=1}^{N} e_i \cdot \mathbb{1}\left[t \in [t_{i-1} - t_i)\right] \tag{4.43}$$

where $\mathbb{1}[\cdot]$ denotes the indicator function, $N \in \mathbb{N}$ and e_i are $\mathcal{F}_{t_{i-1}}$-measurable random variables where $\mathcal{F}_t$ is the σ-algebra generated by the random variables $\{W_s\}_{s\in[0,t]}$.

Lemma 4.3.1. [Øksendal, 2007, Lemma 3.1.5]. *Let $\phi_N(t)$ be bounded and given by* (4.43)*, then*

$$\mathbb{E}\left[\left(\int_0^T \phi_N(t)\ \mathrm{d}W_t\right)^2\right] = \mathbb{E}\left[\int_0^T \phi_N(t)^2\ \mathrm{d}t\right]. \tag{4.44}$$

Proof. Recalling that Brownian motion has independent increments (see Property 4.1.2) which are normally distributed (see Property 4.1.3), then we have

$$\begin{aligned}\mathbb{E}\left[\left(\int_0^T \phi_N(t)\ \mathrm{d}W_t\right)^2\right] &= \mathbb{E}\left[\left(\sum_{i=1}^{N} e_i \cdot [W_{t_i} - W_{t_{i-1}}]\right)^2\right] \\ &= \mathbb{E}\left[\sum_{i=1}^{N} e_i^2 \cdot [W_{t_i} - W_{t_{i-1}}]^2 + \sum_{\substack{i,j=1 \\ j\neq i}} e_i \cdot [W_{t_i} - W_{t_{i-1}}] \cdot e_j \cdot [W_{t_j} - W_{t_{j-1}}]\right] \\ &= \sum_{i=1}^{N} \mathbb{E}\left[e_i^2\right] \cdot \mathbb{E}\left[[W_{t_i} - W_{t_{i-1}}]^2\right]\end{aligned}$$

$$= \sum_{i=1}^{N} \mathbb{E}\left[e_i^2\right] \cdot (t_i - t_{i-1})$$
$$= \mathbb{E}\left[\int_0^T \phi_N^2(t)\ \mathrm{d}t\right]. \qquad \blacksquare$$

To construct Itô's integral, after defining the integral $\mathcal{I}(\phi_N)$ for a simple class of functions ϕ_N for a simple class of functions ϕ_N, Øksendal [2007, Chapter 3.1] showed that any continuous process $f(t)$ with $\mathbb{E}\left[\int_0^T f^2(t)\right] < \infty$ can be approximated by a sequence of elementary processes, i.e. there exists a sequence of processes $\{\phi_N : N \in \mathbb{N}\}$ each defined as in (4.43) such that

$$\lim_{N\to\infty} \mathbb{E}\left[\int_0^T |f_t - \phi_N(t)|\ \mathrm{d}t\right] = 0.$$

Together with (4.44), we conclude there exists a limit $I(T) := \lim_{N\to\infty} \mathcal{I}[\phi_N]$, such that

$$\lim_{N\to\infty} \mathbb{E}\left[\left|\int_0^T \phi_N(t)\ \mathrm{d}W_t - I(T)\right|^2\right] = 0.$$

Consequently, we define *Itô's integral* to be this limit,

$$\int_0^T f(t)\ \mathrm{d}W_t := I(T) = \lim_{N\to\infty} \int_0^T \phi_N(t)\ \mathrm{d}W_t. \tag{4.45}$$

Further, we note that (4.44) also holds for this limit which is a property known as *Itô's isometry* [Øksendal, 2007, Corollary 3.1.7],

$$\mathbb{E}\left[\left(\int_0^T f(t)\ \mathrm{d}W_t\right)^2\right] = \mathbb{E}\left[\int_0^T f(t)^2\ \mathrm{d}t\right]. \tag{4.46}$$

The basic definition of Itô's integral is often impractical when tasked with evaluating a given integral. However, it is possible to establish an Itô integral version of the chain rule. Let $F \in C^2$ (i.e. F is twice continuously differentiable on $\mathbb{R}$) be the anti-derivative of f, then by considering a Taylor series expansion of F, we have

$$\begin{aligned} F(T) &= F(0) + \lim_{N\to\infty}\left[\frac{1}{1!}\sum_{i=1}^{N} f(t_i)\cdot[W_{t_i} - W_{t_{i-1}}] + \frac{1}{2!}\sum_{i=1}^{N} f'(t_i)\cdot[W_{t_i} - W_{t_{i-1}}]^2 + \dots\right] \\ &= F(0) + \int_0^T f(t)\ \mathrm{d}W_t + \frac{1}{2}\int_0^T f'(t)\ \mathrm{d}t + 0. \end{aligned} \tag{4.47}$$

This result is known as *Itô's formula* [Øksendal, 2007, Definition 4.1.1], which can also be given in differential form,

$$\mathrm{d}F(t) = f(t)\ \mathrm{d}W_t + \frac{1}{2}f'(t)\ \mathrm{d}t. \tag{4.48}$$

The key to arriving at this result lies in noting that by considering the variation of Brownian motion

over the time interval $[0, T]$, we have

$$\lim_{N\to\infty} \sum_{i=1}^{N} \left|W_{t_i} - W_{t_{i-1}}\right|^p = \begin{cases} \infty & \text{if } p = 1, \\ T & \text{if } p = 2, \\ 0 & \text{if } p \geq 3. \end{cases}$$

We can generalise this for diffusions given by (4.34) where $\mathbb{E}\left[\int_0^T \sigma^2(V_t)\ \mathrm{d}t\right] < \infty$. Consider a Taylor series expansion of $X_t := g(t, V_t)$ where $g \in C^2([0,\infty) \times \mathbb{R})$ (i.e. g is twice continuously differentiable on $[0,\infty) \times \mathbb{R}$) and the derivatives of g are evaluated at (t_i, V_{t_i}) for $i = 1, \ldots, N$,

$$\begin{aligned} X_T = X_0 + \Bigg[&\sum_{i=1}^{N} \frac{\partial g}{\partial t}[t_i - t_{i-1}] + \sum_{i=1}^{N} \frac{\partial g}{\partial v}[V_{t_i} - V_{t_{i-1}}] + \frac{1}{2}\sum_{i=1}^{N} \frac{\partial^2 g}{\partial t^2}[t_i - t_{i-1}]^2 \\ &+ \sum_{i=1}^{N} \frac{\partial^2 g}{\partial t \partial v}[t_i - t_{i-1}][V_{t_i} - V_{t_{i-1}}] + \frac{1}{2}\sum_{i=1}^{N} \frac{\partial^2 g}{\partial v^2}[V_{t_i} - V_{t_{i-1}}]^2 + \ldots \Bigg]. \end{aligned} \tag{4.49}$$

For small time intervals of size $h > 0$, $V_{t+h} - V_t \approx \beta(V_t)[(t+h) - t] + \sigma(V_t)[W_{t+h} - W_t]$, so we have,

$$\begin{aligned} \lim_{N\to\infty} \left[\sum_{i=1}^{N} \frac{\partial g}{\partial v}(V_{t_i} - V_{t_{i-1}})\right] &= \lim_{N\to\infty} \left[\sum_{i=1}^{N} \frac{\partial g}{\partial v}\beta(V_{t_i})[t_i - t_{i-1}] + \sum_{i=1}^{N} \frac{\partial g}{\partial v}\sigma(V_{t_i})[W_{t_i} - W_{t_{i-1}}]\right] \\ &= \int_0^T \frac{\partial g(t, V_t)}{\partial v}\beta(V_t)\ \mathrm{d}t + \int_0^T \frac{\partial g(t, V_t)}{\partial v}\sigma(V_t)\ \mathrm{d}W_t, \end{aligned} \tag{4.50}$$

and

$$\begin{aligned} \lim_{N\to\infty} \left[\sum_{i=1}^{N} \frac{\partial g}{\partial v}(V_{t_i} - V_{t_{i-1}})\right] &= \lim_{N\to\infty} \Bigg[\sum_{i=1}^{N} \frac{\partial^2 g}{\partial v^2}\beta^2(V_{t_i})[t_i - t_{i-1}]^2 + \sum_{i=1}^{N} \frac{\partial^2 g}{\partial v^2}\sigma^2(V_{t_i})[W_{t_i} - W_{t_{i-1}}]^2 \\ &\qquad + 2\sum_{i=1}^{N} \frac{\partial^2 g}{\partial v^2}\beta(V_{t_i})\sigma(V_{t_i})[t_i - t_{i-1}][W_{t_i} - W_{t_{i-1}}]\Bigg] \\ &= 0 + \int_0^T \frac{\partial^2 g(t, V_t)}{\partial v^2}\sigma^2(V_t)\ \mathrm{d}t + 0, \end{aligned} \tag{4.51}$$

By substituting (4.50) and (4.51) into (4.49), we have

$$\begin{aligned} X_T = X_0 &+ \int_0^T \left(\frac{\partial g(t, V_t)}{\partial t} + \frac{\partial g(t, V_t)}{\partial v}\beta(V_t) + \frac{1}{2}\frac{\partial^2 g(t, V_t)}{\partial v^2}\sigma^2(V_t)\right)\ \mathrm{d}t \\ &+ \int_0^T \frac{\partial g(t, V_t)}{\partial v}\sigma(V_t)\ \mathrm{d}W_t. \end{aligned} \tag{4.52}$$

This is *Itô's formula for diffusions* [Øksendal, 2007, Theorem 4.1.2], and its differential form is

$$\mathrm{d}X_t = \left(\frac{\partial g(t, V_t)}{\partial t} + \frac{\partial g(t, V_t)}{\partial v}\beta(V_t) + \frac{1}{2}\frac{\partial^2 g(t, V_t)}{\partial v^2}\sigma^2(V_t)\right)\ \mathrm{d}t + \frac{\partial g(t, V_t)}{\partial v}\sigma(V_t)\ \mathrm{d}W_t. \tag{4.53}$$

4.3.2 Lamperti transformation

The methodology discussed in the subsequent sections for the simulation of sample paths of diffusions are for those which have unit volatility (i.e. $\sigma(V_t) = 1$ in (4.34)). However, this is not restrictive, since we can also apply this methodology to a broader class of diffusions with non-unit volatility (as in (4.34)) by means of first transforming the target diffusion in (4.34) into one with unit volatility. The simulated sample paths are then transformed back using the inverse transformation. This transformation is obtained by using Itô's formula (see Section 4.3.1) and by defining a process $Y_t := g(t, V_t)$ such that $\frac{\partial g(t,V_t)}{\partial v} = \frac{1}{\sigma(V_t)}$, then the differential form of Y_t has unit volatility [Beskos and Roberts, 2005, Section 1]. This is known as the *Lamperti transform* (see for instance Kloeden and Platen [1992]; Aït-Sahalia [2008]; Casella and Roberts [2011]),

$$Y_t = \int_{v_0}^{V_t} \frac{1}{\sigma(u)} \, \mathrm{d}u \tag{4.54}$$

where v_0 is an arbitrary element of the state space of V. By applying Itô's formula (4.53) to find $\mathrm{d}Y_t$, first note that,

$$\begin{aligned}
\frac{\partial g}{\partial v} &= \frac{1}{\sigma(V_t)}, \\
\frac{\partial g}{\partial t} &= \frac{\partial}{\partial t} \int_{v_0}^{V_t} \frac{1}{\sigma(u)} \mathrm{d}u = 0, \\
\frac{\partial^2 g}{\partial v^2} &= \frac{-\sigma'(V_t)}{\sigma^2(V_t)}.
\end{aligned}$$

where $\sigma'(V_t) = \frac{\partial \sigma(V_t)}{\partial v}$. By substitution into (4.53), we can obtain,

$$\begin{aligned}
\mathrm{d}Y_t &= \frac{1}{\sigma(V_t)} \left(\beta(V_t) \, \mathrm{d}t + \sigma(V_t) \, \mathrm{d}W_t\right) + \frac{1}{2} \left(\frac{-\sigma'(V_t)}{\sigma^2(V_t)}\right) \sigma^2(V_t) \, \mathrm{d}t \\
&= \alpha(Y_t) \, \mathrm{d}t + \, \mathrm{d}W_t
\end{aligned} \tag{4.55}$$

where $\alpha(Y_t) = \left(\frac{\beta(g^{-1}(Y_t))}{\sigma'(g^{-1}(Y_t))} - \frac{\sigma(g^{-1}(Y_t))}{2}\right)$ and let $V_t = g^{-1}(t, Y_t)$ to be the inverse of the Lamperti transformation. Condition 4.3.2 and Condition 4.3.3 are sufficient to allow us to transform any non-unit volatility SDE into one with unit volatility as per the Lamperti transformation.

4.3.3 Cameron-Martin-Girsanov's theorem

The Lamperti transformation discussed in Section 4.3.2 allows us to transform our SDE into one with unit volatility. In this section, we restrict our attention to diffusions with the following SDE:

$$\mathrm{d}X_t = \alpha(X_t) \, \mathrm{d}t + \mathrm{d}W_t \quad \text{ with } X_0 = x \in \mathbb{R}, \; t \in [0, T]. \tag{4.56}$$

As discussed earlier in the section, the methodology for simulating skeletons of (4.56) (see Definition 4.3.2) is based on a rejection sampling idea (see Section 2.2). Consequently, we must find an expression for the Radon-Nikodým derivative of our target measure with respect to a proposal measure. Let $\mathbb{Q}^x_{0,T}$ denote the *target* measure induced by (4.56) and let $\mathbb{W}^x_{0,T}$ be the *proposal* measure induced by the driftless version of (4.56) (i.e. $\mathbb{W}^x_{0,T}$ is the *Wiener measure*):

$$\mathrm{d}X_t = \mathrm{d}W_t \quad \text{with } X_0 = x \in \mathbb{R},\ t \in [0,T]. \tag{4.57}$$

Since both (4.56) and (4.57) have unit volatility, $\mathbb{Q}^x_{0,T}$ is absolutely continuous with respect to $\mathbb{W}^x_{0,T}$.

Theorem 4.3.2. The Cameron-Martin-Girsanov theorem. (see for instance Cameron and Martin [1944], Girsanov [1960], Rogers and Williams [2000, Chapter VI, Section 6.38], Beskos and Roberts [2005, Proposition 2], Øksendal [2007, Chapter 8]). *Assume that the drift coefficient α satisfies* Novikov's condition*,*

$$\mathbb{E}_{\mathbb{W}}\left[\exp\left\{\frac{1}{2}\int_0^T \alpha^2(W_t)\ \mathrm{d}t\right\}\right] < \infty. \tag{4.58}$$

Then the Radon-Nikodým derivative of $\mathbb{Q}^x_{0,T}$ with respect to $\mathbb{W}^x_{0,T}$ exists and is given by the Cameron-Martin-Girsanov's formula *as follows*

$$\frac{\mathrm{d}\mathbb{Q}^x_{0,T}}{\mathrm{d}\mathbb{W}^x_{0,T}}(X) = \exp\left\{\int_0^T \alpha(X_s)\ \mathrm{d}W_s - \frac{1}{2}\int_0^T \alpha^2(X_s)\ \mathrm{d}s\right\}. \tag{4.59}$$

Proof. See for instance, Øksendal [2007, Chapter 8]. ■

The difficulty in using this result directly in a rejection sampling scheme is that the direct and exact evaluation of (4.59) is not possible. Here, we first simplify (4.59) by using Itô's formula to remove the Itô integral term. Let

$$A(u) := \int_0^u \alpha(s)\ \mathrm{d}s, \tag{4.60}$$

then, under Conditions 4.3.1–4.3.3, we can apply Itô's formula,

$$\mathrm{d}A(X_s) = A'(X_s)\ \mathrm{d}W_s + \frac{1}{2}A''(X_s)\ \mathrm{d}s,$$

and so integrating between 0 and T,

$$A(X_T) - A(X_0) = \int_0^T \alpha(X_s)\ \mathrm{d}W_s + \frac{1}{2}\int_0^T \alpha'(X_s)\ \mathrm{d}s.$$

By rearrangement, we obtain

$$\int_0^T \alpha(X_s)\ \mathrm{d}W_s = A(X_T) - A(X_0) - \frac{1}{2}\int_0^T \alpha'(X_s)\ \mathrm{d}s.$$

If we substitute this into (4.59), and define

$$\phi(X_s) := \frac{1}{2}\left(\alpha^2(X_s) + \alpha'(X_s)\right), \tag{4.61}$$

then

$$\begin{aligned}\frac{\mathrm{d}\mathbb{Q}^x_{0,T}}{\mathrm{d}\mathbb{W}^x_{0,T}}(X) &= \exp\left\{A(X_T) - A(X_0) - \frac{1}{2}\int_0^T \alpha'(X_s)\ \mathrm{d}s - \frac{1}{2}\int_0^T \alpha^2(X_s)\ \mathrm{d}s\right\} \\ &= \exp\left\{A(X_T) - A(X_0) - \int_0^T \phi(X_s)\ \mathrm{d}s\right\}. \end{aligned} \tag{4.62}$$

In Section 4.4, we will see that we can use this result to carry out a rejection scheme for simulating sample paths from our target diffusion (4.57) indirectly using this result.

4.3.4 Transition density of a diffusion

The transition density of the target diffusion given by (4.56) can be denoted as

$$p_{t-s}(x,y) := \mathbb{P}(X_t \in \mathrm{d}y | X_s = x)/\mathrm{d}y, \qquad \text{where } 0 \le s < t. \tag{4.63}$$

We can obtain expressions for the transition density in the style of Dacunha-Castelle and Florens-Zmirou [1986, Lemma 1] by considering its expectation with respect to a Wiener measure. Let $\mathbb{Q}^x_{0,T}$ and $\mathbb{W}^x_{0,T}$ be the measures induced by (4.56) and (4.57), respectively, with the additional constraint that $X_t = y$. If w_{t-s} is the transition density of Brownian motion from x to y over $[s,t]$, then

$$\frac{\mathrm{d}\mathbb{Q}^x_{s,t}}{\mathrm{d}\mathbb{W}^x_{s,t}}(X) = \frac{p_{t-s}(x,y)}{w_{t-s}(x,y)} \cdot \frac{\mathrm{d}\mathbb{Q}^{x,y}_{s,t}}{\mathrm{d}\mathbb{W}^{x,y}_{s,t}}(X).$$

By taking expectations with respect to $\mathbb{W}^{x,y}_{s,t}$, we have

$$\begin{aligned}\mathbb{E}_{\mathbb{W}^{x,y}_{s,t}}\left[\frac{\mathrm{d}\mathbb{Q}^x_{s,t}}{\mathrm{d}\mathbb{W}^x_{s,t}}(X)\right] &= \mathbb{E}_{\mathbb{W}^{x,y}_{s,t}}\left[\frac{p_{t-s}(x,y)}{w_{t-s}(x,y)} \cdot \frac{\mathrm{d}\mathbb{Q}^{x,y}_{s,t}}{\mathrm{d}\mathbb{W}^{x,y}_{s,t}}(X)\right] \\ &= \frac{p_{t-s}(x,y)}{w_{t-s}(x,y)} \cdot 1,\end{aligned}$$

and by rearrangement,

$$\begin{aligned} p_{t-s}(x,y) &= w_{t-s}(x,y) \cdot \mathbb{E}_{\mathbb{W}^{x,y}_{s,t}}\left[\frac{\mathrm{d}\mathbb{Q}^x_{s,t}}{\mathrm{d}\mathbb{W}^x_{s,t}}(X)\right] \\ &= \frac{1}{\sqrt{2\pi(t-s)}}\exp\left\{-\frac{(y-x)^2}{2(t-s)}\right\} \cdot \mathbb{E}_{\mathbb{W}^{x,y}_{s,t}}\left[\exp\left\{A(y) - A(x) - \int_s^t \phi(X_u)\ \mathrm{d}u\right\}\right] \\ &= \frac{1}{\sqrt{2\pi(t-s)}}\exp\left\{A(y) - A(x) - \frac{(y-x)^2}{2(t-s)}\right\} \cdot \mathbb{E}_{\mathbb{W}^{x,y}_{s,t}}\left[\exp\left\{-\int_s^t \phi(X_u)\ \mathrm{d}u\right\}\right]. \end{aligned} \tag{4.64}$$

The difficulty with using this expression is that the expectation with respect to Brownian bridge measure cannot be evaluated directly. However, the methodology discussed in Section 4.4 enables us to simulate from the transition density exactly. This is particularly useful for the Fusion methodology developed in this thesis since the mathematics underpinning the methodology includes transition densities of a Langevin diffusion process (which we detail later in Section 5).

4.4 Simulating diffusion processes

The Fusion methodologies developed in this thesis rely on the computation of unbiased estimators which arise in the simulation of finite dimensional subsets of diffusion sample paths. In Section 4.3, we provided an overview of diffusions and introduced some elements of stochastic calculus which underpin the methodology to simulate sample paths of diffusions. In this section, we introduce the mathematical framework for simulating diffusion sample path skeletons (see Definition 4.3.2) without approximation error. This methodology is referred to as *path-space rejection sampling* or *Exact Algorithms* and can be found in a number of texts: Beskos and Roberts [2005]; Beskos et al. [2006b,a, 2008, 2012]; Chen and Huang [2013]; Dai [2014]; Pollock [2013]; Pollock et al. [2016].

As a result of the Lamperti transformation (see Section 4.3.2), we will focus on diffusions with unit volatility (4.56) (whilst noting the simulation of diffusions with non-unit volatility (4.34) can be achieved via (4.54)). The underpinning aim of path-space rejection sampling is that to simulate sample paths from the target measure $\mathbb{Q}^x_{0,T}$ induced by (4.56) means of rejection sampling (see Section 2.2). In this setting, sample paths from an accessible proposal measure can be accepted as being paths from the target measure so long as we are able to find a bound, M, on the Radon-Nikodým derivative of our target measure with respect to our proposal measure. Naturally, since we are able to simulate Brownian motion exactly, we choose the Wiener measure $\mathbb{W}^x_{0,T}$ (induced by the driftless SDE (4.57)) as the proposal measure. As noted previously, since (4.56) and (4.57) both have unit volatility, then $\mathbb{Q}^x_{0,T}$ is absolutely continuous with respect to $\mathbb{W}^x_{0,T}$. Algorithm 4.4.1 provides an outline of a path-space rejection sampling approach.

Algorithm 4.4.1 Outline of path-space rejection sampling to simulate sample paths $X \sim \mathbb{Q}^x_{0,T}$ [Beskos and Roberts, 2005].

1. Draw $X \sim \mathbb{W}^x_{0,T}$.
2. Return X with probability

$$P_{\mathbb{W}^x_{0,T}}(X) := \frac{1}{M}\frac{\mathrm{d}\mathbb{Q}^x_{0,T}}{\mathrm{d}\mathbb{W}^x_{0,T}}(X) \in [0,1],$$

   else reject and return to Step 1.

Unfortunately, Algorithm 4.4.1 cannot be directly implemented for two reasons. Firstly, recall from Section 4.3.3, by using the Cameron-Martin-Girsanov theorem (see Theorem 4.3.2), we obtained an expression for the Radon-Nikodým derivative in (4.62). In a rejection sampling algorithm, we must

be able to find an appropriate upper bound M of the Radon-Nikodým derivative such that the acceptance probability in Step 2 is bounded in $[0, 1]$. However, $A(u)$ (4.60) in (4.62) is not bounded since it only has a quadratic growth bound (as a result of Condition 4.3.3), and so typically no appropriate bound $M < \infty$ exists. In Section 4.4.1, we show that this can be circumvented by considering an alternative probability measure to propose sample paths from. Secondly, diffusion sample paths are infinite dimensional random variables and so it is not possible to draw entire proposal sample paths in Step 1. As such, it is not possible to evaluate the Radon-Nikodým derivative in Step 2 directly. In particular, we are not able to evaluate expressions of the form,

$$\exp\left\{-\int_s^t \phi(X_u)\ \mathrm{d}u\right\}. \tag{4.65}$$

We will see in Section 4.4.2 that it is possible to construct an unbiased estimator (which is bounded above) for (4.65) only using a finite dimensional subset of the proposal sample path.

The methodology that we develop in Section 6 embeds the Fusion methodology within a Divide-and-Conquer Sequential Monte Carlo (see Section 3.4) approach Lindsten et al. [2017]. In this setting, since we move away from a rejection based scheme, we no longer require that the unbiased estimator for (4.65) to be bounded above by a constant. In Section 4.4.3, we explore alternative unbiased estimators for expressions of the form (4.65) which have better asymptotic properties.

4.4.1 Path-space rejection sampling

In order to implement Algorithm 4.4.1, we must be able to find an appropriate upper bound on the Radon-Nikodým derivative in (4.62). As noted previously, the function $A(u)$ (4.60) is not bounded as it only has a quadratic growth bound (see Condition 4.3.3). To deal with this unbounded function in the Radon-Nikodým derivative, an end point, y, must be specified which subsequently modifies the acceptance probability. In this setting, the proposal measure becomes a Brownian bridge (see Definition 4.1.2). To remove the unbounded function, we can use *biased Brownian motion* (as introduced in Beskos and Roberts [2005]) to propose an end point y of the bridge.

Definition 4.4.1. Biased Brownian Motion (BBM). [Beskos and Roberts, 2005; Pollock et al., 2016] Biased Brownian motion is the process $Z_t \sim (W_t | W_0 = x, W_T := y \sim h)$ with measure $\mathbb{Z}_{0,T}^x$, where $x, y \in \mathbb{R}$, $t \in [0, T]$ and h is defined as

$$h(y; x, T) := \frac{1}{c(x, T)} \exp\left\{A(y) - \frac{(y - x)^2}{2T}\right\}. \tag{4.66}$$

Note that by Condition 4.3.3, A has quadratic growth bound, and so

$$c(x, T) := \int_{\mathbb{R}} \exp\left\{A(u) - \frac{u^2}{2T}\right\}\ \mathrm{d}u < \infty.$$

Theorem 4.4.1. [Beskos and Roberts, 2005, Proposition 3]. *Let $\mathbb{Q}^x_{0,T}$ denote the measure induced by* (4.56) *and $\mathbb{Z}^x_{0,T}$ denote the Biased Brownian bridge measure defined in Definition 4.4.1, then*

$$\frac{\mathrm{d}\mathbb{Q}^x_{0,T}}{\mathrm{d}\mathbb{Z}^x_{0,T}}(X) \propto \exp\left\{-\int_0^T \phi(X_s)\ \mathrm{d}s\right\}. \tag{4.67}$$

Proof. Denote $\mathbb{Z}^{x,y}_{0,T}$ be the measure induced by Z and $\mathbb{W}^{x,y}_{0,T}$ be the Wiener measure on $[0,T]$ conditioned on $X_0 = x \in \mathbb{R}$ and $X_T = y \in \mathbb{R}$. Then noting that $\mathbb{Z}^{x,y}_{0,T} = \mathbb{W}^{x,y}_{0,T}$, then

$$\frac{\mathrm{d}\mathbb{Z}^x_{0,T}}{\mathrm{d}\mathbb{W}^x_{0,T}}(X) = \frac{h(y;x,T)}{\frac{1}{\sqrt{2\pi T}}\exp\left\{-\frac{(y-x)^2}{2T}\right\}} \cdot \frac{\mathrm{d}\mathbb{Z}^{x,y}_{0,T}}{\mathrm{d}\mathbb{W}^{x,y}_{0,T}}(X) \propto \exp\{A(X_T)\},$$

Now note that

$$\begin{aligned}
\frac{\mathrm{d}\mathbb{Q}^x_{0,T}}{\mathrm{d}\mathbb{Z}^x_{0,T}} &= \frac{\mathrm{d}\mathbb{Q}^x_{0,T}}{\mathrm{d}\mathbb{W}^x_{0,T}} \cdot \frac{\mathrm{d}\mathbb{W}^x_{0,T}}{\mathrm{d}\mathbb{Z}^x_{0,T}} \\
&\propto \exp\left\{A(X_T) - A(X_0) - \frac{1}{2}\int_0^T \phi(X_s)\ \mathrm{d}s\right\} \cdot \exp\{-A(X_T)\} \\
&\propto \exp\left\{-\int_0^T \phi(X_s)\ \mathrm{d}s\right\},
\end{aligned}$$

which gives the result as required. ■

Sample paths can be drawn from $\mathbb{Z}^x_{0,T}$ in two steps: (i) simulate the end point $X_T := y \sim h$ (which can be simulated via a rejection sampler with Gaussian proposal), and (ii) simulate the remainder of the sample path in $(0,T)$ from the law of a Brownian bridge, $(X|X_0 = x, X_T = y) \sim \mathbb{W}^{x,y}_{0,T}$ (which can be simulated as per Algorithm 4.1.2). Before discussing how to proceed in constructing a rejection sampler to draw sample paths from $\mathbb{Q}^x_{0,T}$, we first introduce the following condition:

Condition 4.4.1. *There exists a constant $\mathbf{\Phi} > -\infty$ such that $\mathbf{\Phi} \leq \phi(x)$ for all $x \in \mathbb{R}$.*

With this condition, we can bound the Radon-Nikodým derivative in (4.67) with $M := \exp\{-\mathbf{\Phi}T\}$, and so we now construct an (idealised) rejection sampler (as outlined in Algorithm 4.4.2) to draw sample paths from $\mathbb{Q}^x_{0,T}$ by drawing proposal paths from $\mathbb{Z}^x_{0,T}$ and accepting such paths with probability $P_{\mathbb{Z}^x_{0,T}}(X) = \frac{1}{M}\frac{\mathrm{d}\mathbb{Q}^x_{0,T}}{\mathrm{d}\mathbb{Z}^x_{0,T}} = \exp\{\mathbf{\Phi}T\} \cdot \exp\left\{-\int_0^T \phi(X_s)\ \mathrm{d}s\right\}$.

However, implementing a rejection sampler is still not yet possible since, as previously noted, diffusion sample paths (and Brownian bridge sample paths) are infinite dimensional random variables. As a result, it is not possible to draw entire paths or evaluate the acceptance probability in Step 2 in Algorithm 4.4.2. To circumvent this problem, we can construct *unbiased estimators* for the acceptance probability which can be evaluated using only a finite dimensional subset of the proposal path [Pollock et al., 2016], the details of which are covered in the subsequent section.

Algorithm 4.4.2 Idealised path-space rejection sampler to simulate sample paths $X \sim \mathbb{Q}^x_{0,T}$ [Beskos and Roberts, 2005].

1. Draw $X \sim \mathbb{Z}^x_{0,T}$,
   (a) Simulate $X_T =: y \sim h$ (4.66).
   (b) Simulate $X_{(0,T)} \sim \mathbb{W}^{x,y}_{0,T}$.
2. Return X with probability

$$P_{\mathbb{Z}^x_{0,T}}(X) := \exp\{\mathbf{\Phi}T\} \cdot \exp\left\{-\int_0^T \phi(X_s)\ \mathrm{d}s\right\} \in [0,1],$$

   else reject and return to Step 1.

4.4.2 Unbiased estimator construction for path-space rejection sampling

In this section, we focus on constructing an implementable rejection sampler to simulate sample paths $X \sim \mathbb{Q}^x_{0,T}$ which only require finite computation. As discussed in Section 4.4.1, Algorithm 4.4.2 cannot be implemented since Brownian bridge sample paths are infinite dimensional random variables. Consequently, we are unable to compute the acceptance probability,

$$P_{\mathbb{Z}^x_{0,T}}(X) = \exp\{\mathbf{\Phi}T\} \cdot \exp\left\{-\int_0^T \phi(X_s)\ \mathrm{d}s\right\}. \tag{4.68}$$

We can overcome this issue by noting that an auxiliary finite dimensional random variable $F \sim \mathbb{F}$ can be simulated in order to construct an unbiased estimator for the acceptance probability using only a finite dimensional subset of the proposal path [Beskos and Roberts, 2005; Beskos et al., 2008; Pollock et al., 2016]. We can use the simulation of F to provide the information about what finite subset of $\mathbb{W}^{x,y}_{0,T}$ to simulate, $X^{\text{fin}} \sim \mathbb{W}^{x,y}_{0,T}|F$, in order to evaluate the acceptance probability in (4.68). The rest of the sample path can be simulated after the acceptance of the sample path from the proposal measure conditioned on the simulations already performed, $X^{\text{rem}} \sim \mathbb{W}^{x,y}_{0,T}|(X^{\text{fin}}, F)$, where $X = X^{\text{fin}} \cup X^{\text{rem}}$. The key insight is that although we are not able to simulate entire sample paths from the proposal, we are able to simulate *exactly* a finite dimensional subset of the sample path, characterised by its *skeleton*, $\mathcal{S}(X) := \{X_0, X^{\text{fin}}, X_T, F\}$ (see Definition 4.3.2).

Beskos et al. [2006b] first noted that it is possible to construct an unbiased estimator for the acceptance probability in (4.68) provided that $\phi(X_{[0,T]})$ can be bounded above. However, Beskos et al. [2008] removed this condition by noting that local bounds for $\phi(X_{[0,T]})$ were sufficient in obtaining an unbiased estimator. In particular, Beskos et al. [2008] noted that if the proposal Brownian bridge sample path was constrained to an interval, then conditional on this interval, $\phi(X_{[0,T]})$ was bounded above and below and hence the random variable F can be simulated, as required. This methodology is based on the notion that the path-space of the proposal measure $\mathbb{P}^x_{0,T}$ (in the case of Biased Brownian motion, $\mathbb{P}^x_{0,T} = \mathbb{Z}^x_{0,T}$) can be partitioned into a set of *layers*, and that they layer to which any sample path belongs to can be simulated.

Definition 4.4.2. Layer. [Pollock et al., 2016]. A *layer* $R(V)$ is a function of a diffusion sample path $V \sim \mathbb{P}^x_{0,T}$ which determines the compact interval to which the sample path is constrained.

The construction of path-space *layers* of Brownian bridge was discussed in Section 4.2.2 allow us to partition the space of $\mathbb{Z}^{x,y}_{0,T}$ into disjoint layers and simulate the layer, $R \sim \mathcal{R}$, to which our sample path belongs (as per Algorithm 4.2.4). Conditional on this layer information, upper and lower bounds for $\phi(X_{[0,T]})$ can always be found, denoted $U_X \in \mathbb{R}$ and $L_X \in \mathbb{R}$, respectively. Further, as noted in Section 4.2.2, we are able to simulate any finite collection of intermediate points of the trajectory of the proposal layered Brownian bridge exactly (as per Algorithm 4.2.5).

Following in the style of Beskos et al. [2006b, 2008], we can construct construct a finite dimensional unbiased estimator for (4.68) by considering a Taylor series expansion as follows,

$$\begin{aligned} P_{\mathbb{Z}^x_{0,T}}(X) &= e^{-(L_X-\mathbf{\Phi})T} \cdot e^{-(U_X-L_X)T} \cdot \exp\left\{\int_0^T U_X - \phi(X_s)\ \mathrm{d}s\right\} \\ &= e^{-(L_X-\mathbf{\Phi})T} \cdot \left[\sum_{j=0}^{\infty} \frac{e^{-(U_X-L_X)T}[(U_X-L_X)T]^j}{j!} \left\{\int_0^T \frac{U_X-\phi(X_s)}{(U_X-L_X)T}\ \mathrm{d}s\right\}^j\right], \end{aligned} \tag{4.69}$$

and letting $\mathbb{K}$ denote the law of $\kappa \sim \mathrm{Poi}\big((U_X-L_X)T\big)$ and $\mathbb{U}$ be the distribution of $(\xi_1,\ldots,\xi_\kappa) \overset{\text{iid}}{\sim} \mathcal{U}[0,T]$ we have,

$$\begin{aligned} P_{\mathbb{Z}^x_{0,T}}(X) &= e^{-(L_X-\mathbf{\Phi})T} \cdot \mathbb{E}\left[\mathbb{E}\left[\mathbb{E}\left[\left(\int_0^T \frac{U_X-\phi(X_s)}{(U_X-L_X)T}\ \mathrm{d}s\right)^\kappa \middle| \{X,R\}\right] \middle| R\right]\right] \\ &= e^{-(L_X-\mathbf{\Phi})T} \cdot \mathbb{E}\left[\mathbb{E}\left[\mathbb{E}\left[\mathbb{E}\left[\prod_{i=1}^{\kappa}\left(\frac{U_X-\phi(X_{\xi_i})}{U_X-L_X}\right) \middle| \{\kappa,X,R\}\right] \middle| \{X,R\}\right] \middle| R\right]\right] \\ &= e^{-(L_X-\mathbf{\Phi})T} \cdot \mathbb{E}_{\mathcal{R}}\mathbb{E}_{\mathbb{Z}^x_{0,T}|\mathcal{R}}\mathbb{E}_{\mathbb{K}}\mathbb{E}_{\mathbb{U}}\left[\prod_{i=1}^{\kappa}\left(\frac{U_X-\phi(X_{\xi_i})}{U_X-L_X}\right)\right], \end{aligned} \tag{4.70}$$

where for readability, the subscripts for each expectation denotes the law with which we are taking the expectation. From (4.70), we can evaluate the acceptance probability of a sample path $X \sim \mathbb{Z}^x_{0,T}$ using just a finite dimensional realisation of the entire sample path. Simulating a finite dimensional proposal and evaluating the acceptance probability as suggested in (4.70) results in an implementable path-space rejection sampler for simulating sample paths $X \sim \mathbb{Q}^x_{0,T}$ and is presented in Algorithm 4.4.3. Further, note that since we know that for a given simulated layer information, $\phi(X_{[0,T]})$ is bounded in $[U_X, L_X]$, it follows that the unbiased estimator in (4.70) lies in $[0,1]$. Algorithm 4.4.3 summarises this argument and can be viewed as a two-step rejection sampler whereby the acceptance probability is broken into a computationally inexpensive step (Step 3) and a computationally expensive step (Step 5) which first requires the partial simulation of the proposed path in Step 4.

Algorithm 4.4.3 Path-space rejection sampling to simulate paths $X \sim \mathbb{Q}^x_{0,T}$ [Beskos et al., 2008; Pollock et al., 2016].

1. Simulate $X_T =: y \sim h$.
2. Simulate layer information $R \sim \mathcal{R}$ as per Algorithm 4.2.4.
3. With probability $(1 - \exp\{-(L_X - \mathbf{\Phi})T\})$ reject path and return to Step 1.
4. Simulate skeleton points $(X_{\xi_1}, \ldots, X_{\xi_\kappa})|R$:
   (a) Simulate $\kappa \sim \text{Poi}\big((U_X - L_X)T\big)$ and skeleton times $\xi_1, \ldots, x_\kappa \overset{\text{iid}}{\sim} \mathbb{U}[0,T]$.
   (b) Simulate sample path at skeleton times $X_{\xi_1}, \ldots, X_{\xi_\kappa} \sim \mathbb{W}^{x,y}_{0,T}|R$ as per Algorithm 4.2.5.
5. Return entire skeleton $\mathcal{S}(X) := \{X_0, \{X_{\xi_i}\}_{i=1}^\kappa, X_T, R\}$ with probability

$$\prod_{i=1}^{\kappa} \left(\frac{U_X - \phi(X_{\xi_i})}{U_X - L_X} \right),$$

else reject and return to Step 1.

The skeleton that is (potentially) returned in Step 5 includes the layer information which is necessary in any subsequent simulation. If we wish to simulate the sample path at further intermediate points, then we require additional computation by augmenting the skeleton with sub-interval layer information,

$$\begin{aligned} \mathcal{S}'(X) &:= \left\{ (\xi_i, X_{\xi_i})_{i=1}^\kappa, R, \left(R_X^{[\xi_{i-1}, \xi_i]} \right)_{i=1}^\kappa \right\} \\ &= \left\{ (\xi_i, X_{\xi_i})_{i=1}^\kappa, \left(R_X^{[\xi_{i-1}, \xi_i]} \right)_{i=1}^\kappa \right\}, \end{aligned} \tag{4.71}$$

where $R_X^{[a,b]}$ denotes the layer information for the sub-interval $[a,b] \subset [0,T]$. This resulting skeleton is thus decomposed into conditionally independent paths between each of the skeletal points. We can then simulate the sample path at further times after the acceptance of the path directly,

$$X^{\text{rem}} \sim \mathbb{W}^{x,y}_{0,T} \Big| \mathcal{S}'(X) = \otimes_{i=1}^\kappa \left(\mathbb{W}^{X_{\xi_{i-1}}, X_{\xi_i}}_{\xi_{i-1}, \xi_i} \Big| R_X^{[\xi_{i-1}, \xi_i]} \right). \tag{4.72}$$

In the case that $\phi(X_{[0,T]})$ is almost surely bounded, then it is not necessary to simulate layer information in Algorithm 4.4.3 and hence the skeleton can be simulated from the law of a Brownian bridge and any subsequent simulation can be done by sampling from the law of a Brownian bridge.

The construction of the unbiased estimators in this section are crucial in the development of the Fusion methodology discussed in this thesis. However, there are some settings where we do not necessarily require our unbiased estimators to be bounded in $[0,1]$, for instance, in the case of implementing an importance sampler (see Section 2.3) or a sequential Monte Carlo algorithm (see Chapter 3). In those settings, it is more desirable to have estimators which have lower variance. In Section 4.4.3, we look at a class of unbiased estimators for $\exp\left\{ \int_s^t \phi(X_u)\, \mathrm{d}u \right\}$.

4.4.3 Poisson Estimators

We now restrict our attention to simulating positive unbiased estimators for functions of the form

$$\psi(X) := \exp\left\{-\int_s^t \phi(X_u)\ \mathrm{d}u\right\}, \qquad \text{where } X \sim \mathbb{W}_{s,t}^{x,y}, \tag{4.73}$$

which only require a finite dimensional subset of the proposal path. Since we will use these estimators within a rejection sampling or an importance sampling context, then we require that the estimators we construct to be positive (and bounded in the case of rejection sampling). If used within an importance sampling setting, it is natural to favour estimators with lower variance. In this section, we review a class of unbiased estimators for (4.73) proposed by Beskos et al. [2006a], Fearnhead et al. [2008] and Fearnhead et al. [2010] named *Poisson Estimators*. The (Vanilla) Poisson Estimator [Beskos et al., 2006a] provided an unbiased estimator for (4.73), however, it is not necessarily positive and does not necessarily have finite variance, so we focus on reviewing the positive and finite variance *Generalised Poisson Estimators* first introduced by Fearnhead et al. [2008]. Details of such estimators have also been summarised in a number of texts, for instance Dai et al. [2021, Appendix B] and Pollock [2013, Chapter 7].

As noted in Section 4.4.2, conditional on some *layer information* of a Brownian bridge path, upper and lower bounds on $\phi(X)$ can be found, denoted U_X and L_X, respectively. In particular, we can partition the path-space of $\mathbb{W}_{s,t}^{x,y}$ into disjoint layers and simulate the layer to which a proposal sample path belongs (see Section 4.2). Denoting $R := R(X) \sim \mathcal{R}$ as the simulated layer, we can focus on finding unbiased estimators for functions of the form

$$\psi(X|R) := \exp\left\{-\int_s^t \phi(X_u)\ \mathrm{d}u\right\}, \qquad \text{where } X \sim \mathbb{W}_{s,t}^{x,y}|R. \tag{4.74}$$

Following a similar approach to Section 4.4.2, Fearnhead et al. [2008, Section 4] considers a Taylor series expansion of the exponential function in (4.74),

$$\begin{aligned}
\psi(X|R) &= \exp\left\{-\int_s^t \phi(X_u)\ \mathrm{d}u\right\} \\
&= e^{-U_X(t-s)} \cdot \exp\left\{\int_s^t U_X - \phi(X_u)\ \mathrm{d}u\right\} \\
&= e^{-U_X(t-s)} \cdot \left[\sum_{j=0}^{\infty} \frac{(t-s)^j}{j!} \left\{\int_s^t \frac{U_X - \phi(X_u)}{t-s}\ \mathrm{d}u\right\}^j\right],
\end{aligned} \tag{4.75}$$

then by denoting $\mathbb{K}$ to be the law of the discrete random variable $\kappa = 0, 1, \ldots,$ with probability mass function $p(\kappa|R)$, and $\mathbb{U}$ to be the distribution of $(\xi_1, \ldots, \xi_\kappa) \sim \mathcal{U}[s,t]$, then

$$\psi(X|R) = e^{-U_X(t-s)} \sum_{j=0}^{\infty} \left[\frac{(t-s)^j}{j!} \left\{\int_s^t \frac{U_X - \phi(X_u)}{t-s}\ \mathrm{d}u\right\}^j\right]$$

$$= \mathbb{E}_{\mathbb{K}}\left[e^{U_X(t-s)} \cdot \frac{(t-s)^\kappa}{\kappa! \cdot p(\kappa|R)} \left\{\int_s^t \frac{U_X - \phi(X_u)}{t-s}\ \mathrm{d}u\right\}^\kappa \middle| X\right]$$
$$= \mathbb{E}_{\mathbb{K}}\mathbb{E}_{\mathbb{U}_\kappa}\left[e^{U_X(t-s)} \cdot \frac{(t-s)^\kappa}{\kappa! \cdot p(\kappa|R)} \prod_{i=1}^{\kappa} [U_X - \phi(X_{\xi_i})] \middle| X\right] \tag{4.76}$$

By uniformly scattering $\kappa \sim p(\kappa|R)$ points in $[s,t]$, we can simulate a sample path $X \sim \mathbb{W}_{s,t}^{x,y}|R$ at these points and define the *Generalised Poisson Estimator (GPE)* [Fearnhead et al., 2008]:

$$\hat{\psi}(X|R) := e^{U_X(t-s)} \cdot \frac{(t-s)^\kappa}{\kappa! \cdot p(\kappa|R)} \cdot \prod_{i=1}^{\kappa} [U_X - \phi(X_{\xi_i})] \tag{4.77}$$

The algorithm for simulating unbiased estimators for (4.74) is summarised in Algorithm 4.4.4.

Algorithm 4.4.4 Generalised Poisson Estimator (GPE) for unbiasedly estimating $\psi(X)$ in (4.73) [Fearnhead et al., 2008].

1. Simulate $R \sim \mathcal{R}$ as per Algorithm 4.2.4.
2. Simulate $\kappa \sim p(\kappa|R)$.
3. Simulate skeleton times $\xi_1, \ldots, \xi_\kappa \sim \mathcal{U}[s,t]$.
4. Simulate sample path at skeleton times $X_{\xi_1}, \ldots, X_{\xi_\kappa} \sim \mathbb{W}_{s,t}^{x,y}|R$ as per Algorithm 4.2.5.
5. Evaluate and return (4.77).

show that the variance of the unbiased estimator $a_j\tilde{\rho}_j$ is minimised when $p(\kappa_c|R_c) \sim \mathrm{Poi}(\lambda_c)$, where

Fearnhead et al. [2008] notes that we can derive various estimators for (4.73) by specifying different discrete probability mass functions for κ, $p(\kappa|R)$. The family of these estimators are referred to as *Generalised Poisson Estimators (GPEs)*. Fearnhead et al. [2008, Theorem 1] showed that the optimal proposal (in the sense of lowest variance) is obtained by selecting $p(\kappa_c|R_c) \sim \mathrm{Poi}(\lambda_c)$, where

$$\lambda_{opt} := \left((t-s)\int_s^t [U_X - \phi(X_u)]^2\ \mathrm{d}u\right)^{\frac{1}{2}}. \tag{4.78}$$

Whilst this cannot be evaluated analytically, Fearnhead et al. [2008] uses this to obtain two estimators. A natural choice is to choose a Poisson distribution with intensity λ which bounds λ_{opt},

$$\left((t-s)\int_s^t [U_X - \phi(X_u)]^2\ \mathrm{d}u\right)^{\frac{1}{2}} \leq \left((t-s)\int_s^t [U_X - L_X]^2\ \mathrm{d}u\right)^{\frac{1}{2}}$$
$$= (U_X - L_X)\cdot(t-s). \tag{4.79}$$

By choosing $p(\kappa|R)$ to be the Poisson with mean $\lambda = (U_X - L_X)\cdot(t-s)$, then we obtain the *GPE-1* unbiased estimator which is positive and defined as,

$$\hat{\psi}_{\text{GPE-1}}(X) = e^{U_X(t-s)} \cdot \frac{(t-s)^\kappa}{\kappa! \cdot p(\kappa|R)} \cdot \prod_{i=1}^{\kappa} [U_X - \phi(X_{\xi_i})]$$

$$:= e^{-L_X(t-s)} \cdot \prod_{i=1}^{\kappa} \frac{U_X - \phi(X_{\xi_i})}{U_X - L_X} \in \left[0, e^{-L_X(t-s)}\right], \tag{4.80}$$

with second moment bounded above by $\mathbb{E}\left[e^{-2L_X(t-s)}\right] < \infty$ [Fearnhead et al., 2008]. This is the estimator that was constructed in Section 4.4.2 which has the benefit of being bounded so can be used in a rejection sampling setting.

An alternative choice proposed by Fearnhead et al. [2008] was to choose a heavier tailed negative binomial distribution parameterised with mean γ, set to be equal to an approximation of the optimal intensity λ_{opt},

$$\gamma := U_X(t-s) - \int_s^t \phi\left(x \cdot \frac{t-u}{t-s} + y \cdot \frac{u-s}{t-s}\right) \, \mathrm{d}u, \tag{4.81}$$

and dispersion parameter β. This results in the *GPE-2* unbiased estimator, given by

$$\begin{aligned} \hat{\psi}_{\text{GPE-2}}(X) &= e^{U_X(t-s)} \cdot \frac{(t-s)^{\kappa}}{\kappa! \cdot p(\kappa|R)} \cdot \prod_{i=1}^{\kappa} \left[U_X - \phi(X_{\xi_i})\right] \\ &:= e^{-U_X(t-s)} \cdot \frac{(t-s)^{\kappa} \cdot \Gamma(\beta) \cdot (\gamma+\beta)^{\gamma+\kappa}}{\Gamma(\beta+\kappa) \cdot \gamma^{\kappa} \cdot \beta^{\beta}} \cdot \prod_{i=1}^{\kappa} \left[U_X - \phi(X_{\xi_i})\right]. \end{aligned} \tag{4.82}$$

Fearnhead et al. [2008] noted that the GPE-2 estimator has finite variance and showed empirically with simulations that GPE-2 typically outperforms GPE-1, and in some simulations had up to several orders of magnitude smaller variance than GPE-1. Further, they showed that GPE-2 is quite robust to the choice of the dispersion parameter β. However, since (4.82) is unbounded, it cannot be used for a rejection sampling scheme directly, but is the favourable choice in any importance sampling or sequential Monte Carlo scheme.

Chapter 5

Fusion methodologies

In this thesis, we focus on developing methodology for an *exact* Monte Carlo approximation of the fusion density, $f(\boldsymbol{x}) \propto \prod_{c=1}^{C} f_c(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^d$ and f_c denote the *sub-posterior* densities that we wish to unify. As discussed in Chapter 1, the majority of existing approaches have relied upon *approximations* of the sub-posterior densities. Whilst these methodologies are typically computationally efficient, the quality of the posterior approximation for these approaches is poor when the sub-posteriors fall out-with a narrow range of distributional form. In contrast, the *Fusion* approach to dealing with this problem is to construct an exact sample approximation of f, as opposed to approximating f by combining approximations of the sub-posteriors.

In this chapter, we review the two existing Fusion methodologies. We start this chapter by introducing the *Monte Carlo Fusion* (MCF) approach of Dai et al. [2019] in Section 5.1, which was the first general approach to the fusion problem that avoided any approximation error in sampling from the fusion density (1.1). MCF provides a theoretical framework to sample independent draws from (1.1) exactly, and achieves this by constructing a two-step rejection sampler (see Section 2.2) on an extended state space. Within Dai et al. [2019], the authors introduce two Monte Carlo Fusion approaches, one based on *Brownian bridges (BB)* and one based on *Ornstein-Uhlenbeck (OU)* bridges. We begin this section by describing the BB approach in Section 5.1.1. In Part II, we do not develop the OU approach further and hence we only provide a short discussion of the OU approach and its differences to the BB approach in Section 5.1.2 and refer the reader to Dai et al. [2019, Section 4] for further details. *Bayesian Fusion* (BF) [Dai et al., 2021] is an alternative sequential Monte Carlo approach (see Chapter 3) developed to help mitigate some of the computational challenges faced with the MCF approach. We provide an overview of BF in Section 5.2.

In this chapter, several results from Dai et al. [2019] and Dai et al. [2021] are stated without proof since in Chapter 6 and Chapter 7, we will generalise both of these approaches and provide results and algorithms which will admit these two methods as special cases.

5.1 Monte Carlo Fusion

The Fusion methodologies are based on the simple intuition that if we have independent random variables $\boldsymbol{x}^{(c)} \sim f_c$, for $c = 1, \ldots, C$, with joint density $\prod_{c=1}^{C} f_c\left(\boldsymbol{x}^{(c)}\right)$, then if we condition on the event $\left(\boldsymbol{x} := \boldsymbol{x}^{(1)} = \cdots = \boldsymbol{x}^{(c)}\right)$, then we have $\boldsymbol{x} \sim f$, where f is the fusion density defined in (1.1). This argument is summarised in Algorithm 5.1.1 to obtain N random samples from f.

Algorithm 5.1.1 Perfect Fusion algorithm for generating N random samples from (1.1).

1. For i in 1 to N,
   (a) For $c = 1, \ldots, C$, simulate $\boldsymbol{x}^{(1)} \sim f_1, \boldsymbol{x}^{(2)} \sim f_2, \ldots, \boldsymbol{x}^{(c)} \sim f_c$.
   (b) If $\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(2)} = \cdots = \boldsymbol{x}^{(c)}$, set $\boldsymbol{x}_i = \boldsymbol{x}^{(1)}$, else return to Step 1a.
2. Return samples $\{\boldsymbol{x}_i\}_{i=1}^{N}$.

However, the conditioning event of the sub-posterior samples $\boldsymbol{x}^{(c)} \sim f_c$ being equal for all $c = 1, \ldots, C$, has probability zero if any of the sub-posteriors are continuous densities, so Algorithm 5.1.1 is not practical for simulating from (1.1). However, we could consider simulating C *independent* stochastic processes with respective invariant densities f_c for $c = 1, \ldots, C$. The time point for which the stochastic processes coincide would give us a sample from (1.1) (illustrated in Figure 5.1a for $C = 2$). However, for $C > 2$, these events would be too rare and again have probability zero. To circumvent this, we can construct a setting whereby we can *force* the C stochastic processes to coalesce at a certain time (illustrated in Figure 5.1b). With this approach, however, it is unclear how we propose the end point of these stochastic processes. Monte Carlo Fusion (MCF) [Dai et al., 2019] is an implementable algorithm which provides a coalescence of C stochastic processes at a fixed time $T > 0$, and consequently the distribution of this common value at the time T is distributed according to f in (1.1). The key idea of MCF is given by the following proposition.

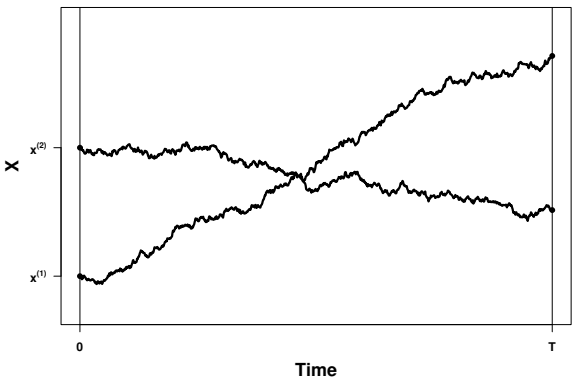
Proposition 5.1.1. *[Dai et al., 2021, Proposition 1] Suppose that p_c is the transition density of a Markov chain on $\mathbb{R}^d$ with a stationary probability density proportional to f_c^2. Then the $(C+1)d$-dimensional probability density proportional to the integrable function*

$$g\left(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y}\right) := \prod_{c=1}^{C} \left[f_c^2\left(\boldsymbol{x}^{(c)}\right) \cdot p_c\left(\boldsymbol{y} \middle| \boldsymbol{x}^{(c)}\right) \cdot \frac{1}{f_c\left(\boldsymbol{y}\right)} \right], \tag{5.1}$$

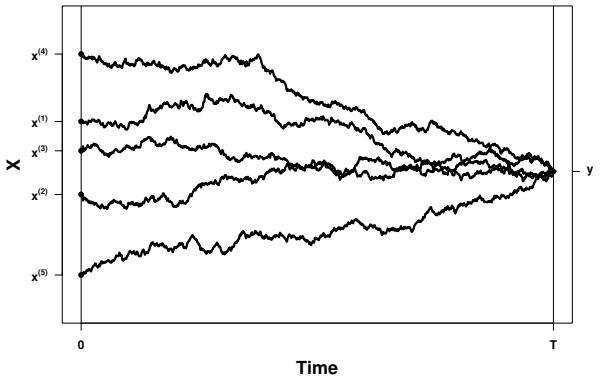
admits marginal density f for $\boldsymbol{y}$.

Proof. By integrating out $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}$, we have

$$\begin{aligned}
&\int_{\mathbb{R}_d} \cdots \int_{\mathbb{R}_d} g\left(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y}\right) \, \mathrm{d}\boldsymbol{x}^{(1)} \cdots \, \mathrm{d}\boldsymbol{x}^{(C)} \\
&\quad = \int_{\mathbb{R}_d} f_1^2\left(\boldsymbol{x}^{(1)}\right) \cdot p_1\left(\boldsymbol{y} \middle| \boldsymbol{x}^{(1)}\right) \cdot \frac{1}{f_1\left(\boldsymbol{y}\right)} \, \mathrm{d}\boldsymbol{x}^{(1)} \cdots \int_{\mathbb{R}_d} f_C^2\left(\boldsymbol{x}^{(C)}\right) \cdot p_C\left(\boldsymbol{y} \middle| \boldsymbol{x}^{(C)}\right) \cdot \frac{1}{f_C\left(\boldsymbol{y}\right)} \, \mathrm{d}\boldsymbol{x}^{(C)} \\
&\quad \propto \frac{f_1^2(\boldsymbol{y})}{f_1(\boldsymbol{y})} \cdots \frac{f_C^2(\boldsymbol{y})}{f_C(\boldsymbol{y})}
\end{aligned}$$



(a) Illustration of $C = 2$ independent stochastic processes with invariant density proportional to f_c and initialised at $\boldsymbol{x}^{(c)} \sim f_c$ for $c = 1, 2$. The value where these processes coincide (cross-over) is distributed according to $\prod_{c=1}^{C} f_c$.



(b) Illustration of $C = 5$ independent stochastic processes with invariant distribution proportional to f_c and initialised at $\boldsymbol{x}^{(c)} \sim f_c$ for $c = 1, \ldots, 5$, which coalesce at $\boldsymbol{y}$ at time T. The common value at time T is distributed according to $\prod_{c=1}^{C} f_c$.

Figure 5.1: Illustrative plots for intuition for Fusion methodology.

$$= f_1(\boldsymbol{y}) \cdots f_C(\boldsymbol{y}) = f(\boldsymbol{y}). \tag{5.2}$$

Hence, $\boldsymbol{y}$ has marginal density f. ■

The key takeaway from Proposition 5.1.1 is that if it were possible to sample from the $(C+1)d$-dimensional extended target density g in (5.1), then we could obtain a draw from the fusion target density f by simply taking the marginal samples $\boldsymbol{y}$.

In the subsequent sections (and in Proposition 5.1.2) we will a consider Langevin diffusion with invariant measure f_c^2 which we can sample from exactly using methodology from Chapter 4. The transition density for such diffusion can be expressed using the Dacunha-Castelle representation [Dacunha-Castelle and Florens-Zmirou, 1986] discussed in Section 4.3.4 which would include a $\frac{f_c(\boldsymbol{y})}{f_c(\boldsymbol{x}^{(c)})}$ term and subsequently result in a cancelling of terms (see Proposition 6.1.2). From there, it is easy to construct a proposal distribution, h, which is easy to sample from that results in a bounded ratio of g/h to form a valid rejection sampler. However, although we do not explore it here in this thesis, note that there may be variations of g which could lead to other valid Fusion methodologies such as letting $p(\boldsymbol{y}|\boldsymbol{x}^{(c)})$ be a f_c invariant Markov process and letting $g^{alt}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y}) := \prod_{c=1}^{C} f_c(\boldsymbol{x}^{(c)}) p_c(\boldsymbol{y}|\boldsymbol{x}^{(c)})$. Following a similar approach to Proposition 5.1.1, we see that g^{alt} also admits marginal density f for $\boldsymbol{y}$.

5.1.1 Brownian bridge approach

Let $p_c(\boldsymbol{y}|\boldsymbol{x}) = p_{T,c}^{dl}(\boldsymbol{y}|\boldsymbol{x})$, the transition density of a *double Langevin diffusion* for f_c (i.e. a Langevin diffusion with invariant distribution f_c^2) from $\boldsymbol{x}$ to $\boldsymbol{y}$ over a pre-determined, user-specified, time

$T > 0$. In particular, $p^{dl}_{T,c}(\boldsymbol{y}|\boldsymbol{x})$ is the transition density of the d-dimensional (double) Langevin (DL) diffusion processes $\boldsymbol{X}^{(c)}_t$ for $c = 1, \ldots, C$, from $\boldsymbol{x}$ to $\boldsymbol{y}$ for time $T > 0$ given by

$$\mathrm{d}\boldsymbol{X}^{(c)}_t = \nabla \log f_c\left(\boldsymbol{X}^{(c)}_t\right)\, \mathrm{d}t + \mathrm{d}\boldsymbol{W}^{(c)}_t, \quad \boldsymbol{X}^{(c)}_0 = \boldsymbol{x}^{(c)},\ t \in [0, T], \tag{5.3}$$

where $\boldsymbol{W}^{(c)}_t$ is the d-dimensional Brownian motion and ∇ is the gradient operator over $\boldsymbol{x}$. In this case, the $(C+1)d$-dimensional fusion density is given by

$$g^{dl}\left(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y}\right) := \prod_{c=1}^{C}\left[f^2_c\left(\boldsymbol{x}^{(c)}\right) \cdot p^{dl}_{T,c}\left(\boldsymbol{y}\middle|\boldsymbol{x}^{(c)}\right) \cdot \frac{1}{f_c(\boldsymbol{y})}\right]. \tag{5.4}$$

To sample from g^{dl}, Dai et al. [2019, Section 3] construct a rejection sampler for the extended target density, and consider the following proposal density which is proportional to the function,

$$h^{bm}\left(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y}\right) := \prod_{c=1}^{C}\left[f_c\left(\boldsymbol{x}^{(c)}\right)\right] \cdot \exp\left\{-\frac{C \cdot \|\boldsymbol{y} - \bar{\boldsymbol{x}}\|^2}{2T}\right\}, \tag{5.5}$$

where $\bar{\boldsymbol{x}} = \frac{1}{C}\sum_{c=1}^{C} \boldsymbol{x}^{(c)}$ and $T > 0$. Simulation from the proposal h^{bm} can be achieved directly:

1. simulate $\boldsymbol{x}^{(c)} \sim f_c$ independently for $c = 1, \ldots, C$,
2. simulate $\boldsymbol{y} \sim \mathcal{N}_d\left(\bar{\boldsymbol{x}}, \frac{T}{C}\mathbb{I}_d\right)$, where $\mathbb{I}_d$ is the identity matrix of dimension d.

Before obtaining an expression for g^{dl}/h^{bm}, which gives us the acceptance probability of a rejection sampling scheme, Dai et al. [2019] imposes the following standard regularity conditions.

Condition 5.1.1. $\nabla \log f_c(\boldsymbol{x})$ *is at least once continuously differentiable.*

Condition 5.1.2. *Define*

$$\phi^{dl}_c(\boldsymbol{x}) := \frac{1}{2}\left(\|\nabla \log f_c(\boldsymbol{x})\|^2 + \Delta \log f_c(\boldsymbol{x})\right), \tag{5.6}$$

where ∇ is the gradient operator over $\boldsymbol{x}$ and Δ is the Laplacian operator, then there exists a constant $\boldsymbol{\Phi}^{bm}_c > -\infty$ such that for all $\boldsymbol{x}$ and each $c = 1, \ldots, C$, $\phi^{dl}_c(\boldsymbol{x}) \geq \boldsymbol{\Phi}^{bm}_c$.

Then we have the following proposition that gives the acceptance probability for a rejection sampling algorithm for g^{dl} (5.4) with a proposal given by h^{bm} in (5.5).

Proposition 5.1.2. *[Dai et al., 2019, Proposition 2] Under Condition 5.1.1 and Condition 5.1.2, we can write*

$$\frac{g^{dl}\left(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y}\right)}{h^{bm}\left(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y}\right)} = \left[\frac{\sqrt{C}}{\sqrt{2\pi T}}\right]^C \cdot \rho^{bm} \cdot Q^{bm} \cdot \prod_{c=1}^{C} e^{-T\boldsymbol{\Phi}^{bm}_c}, \tag{5.7}$$

where

$$\rho^{bm}\left(\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(C)}\right)=e^{-\frac{C\sigma^2}{2T}},\ \textit{with}\ \sigma^2=\frac{1}{C}\sum_{c=1}^{C}\left\|\bar{\boldsymbol{x}}-\boldsymbol{x}^{(c)}\right\|^2, \tag{5.8}$$

and

$$Q^{bm}\left(\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(C)},\boldsymbol{y}\right)=\prod_{c=1}^{C}\mathbb{E}_{\mathbb{W}_c}\left[\exp\left\{-\int_0^T\left(\phi_c^{dl}\left(\boldsymbol{X}_t^{(c)}\right)-\boldsymbol{\Phi}_c^{bm}\right)\,\mathrm{d}t\right\}\right], \tag{5.9}$$

where $\mathbb{W}_c$ *denotes the law of a Brownian bridge* $\{\boldsymbol{X}_t^{(c)},t\in[0,T]\}$ *with* $\boldsymbol{X}_0^{(c)}:=\boldsymbol{x}^{(c)}$ *and* $\boldsymbol{X}_T^{(c)}:=\boldsymbol{y}$ *in the time interval* $[0,T]$.

Proof. See Dai et al. [2019, Proposition 2]. This also follows directly from Proposition 6.1.2 by setting $\boldsymbol{\Lambda}_c=\mathbb{I}_d$ for all $c\in\mathcal{C}:=\{1,\ldots,C\}$. ■

Both ρ^{bm} and Q^{bm} are bounded by 1, and correspond to separate acceptance steps within a rejection sampling framework. An event of probability ρ^{bm} can be simulated by directly computing (5.8) using the sub-posterior samples. Although Q^{bm} in (5.9) cannot be evaluated directly, we can construct bounded unbiased estimators for Q^{bm}, denoted $\hat{Q}^{bm}$. an event of probability Q^{bm} can be simulated using an auxiliary diffusion bridge path-space rejection sampler since Q^{bm} is the product of C path-space rejection sampling acceptance probabilities when simulating sample path skeletons from (5.3) via Brownian bridge proposals (see for instance the similarity to (4.68)). Recall in Section 4.4.2, we constructed an unbiased estimator for the acceptance probability in (4.68), but the key difference is that we do not need to use Biased Brownian Motion, $\mathbb{Z}_{0,T}^x$ (see Definition 4.4.1), since the common end point of the C paths, $\boldsymbol{y}$, is fixed here and is proposed from $\mathcal{N}_d\left(\bar{\boldsymbol{x}},\frac{T}{C}\mathbb{I}_d\right)$.

Recall from Section 4.4, in order to find an unbiased estimator for Q^{bm} which is positive and bounded (which is necessary for a rejection sampler), we require for a given sample path $\boldsymbol{X}_{[0,T]}^{(c)}\sim\mathbb{W}_c$, that upper and lower bounds for $\phi_c(\boldsymbol{X}_{[0,T]}^{(c)})$, denoted $U_X^{(c)}$ and $L_X^{(c)}$, are available. Typically, we cannot find global bounds for ϕ_c, and so following the approach outlined in Section 4.4.2, we can use layered Brownian bridge constructions (see Section 4.2.2) to partition the space of $\mathbb{W}_c$ into disjoint layers. Let $R_c:=R_c(\boldsymbol{X}_{[0,T]}^{(c)})$ define the compact subset of $\mathbb{R}^d$ for which $\boldsymbol{X}_t^{(c)}$ is constrained in time $t\in[0,T]$, then we simulate a layer $R_c\sim\mathcal{R}_c$ (as per Algorithm 4.2.4) which constrains a given path. This allows us to find *local* bounds for $\phi_c^{dl}(\boldsymbol{X}_{[0,T]}^{(c)})$. An interpretation of the Q^{bm} accept/reject step is that we are driving the sub-posterior samples $\{\boldsymbol{x}^{(c)}\}_{c=1,\ldots,C}$ to the proposed point $\boldsymbol{y}$ using auxiliary Brownian bridge proposals, and if all of these bridges are accepted as a sample path from the target diffusion in (5.3) for each $c=1,\ldots,C$, then we accept this step. To summarise, let

$$\hat{Q}^{bm}\left(\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(C)},\boldsymbol{y}\right)=\prod_{c=1}^{C}\left(e^{-\left(L_X^{(c)}-\boldsymbol{\Phi}_c^{bm}\right)T}\cdot\prod_{k_c=1}^{\kappa_c}\left[\frac{U_X^{(c)}-\phi_c^{dl}(\boldsymbol{X}_{\xi_c,k_c})}{U_X^{(c)}-L_X^{(c)}}\right]\right), \tag{5.10}$$

then we have

$$Q^{bm} = \mathbb{E}_{\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{W}}|\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{K}}}\mathbb{E}_{\bar{\mathbb{U}}}\left[Q^{\hat{bm}}\right],$$

where for readability, the expectation subscripts denotes the law with which we are taking. $\bar{\mathcal{R}}$ denotes the law of $\{R_c \sim \mathcal{R}_c : c = 1, \ldots, C\}$, $\bar{\mathbb{W}}$ denotes the law of $\{\mathbb{W}_c : c = 1, \ldots, C\}$, $\bar{\mathbb{K}}$ denotes the law of $\{\kappa_c : c = 1, \ldots, C\}$ with $\kappa_c \sim \text{Poi}\big((U_X^{(c)} - L_X^{(c)})T\big)$ and $\bar{\mathbb{U}}$ denotes the law of $\{\xi_{c,1}, \ldots, \xi_{c,\kappa_c} : c = 1, \ldots, C\} \overset{\text{iid}}{\sim} \mathcal{U}[0, T]$. We can simulate $\hat{Q}^{bm}$ as per Algorithm 5.1.2.

Algorithm 5.1.2 Simulating the unbiased estimator for Q^{bm} (5.9).

1. For $c = 1, \ldots, C$,
   (a) R_c: Simulate $R_c \sim \mathcal{R}$ as per Algorithm 4.2.4.
   (b) $L_X^{(c)}, U_X^{(c)}$: Compute lower and upper bounds, $L_X^{(c)}$ and $U_X^{(c)}$, of $\phi_c^{dl}(\boldsymbol{x})$ for $\boldsymbol{x} \in R_c$.
   (c) p_c: Choose $p(\cdot|R_c)$ to be Poisson distributed with mean $\lambda_c = \left(U_X^{(c)} - L_X^{(c)}\right)T$.
   (d) κ_c, ξ: Simulate $\kappa_c \sim p(\cdot|R_c)$, and simulate $\xi_{c,1}, \ldots, \xi_{c,\kappa_c} \sim \mathcal{U}[0, T]$.
   (e) $\boldsymbol{X}^{(c)}$: Simulate $\boldsymbol{X}_{\xi_{c,1}}^{(c)}, \ldots, \boldsymbol{X}_{\xi_{c,\kappa_c}}^{(c)} \sim \mathbb{W}_c|R_c$ as per Algorithm 4.2.5.
2. Output $\hat{Q}^{bm}$ (5.10).

The fundamental idea behind MCF is that we can sample from f (1.1) by means of constructing a rejection sampling for an extended target density g which admits f as a marginal. To summarise, we can propose samples from h^{bm}, where the proposed value $\boldsymbol{y}$ is generated from a Gaussian distribution with mean $\bar{\boldsymbol{x}}$ and covariance matrix $\frac{T}{C}\mathbb{I}_d$. Using a rejection sampling approach, we then accept the value of $\boldsymbol{y}$ as a sample from f with probability $\frac{g^{dl}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y})}{h^{bm}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y})} \propto \rho^{bm} \cdot Q^{bm}$. Viewing this approach as a two-step rejection sampler is advantageous since the computational cost of simulating the diffusion bridge rejection sampling in Q^{bm} is comparatively more expensive than computing ρ^{bm}. Hence if a sample is rejected after an accept/reject step with ρ^{bm}, we can avoid unnecessary computation of the unbiased estimator for Q^{bm}. Dai et al. [2019] points out that the algorithm indicates exactly how the simple average of these independent sub-posterior samples, $\bar{\boldsymbol{x}}$, can be corrected to obtain a draw from f (1.1). This adjustment comes in the form of the accept/reject step with acceptance probability $\rho^{bm} \cdot Q^{bm}$. The Monte Carlo Fusion approach (with Brownian bridge proposals) for sampling from f is presented in Algorithm 5.1.3.

Algorithm 5.1.3 Monte Carlo Fusion (Brownian bridge approach) [Dai et al., 2019, Algorithm 1].

1. For i in 1 to N,
   (a) For $c = 1, \ldots, C$, simulate $\boldsymbol{x}^{(c)} \sim f_c(\boldsymbol{x})$.
   (b) Compute ρ^{bm} as per (5.8).
   (c) Generate standard uniform random variable U_1.
   (d) If $U_1 \leq \rho^{bm}$,
      i. Simulate from $\boldsymbol{y} \sim \mathcal{N}_d\left(\bar{\boldsymbol{x}}, \frac{T}{C}\mathbb{I}_d\right)$.
      ii. Compute $\hat{Q}^{bm}$ as per Algorithm 5.1.2.
      iii. Generate standard uniform random variable U_2.
      iv. If $U_2 \leq \hat{Q}^{bm}$, set $\boldsymbol{Y}_i = \boldsymbol{y}$, else return to Step 1a.
2. Return samples $\{\boldsymbol{Y}_i\}_{i=1}^N$.

Dai et al. [2019] remarks that for small T, ρ^{bm} will likely be small, while Q^{bm} is large, and for larger T, the opposite will be true. In practice, a small value of T is generally preferred since the computational cost for simulating unbiased estimators for Q^{bm} is comparatively more expensive. Further, note that $\sigma^2 = \frac{1}{C}\sum_{c=1}^{C} \left\|\boldsymbol{x}^{(c)} - \bar{\boldsymbol{x}}\right\|^2$ is the *sample variance* of the simulated starting points $\boldsymbol{x}^{(c)}$. This implies that $\boldsymbol{y}$ has a reasonable probability of acceptance as a sample of f only when the variance of the simulated sub-posterior realisations, $\boldsymbol{x}^{(c)}$ for $c = 1, \ldots, C$, is small enough.

Recalling the intuition of the Fusion approaches at the start of this section, consider the case where $T = 0$, then the event $U_2 \leq Q^{bm}$ will definitely occur, but the event $U_1 \leq \rho^{bm}$ only occurs if $\boldsymbol{x}^{(1)} = \cdots = \boldsymbol{x}^{(c)} = \boldsymbol{y}$. Further, the proposal in this setting is

$$h^{bm}\left(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(C)}, \boldsymbol{y}\right) := \prod_{c=1}^{C} f_c\left(\boldsymbol{x}^{(c)}\right).$$

MCF with $T = 0$ is therefore equivalent to the idealised Fusion algorithm in Algorithm 5.1.1. As noted at the start of this section, in practice, we have to choose $T > 0$, since the independent sub-posterior samples $\boldsymbol{x}^{(c)}$ have zero probability of coinciding with each other if any of them are continuous, and generally have low probability if they are all discrete probability mass functions.

5.1.2 Ornstein-Uhlenbeck bridges approach

The proposal density h^{bm} in (5.5) uses a simple average of the sub-posterior samples as the mean of the proposal of $\boldsymbol{y}$. To obtain a more tailored proposal for g^{dl} (5.4), we could alternatively consider a *weighted average* of the sub-posterior samples which incorporates global information for each sub-posterior, which can often be obtained through sub-posterior mean and covariance estimates. The methodology that we later develop in Chapter 6 does this by having the flexibility to weight the contribution from each sub-posterior through a user-specified matrix $\boldsymbol{\Lambda}_c$ for $c = 1, \ldots, C$. Alternatively, Dai et al. [2019, Section 4] considers an underlying *Ornstein-Uhlenbeck (OU)* proposal measure, which is approximately parameterised by obtaining estimates of the mean and covariance of the sub-posteriors. The motivation is that if the sub-posterior densities were approximately Gaussian, which is often a reasonable assumption in large data settings [Le Cam, 1986; Scott et al., 2016; Dai et al., 2019], the proposal measure will be more closely matched to the target which would lead to a more efficient algorithm. The approximations to parameterise the OU process does not introduce any bias to the samples obtained for f.

As with the Brownian bridge approach described in Section 5.1.1, the OU approach is a rejection sampling scheme for g^{dl}. Since we will not develop this particular approach further, we will not provide more details on the OU approach as this would require details on constructing an unbiased estimator for an intractable acceptance probability, denoted Q^{ou} (in a similar fashion to unbiasedly estimating Q^{bm}) that involves simulating OU bridges which are not covered in this thesis.

5.1.3 Illustrative toy examples

In this section, we implement the MCF methodology of Dai et al. [2019] and apply it to simple toy examples for illustrative purposes. With each example, we contrast our implementation of MCF against the *Consensus Monte Carlo (CMC)* [Scott et al., 2016], the kernel density averaging approach of Neiswanger et al. [2014] (which we term *KDEMC* in this thesis), and the *Weierstrass Sampler (WRS)* method [Wang and Dunson, 2013]. These approximate approaches were discussed in Section 1.2.1. Details of the implementation of these examples (and details explaining where to find the code for these examples) can be found in Appendix A.

5.1.3.1 Univariate distribution with light tails

We consider the univariate distribution with light tails example set out in Dai et al. [2019, Section 5.1], and consider the target density $f(x) \propto e^{-\frac{x^4}{2}}$. The sub-posterior densities are given by $f_c(x) = e^{-\frac{x^4}{2C}}$ for $c = 1, \ldots, C$, with $C = 5$. In this setting, Conditions 5.1.1–5.1.2 are satisfied and

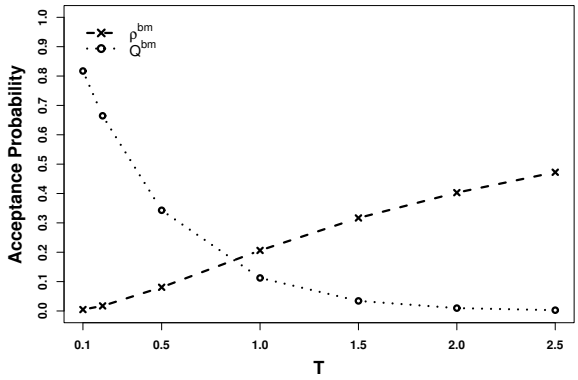
$$\phi_c^{dl}(x) = \frac{2x^6}{C^2} - \frac{3x^2}{C}, \tag{5.11}$$

and $\mathbf{\Phi}_c = 0$ for each $c = 1, \ldots, C$. See Appendix B.1 for the derivation of ϕ_c^{dl} for this example.

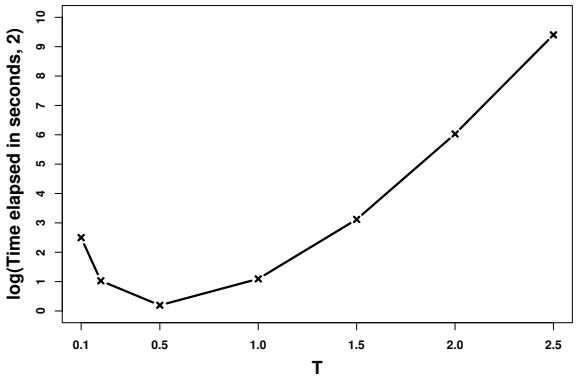
As noted previously, the value T used in Algorithm 5.1.3 has a large effect on the efficiency of the algorithm, which we show in Figure 5.2 by considering the computational cost of the MCF algorithm for simulating $N = 10000$ draws from f using a range of different values for T. In Figure 5.2a, we plot the acceptance rates for the two accept/reject steps in MCF. We can observe the trade-off between wanting to choose a small T to make ρ^{bm} in (5.8) larger, but this results in Q^{bm} in (5.9) (which we unbiasedly estimate as per Algorithm 5.1.2) to be smaller. The opposite is true when T is chosen to be larger. We will see later that for more difficult problems (e.g. in higher dimensions or when the sub-posteriors are more different), finding an optimal value for T which results in sufficiently large acceptance probabilities can be incredibly difficult.

The density curve estimation results are provided in Figure 5.2. We can see here that the CMC and KDEMC methods have very large biases whereas the WRS and MCF approaches are performing well and provide samples very close to the target density. Of course, with the MCF approach (here we have $T = 1$), we are providing direct i.i.d. samples from f.

Lastly, we consider a range of values for C to study how Algorithm 5.1.3 scales with the number of sub-posteriors to combine. Here, we fix $T = 1$ and plot the acceptance rates and computational run-times of Algorithm 5.1.3 in Figures 5.4a and 5.4b, respectively. Given the computational run-times are plotted in log-scale, we can observe that there seems to be an exponential cost with C for the MCF algorithm. In Chapter 6, we will develop the MCF approach further to be more scalable with regards to combining more sub-posteriors.



(a) Acceptance probabilities.



(b) Computational run-times.

Figure 5.2: Computational cost of Monte Carlo Fusion using different values for T with fixed $C = 5$ in Algorithm 5.1.3, as per the example in Section 5.1.3.1.

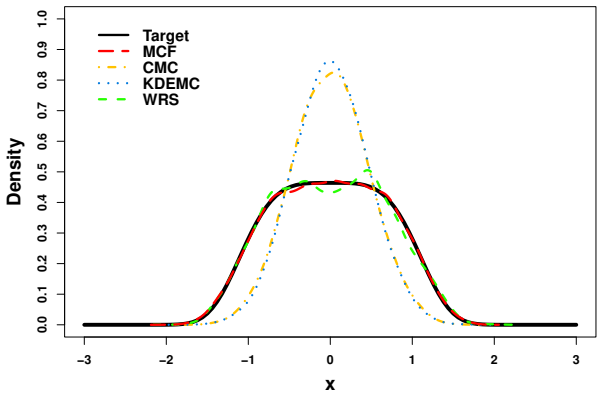
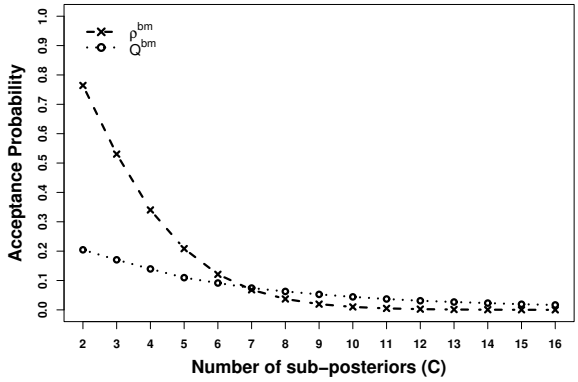
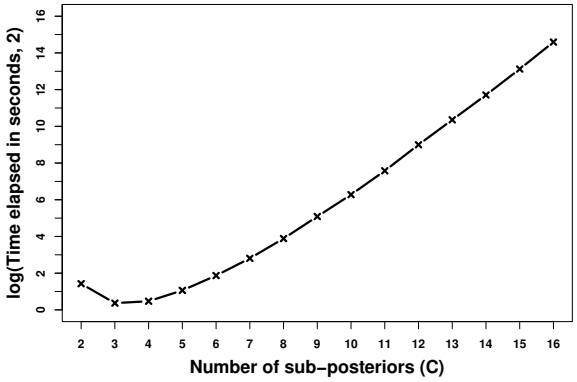


Figure 5.3: Kernel density fitting with bandwidth 0.1 for density $f(x) \propto e^{-\frac{x^4}{2}}$ based on different Monte Carlo methods for unifying sub-posterior samples.



(a) Acceptance probabilities.



(b) Computational run-times.

Figure 5.4: Computational cost of Monte Carlo Fusion with $T = 1$ in Algorithm 5.1.3 with varying C, as per the example in Section 5.1.3.1.

5.1.3.2 Univariate mixture Gaussian

We consider a univariate mixture Gaussian distribution with 3 components with density

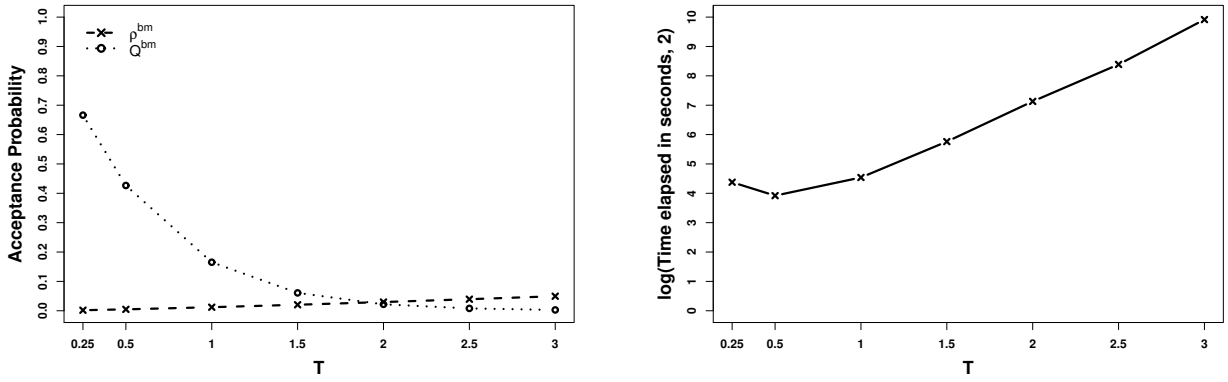
$$f(x) = 0.35 \cdot \mathcal{N}_1\left(x|-3, 1\right) + 0.2 \cdot \mathcal{N}_1\left(x|2, 1.5^2\right) + 0.45 \cdot \mathcal{N}_1\left(x|5, 0.5^2\right), \tag{5.12}$$

where $\mathcal{N}_1(x|\mu, \sigma^2)$ denotes the density of a univariate Normal distribution with mean μ and variance σ^2. In this example, our target density exhibits multi-modality and has 3 modes. Although the separation between the modes is not too extreme here, Markov chain Monte Carlo (MCMC) methods which use localised proposal mechanisms can still struggle to sample from f effectively and can face the difficulty of moving between modes as there is little probability mass (or *bridging mass*) between the modes. Popular approaches to overcome this issue are *simulated tempering* [Marinari and Parisi, 1992] and *parallel tempering* [Geyer, 1991; Geyer and Thompson, 1995; Tawn and Roberts, 2019] approaches which consider a *power-tempered target distribution*, $f_\beta(x) \propto [f(x)]^\beta$, for some inverse temperature level $\beta \in (0, 1]$. As $\beta \to 0$, f_β approaches a uniform distribution over the support of the target distribution. This essentially has the effect of flattening out the target distribution by spreading out mass into the tails of the distribution and forming bridging mass between modes [Tawn and Roberts, 2019], which means MCMC sampling is made much easier. In such approaches, a wide range of inverse temperature levels are considered and MCMC is run on an augmented space. As noted in Chapter 1, we can take an alternative approach by constructing a fusion problem by choosing a single inverse temperature β such that $\frac{1}{\beta} \in \mathbb{N}^+$ and Markov chain sampling from $f_\beta(x)$ can mix well across the entire sample space and by noting

$$f(x) \propto \prod_{i=1}^{\frac{1}{\beta}} f_\beta(x). \tag{5.13}$$

In this setting, we simply sample from each of the power-tempered target distributions (which can be done in parallel) and combine those samples to recover the target f. Here, we let $C = 4$, and sample from $f_c(x) = f^{\frac{1}{C}}(x)$ for $c = 1, \ldots, C$ (so $\beta = \frac{1}{4}$). Derivations of ϕ_c^{dl} and implementational details for this example can be found in Appendix B.2.

We consider the computational cost of running Algorithm 5.1.3 for simulating $N = 10000$ draws from f using a range of different values of T in Figure 5.5. As with the previous example, we observe the effect of choosing T on each of the acceptance probabilities in Figure 5.5a; namely ρ^{bm} increases as T increases while Q^{bm} gets smaller. Comparing the computational run-times in Figure 5.5b to the run-times for the previous example in Figure 5.2b, we can see that this example was much more expensive. One reason is that we struggled to find a value for T which resulted in good acceptance rates for both steps in Algorithm 5.1.3. As we have noted above, this is a common issue for Algorithm 5.1.3 and in the subsequent sections, we will see a number of alternative Fusion methods which attempt to alleviate many of the computational problems that MCF faces while still avoiding any approximations to the target f or the sub-posteriors f_c.



(a) Acceptance probabilities.

(b) Computational run-times.

Figure 5.5: Computational cost of Monte Carlo Fusion using different values for T with fixed $C = 4$ in Algorithm 5.1.3 as per the example in Section 5.1.3.2.

Kernel density curve estimation results for this example are given in Figure 5.6. We noted in Section 1.2.1 that CMC is exact if the sub-posteriors are Gaussian, but clearly in this case where the sub-posteriors exhibit multi-modality, the method does not capture this structure well. The KDEMC and WRS methods are performing better with the latter starting to capture the multi-modality well but still exhibits bias in the samples. Since the MCF approach returns i.i.d. samples from f, we can see that we are able to effectively recover the target distribution in this example.

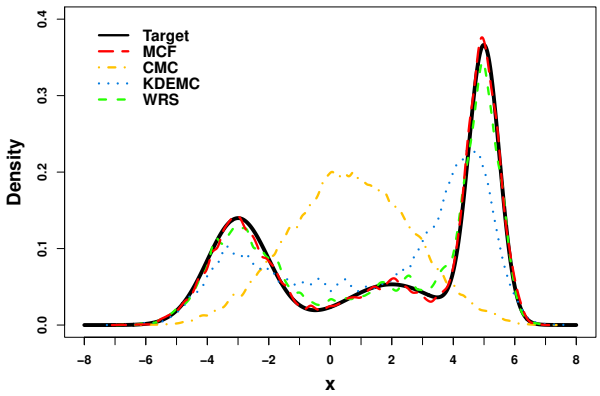


Figure 5.6: Kernel density fitting with bandwidth 0.1 for density based on different Monte Carlo methods for unifying sub-posterior samples.

5.2 Bayesian Fusion

The Monte Carlo Fusion (MCF) approach of Dai et al. [2019] was the first general method in tackling the fusion problem that avoided any approximation error in sampling from (1.1). However, as we have seen and also pointed out in Dai et al. [2021], the MCF approach faces a variety of computational difficulties, such as scalability with respect to number of sub-posteriors to combine and robustness to *conflicting* (or *heterogeneous*) sub-posteriors. The Bayesian Fusion (BF) approach was developed by Dai et al. [2021] to alleviate some of these problems. In this section, we will provide an overview of this approach from Dai et al. [2021].

5.2.1 Theory

One of the central ideas from the MCF approach [Dai et al., 2019] was that sampling from f in (1.1) (by means of sampling and evaluating functionals from sub-posterior densities f_c for $c = 1, \ldots, C$), was possible by sampling from an extended target *fusion measure* on an extended state space, which admitted f as a marginal. In particular, the MCF approach considered simulating C stochastic processes in such a way that they coalesce at a fixed time T, and the marginal distribution at the coalescence time T has the fusion density of (1.1). Sampling from this was achieved through a rejection sampling algorithm. In the BF approach of Dai et al. [2021], this is replaced with a sequential Monte Carlo (SMC) scheme which steps through a sequence of distributions between the initial proposal distribution to the target fusion density. The sequence of distributions is the joint distribution of our C *dependent* stochastic processes at times between 0 and the coalescence time T. The key contribution in Dai et al. [2021] is the development of tractable dynamics for these C stochastic processes which they term the *proposal measure*, denoted by $\mathbb{P}$, which is then suitably importance weighted to find the *fusion measure* $\mathbb{F}$. We first define some notation and terminology to define the Fusion measure.

Let $\mathbb{P}$ denote the *proposal measure* given by the probability law induced by C interacting d-dimensional parallel continuous-time Markov processes in $[0,T]$ where each process is given by the stochastic differential equation (SDE),

$$\mathrm{d}\boldsymbol{X}_t^{(c)} = \frac{\bar{\boldsymbol{X}}_t - \boldsymbol{X}_t^{(c)}}{T-t}\,\mathrm{d}t + \mathrm{d}\boldsymbol{W}_t^{(c)}, \quad \boldsymbol{X}_0^{(c)} := \boldsymbol{x}_0^{(c)} \sim f_c, \quad t \in [0,T], \tag{5.14}$$

where $\boldsymbol{W}_t^{(c)}$ for $c = 1,\ldots,C$, are independent Brownian motions and $\bar{\boldsymbol{X}}_t = \frac{1}{C}\sum_{c=1}^{C} \boldsymbol{X}_t^{(c)}$. To denote realisations from $\mathbb{P}$, we use the notation $\mathfrak{X} := \{\vec{\boldsymbol{x}}_t, t \in [0,T]\}$, where $\vec{\boldsymbol{x}}_t := \boldsymbol{x}_t^{(1:C)}$ is the Cd-dimensional vector of all processes at time t. The C processes interact through their average, $\bar{\boldsymbol{X}}_t$ for $t \in [0,T]$ and most importantly, we have coalescence at time T, so $\boldsymbol{x}_T^{(1)} = \cdots = \boldsymbol{x}_T^{(C)} =: \boldsymbol{y}$. The proof to show that the law of C independent Brownian motions initialised at the respective sub-posteriors, $\boldsymbol{x}_0^{(c)} \sim f_c$ for $c = 1,\ldots,C$, and conditioned to coalesce at time T satisfies (5.14) is given in Dai et al. [2021, Appendix A] which uses Doob h-transforms [Rogers and Williams, 2000, Chapter IV, Section 6.39]. We will defer this proof for Chapter 7 in Part II when we generalise and develop the Bayesian Fusion methodology further (see Theorem 7.1.1).

Given the proposal measure $\mathbb{P}$, Dai et al. [2021] define the *Fusion measure*, denoted $\mathbb{F}$, as the probability measure induced by the following Radon-Nikodým derivative,

$$\frac{\mathrm{d}\mathbb{F}}{\mathrm{d}\mathbb{P}}(\mathfrak{X}) \propto \rho_0^{bm}\left(\vec{\boldsymbol{x}}_0\right) \cdot \prod_{c=1}^{C}\left[\exp\left\{-\int_0^T \phi_c^{dl}\left(\boldsymbol{X}_t^{(c)}\right)\,\mathrm{d}t\right\}\right], \tag{5.15}$$

where $\{\boldsymbol{X}_t^{(c)}, t \in [0,T]\}$ is a Brownian bridge from $\boldsymbol{X}_0^{(c)} := \boldsymbol{x}_0^{(c)} \sim f_c$ to $\boldsymbol{X}_T^{(c)} := \boldsymbol{x}_T^{(c)}$, $\phi_c^{dl}(\boldsymbol{x})$ is defined in (5.6), and $\rho_0^{bm}(\vec{\boldsymbol{x}}_0) := \rho^{bm}(\boldsymbol{x}_0^{(1)},\ldots,\boldsymbol{x}_0^{(C)})$ from (5.8). Dai et al. [2021] establish (through the following theorem) that its possible sample from f by means of the T temporal marginal of $\mathbb{F}$. As with MCF (detailed in Section 5.1), the regularity conditions 5.1.1–5.1.2 are imposed.

Theorem 5.2.1. *[Dai et al., 2021, Theorem 1]. Under Condition 5.1.1 and Condition 5.1.2, with probability* 1*, we have that under the Fusion measure, $\mathbb{F}$, the end points of the C parallel processes have a common value $\boldsymbol{y} := \boldsymbol{x}_T^{(1)} = \cdots = \boldsymbol{x}_T^{(C)}$ and $\boldsymbol{y} \sim f$.*

Proof. See Dai et al. [2021, Appendix A]. This also follows directly from Theorem 7.1.1 by setting $\boldsymbol{\Lambda}_c = \mathbb{I}_d$ for $c \in \mathcal{C} := \{1,\ldots,C\}$. ■

5.2.2 Methodology

Theorem 5.2.1 suggests that we can simulate from the fusion target density f in (1.1) by simulating $\mathfrak{X} \sim \mathbb{F}$ and retaining the T time marginal, $\boldsymbol{y}$. Dai et al. [2019] notes that although direct simulation from $\mathbb{F}$ is not generally available, we could construct a rejection sampler for $\mathbb{F}$ by means of simulating proposals $\mathfrak{X} \sim \mathbb{P}$ and accepting them with probability proportional to the Radon-

Nikodým derivative in (5.15). Since paths $\mathfrak{X} \sim \mathbb{P}$ are infinite dimensional random variables, we cannot draw entire sample paths from $\mathbb{P}$. However, we can simulate a finite dimensional subset of the process at discrete time points without error. Consider the auxiliary temporal partition, $\mathcal{P} = \{t_0, t_1, \ldots, t_n : 0 =: t_0 < t_1 < \cdots < t_n := T\}$ and let $\Delta_j := t_j - t_{j-1}$. For notational simplicity, we suppress subscripts when considering the processes at times given in the temporal partition (i.e. let $\boldsymbol{x}_j^{(c)}$ denote $\boldsymbol{x}_{t_j}^{(c)}$, and let $\vec{\boldsymbol{x}}_j$ denote $\vec{\boldsymbol{x}}_{t_j}$). The following theorem tells us how to simulate from $\mathbb{P}$ without discretisation error.

Theorem 5.2.2. *[Dai et al., 2021, Theorem 2] If $\mathfrak{X}$ satisfies* (5.14)*, then under the proposal measure, $\mathbb{P}$, we have*

(a) For $s < t$,

$$\vec{\boldsymbol{X}}_t \Big| \left(\vec{\boldsymbol{X}}_s = \vec{\boldsymbol{x}}_s \right) \sim \mathcal{N}_{Cd} \left(\vec{\boldsymbol{M}}_{s,t}, \boldsymbol{V}_{s,t} \right), \tag{5.16}$$

where $\vec{\boldsymbol{M}}_{s,t} := \left(\boldsymbol{M}_{s,t}^{(1)}, \ldots, \boldsymbol{M}_{s,t}^{(C)} \right)$, with

$$\boldsymbol{M}_{s,t}^{(c)} = \frac{T-t}{T-s} \boldsymbol{x}_s^{(c)} + \frac{t-s}{T-s} \bar{\boldsymbol{x}}_s, \tag{5.17}$$

and $\boldsymbol{V}_{s,t} = \boldsymbol{\Sigma} \otimes \mathbb{I}_d$, where $\otimes$ denotes the Kronecker product, and $\boldsymbol{\Sigma} \in \mathbb{R}^{C \times C}$ given by

$$\boldsymbol{\Sigma}_{ii} = \frac{(t-s)(T-t)}{T-s} + \frac{(t-s)^2}{C(T-s)}, \qquad \boldsymbol{\Sigma}_{ij} = \frac{(t-s)^2}{C(T-s)}. \tag{5.18}$$

(b) For each $c = 1, \ldots, C$, the distribution of $\{\boldsymbol{X}_q^{(c)}, s \leq q \leq t\}$ given endpoints $\boldsymbol{X}_s^{(c)} = \boldsymbol{x}_s^{(c)}$ and $\boldsymbol{X}_t^{(c)} = \boldsymbol{x}_t^{(c)}$ is a Brownian bridge, so

$$\boldsymbol{X}_u^{(c)} \Big| \left(\boldsymbol{x}_s^{(c)}, \boldsymbol{x}_t^{(c)} \right) \sim \mathcal{N}_d \left(\frac{(t-q)\boldsymbol{x}_s^{(c)} + (q-s)\boldsymbol{x}_t^{(c)}}{t-s}, \frac{(t-q)(q-s)}{t-s} \mathbb{I}_d \right). \tag{5.19}$$

Proof. See Dai et al. [2021, Appendix B]. This also follows directly from Theorem 7.1.2 by setting $\boldsymbol{\Lambda}_c = \mathbb{I}_d$ for $c \in \mathcal{C} := \{1, \ldots, C\}$. ■

Using Theorem 5.2.1 and simplifying notation again by suppressing subscripts to set $\vec{\boldsymbol{M}}_j := \vec{\boldsymbol{M}}_{t_{j-1},t_j}$ and $\boldsymbol{V}_j := \boldsymbol{V}_{t_{j-1},t_j}$, the $(nC+1)d$-dimensional density of the Markov processes at the $(n+1)$ time marginals given by the temporal partition, $\mathcal{P}$, under the proposal measure $\mathbb{P}$ (illustrated in Figure 5.7, given by

$$h^{bf}\left(\vec{\boldsymbol{x}}_0, \ldots, \vec{\boldsymbol{x}}_{n-1}, \boldsymbol{y}\right) \propto \prod_{c=1}^{C} \left[f_c\left(\boldsymbol{x}_0^{(c)}\right) \right] \cdot \prod_{j=1}^{n} \mathcal{N}_{Cd}\left(\vec{\boldsymbol{x}}_j \Big| \vec{\boldsymbol{M}}_j, \boldsymbol{V}_j\right), \tag{5.20}$$

where $\mathcal{N}_d(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the density of a d-dimensional Normal distribution (evaluated at $\boldsymbol{x}$) with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

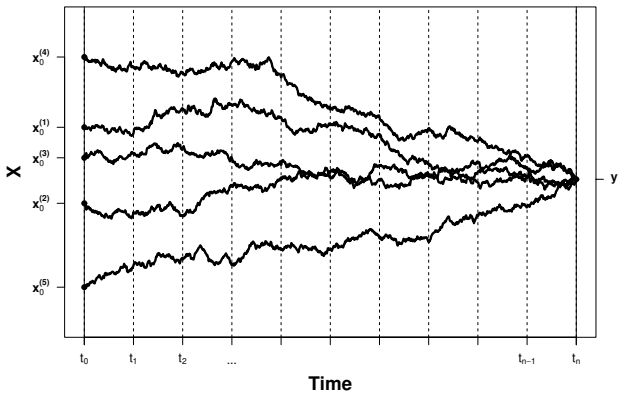


Figure 5.7: Illustration of the $(nC+1)d$-dimensional density (for $d=1$) corresponding to a typical realisation of $\mathfrak{X}$ at the time marginals in $\mathcal{P}$.

Simulating from h^{bf} can be achieved by first simulating $\boldsymbol{x}_0^{(c)} \sim f_c$ independently for $c = 1, \ldots, C$, and directly applying Theorem 5.2.2 iteratively for each time point in $\mathcal{P}$. For simplicity, we typically that we have access to independent realisations from each sub-posterior, but as Dai et al. [2021, Section 3.6] notes, we may naturally only have sample approximations of each sub-posterior obtained by another Monte Carlo scheme (for instance Markov chain Monte Carlo). We discuss the impact of using approximate samples of each sub-posterior in Section 5.2.3.1.

By factorising the Fusion measure in (5.15) according to the time marginals given by the temporal partition $\mathcal{P}$, the $(nC+1)d$-dimensional target density is given by

$$g^{bf}\left(\vec{\boldsymbol{x}}_0, \ldots, \vec{\boldsymbol{x}}_{n-1}, \boldsymbol{y}\right) \propto h^{bf}\left(\vec{\boldsymbol{x}}_0, \ldots, \vec{\boldsymbol{x}}_{n-1}, \boldsymbol{y}\right) \cdot \prod_{j=0}^{n} \rho_j^{bm}, \tag{5.21}$$

where $\rho_0^{bm} := \rho_0^{bm}(\vec{\boldsymbol{x}}_0) = \rho^{bm}(\boldsymbol{x}_0^{(1)}, \ldots, \boldsymbol{x}_0^{(C)})$ from (5.8), and for $j = 1, \ldots, n$,

$$\rho_j^{bm} := \rho_j^{bm}\left(\vec{\boldsymbol{x}}_{j-1}, \vec{\boldsymbol{x}}_j\right) = \prod_{c=1}^{C} \mathbb{E}_{\mathbb{W}_{j,c}}\left[\exp\left\{-\int_{t_{j-1}}^{t_j} \left(\phi_c^{dl}\left(\boldsymbol{X}_t^{(c)}\right) - \boldsymbol{\Phi}_c^{bm}\right) \, \mathrm{d}t\right\}\right] \in (0, 1], \tag{5.22}$$

where for $c = 1, \ldots, C$, $\mathbb{W}_{j,c}$ is the law of a Brownian bridge $\{\boldsymbol{X}_t^{(c)}, t \in (t_{j-1}, t_j)\}$ from $\boldsymbol{X}_{t_{j-1}} := \boldsymbol{x}_{j-1}^{(c)}$ to $\boldsymbol{X}_{t_j} := \boldsymbol{x}_j^{(c)}$, ϕ_c^{dl} is defined in (5.6) and $\boldsymbol{\Phi}_c^{bm} > -\infty$ is the constant such that for all $\boldsymbol{x}$, $\phi_c^{dl}(\boldsymbol{x}) \geq \boldsymbol{\Phi}_c^{bm}$ imposed by Condition 5.1.2.

Following a similar approach to the MCF method in Section 5.1, rather than simulating $\mathfrak{X} \sim \mathbb{F}$, we can simulate from the extended target density g^{bf} and consider the T time marginal to obtain samples from the fusion density f. From (5.21), we can simulate from g^{bf} by means of rejection sampling by proposing from the density h^{bf} and accepting the proposal with probability $\prod_{j=0}^{n} \rho_j^{bm}$.

However, Dai et al. [2021] notes that a rejection sampler can be inefficient in this setting since the acceptance probability in (5.22) typically decays geometrically with increasing number of sub-posteriors C, since each of the terms in the product is bounded by 1. An alternative approach would be to construct an importance sampler (see Section 2.3) whereby importance weights given by $\prod_{j=0}^{n} \rho_j^{bm}$ are assigned to each proposal from h^{bf}. However, this scheme would suffer from similar inefficiencies as the rejection sampler but through the variance of the importance weights instead. The main insight from the BF approach of Dai et al. [2021] is that we can utilise Theorem 5.2.2 to construct a SMC approach. A complication here is that whilst ρ_0^{bm} is readily computable, direct computation of $\rho_1^{bm}, \ldots, \rho_n^{bm}$ is not possible since this requires evaluation of path integrals of functionals of Brownian motion. However, we can construct unbiased estimators for ρ_j^{bm} for $j = 1, \ldots, n$, in a similar fashion as we did for Q^{bm} (5.9) with Algorithm 5.1.2.

Recall from Section 4.2 and Section 5.1, to find an unbiased estimator for ρ_j^{bm}, we need to find upper and lower bounds for $\phi_c^{dl}(\boldsymbol{X}_t^{(c)})$ for $t \in [t_{j-1}, t_j]$. To achieve this in general, we can bound a sample path $\boldsymbol{X}_{[t_{j-1},t_j]}^{(c)} \sim \mathbb{W}_{j,c}$, and conditional on these *layers*/bounds of the sample path, we can find upper and lower bounds of ϕ_c^{dl} denoted $U_j^{(c)}$ and $L_j^{(c)}$, respectively, such that $\phi_c^{dl}(\boldsymbol{X}_t^{(c)}) \in [L_j^{(c)}, U_j^{(c)}]$ for $t \in [t_{j-1}, t_j]$. Here, let $R_c := R_c(\boldsymbol{X}_{[t_{j-1},t_j]})$ denote the compact region in which $\boldsymbol{X}_t^{(c)}$ is constrained in time $[t_{j-1}, t_j]$. We can use Bessel layers (4.25), for instance, to construct partitions of the space of $\mathbb{W}_{j,c}$ into disjoint layers and simulate a layer $R_c \sim \mathcal{R}_c$ using Algorithm 4.2.4. As discussed in Section 4.2.2, it is then possible to further simulate the path at any required time marginals conditional on the simulated layer, $\boldsymbol{X}_t^{(c)} \sim \mathbb{W}_{j,c}|R_c$ as per Algorithm 4.2.5. Let

$$\tilde{\rho}_j^{bm}\left(\vec{\boldsymbol{x}}_{j-1}, \vec{\boldsymbol{x}}_j\right) := \prod_{c=1}^{C} \left(\frac{\Delta_j^{\kappa_c} \cdot e^{-U_X^{(c)}\Delta_j}}{\kappa_c! \cdot p\left(\kappa_c | R_c\right)} \cdot \prod_{k_c=1}^{\kappa_c} \left(U_X^{(c)} - \phi_c^{dl}\left(\boldsymbol{X}_{\xi_{c,k_c}}^{(c)}\right)\right)\right), \tag{5.23}$$

for $j = 1, \ldots, n$, where κ_c in (5.23) denotes a non-negative, integer-valued, random variable with probabilities conditional on R_c, denoted by $p(\cdot|R_c)$ and $\{\xi_{c,1}, \ldots, \xi_{c,\kappa_c}\} \sim \mathcal{U}[t_{j-1}, t_j]$. To find an unbiased estimator of ρ_j^{bm} for $j = 1, \ldots, n$, we establish the following theorem.

Theorem 5.2.3. *[Dai et al., 2021, Theorem 3]. Let $a_j := \exp\{\sum_{c=1}^{C} \boldsymbol{\Phi}_c^{bm} \Delta_j\}$, then for every $j = 1, \ldots, n$, $a_j \tilde{\rho}_j^{bm}$ is an unbiased estimator of ρ_j^{bm}. In particular, we have*

$$\rho_j^{bm} = \mathbb{E}_{\bar{\mathcal{R}}} \mathbb{E}_{\bar{\mathbb{W}}|\bar{\mathcal{R}}} \mathbb{E}_{\bar{\mathbb{K}}} \mathbb{E}_{\bar{\mathbb{U}}} \left[a_j \tilde{\rho}_j^{bm} \right], \tag{5.24}$$

where expectation subscripts denote the law with which they are taking; $\mathcal{R}$ denotes the law of $\{R_c \sim \mathcal{R}_c : c = 1, \ldots, C\}$, $\bar{\mathbb{W}}$ denotes the law of the C Brownian bridges $\{\mathbb{W}_{j,c} : c = 1, \ldots, C\}$, $\bar{\mathbb{K}}$ denotes the law of $\{\kappa_c : c = 1, \ldots, C\}$ and $\bar{\mathbb{U}}$ denotes the law of $\{\xi_{c,1}, \ldots, \xi_{c,\kappa_c} : c = 1, \ldots, C\} \overset{iid}{\sim} \mathcal{U}[t_{j-1}, t_j]$.

Proof. See Dai et al. [2021, Appendix B]. This also follows directly from Theorem 7.1.3 by setting $\boldsymbol{\Lambda}_c = \mathbb{I}_d$ for $c \in \mathcal{C} := \{1, \ldots, C\}$. ■

The problem of choosing an appropriate $p(\kappa_c|R_c)$ is discussed in Section 4.4.3, and the two choices we consider throughout this thesis are the GPE-1 and GPE-2 estimators:

Definition 5.2.1. (GPE-1 for ρ_j^{bm} (5.22)): Choosing the law of $\kappa_c \sim \text{Poi}\big((U_j^{(c)} - L_j^{(c)})\Delta_j\big)$ for $c = 1, \ldots, C$, leads to the following estimator:

$$\tilde{\rho}_j^{bm,(a)}\left(\vec{\boldsymbol{x}}_{j-1}, \vec{\boldsymbol{x}}_j\right) := \prod_{c=1}^{C}\left(e^{-L_j^{(c)}\Delta_j} \cdot \prod_{k_c=1}^{\kappa_c}\left[\frac{U_j^{(c)} - \phi_c^{dl}\left(\boldsymbol{X}_{\xi_{c,k_c}}^{(c)}\right)}{U_j^{(c)} - L_j^{(c)}}\right]\right), \tag{5.25}$$

where $\exp\{\sum_{c=1}^{C} \boldsymbol{\Phi}_c^{bm}\Delta_j\} \cdot \tilde{\rho}_j^{bm,(a)}$ is an unbiased estimator for ρ_j^{bm}.

Definition 5.2.2. (GPE-2 for ρ_j^{bm} (5.22)): Choosing the law of $\kappa_c \sim \text{NB}(\gamma_c, \beta_c)$ for $c = 1, \ldots, C$, with

$$\gamma_c := U_j^{(c)}\Delta_j - \int_{t_{j-1}}^{t_j} \phi_c^{dl}\left(\boldsymbol{x}_{j-1}^{(c)} \cdot \frac{t_j - s}{\Delta_j} + \boldsymbol{x}_j^{(c)} \cdot \frac{s - t_{j-1}}{\Delta_j}\right) \mathrm{d}s, \tag{5.26}$$

leads to the following estimator:

$$\tilde{\rho}_j^{bm,(b)}\left(\vec{\boldsymbol{x}}_{j-1}, \vec{\boldsymbol{x}}_j\right) := \prod_{c=1}^{C}\left(e^{-U_j^{(c)}\Delta_j} \cdot \frac{\Delta_j^{\kappa_c} \cdot \Gamma(\beta_c) \cdot (\beta_c + \gamma_c)^{\beta_c + \kappa_c}}{\Gamma(\beta_c + \kappa_c)\beta_c^{\beta_c}\gamma_c^{\kappa_c}} \cdot \prod_{k_c=1}^{\kappa_c}\left[U_j^{(c)} - \phi_c^{dl}\left(\boldsymbol{X}_{\xi_{c,k_c}}^{(c)}\right)\right]\right), \tag{5.27}$$

where $\exp\{\sum_{c=1}^{C} \boldsymbol{\Phi}_c^{bm}\Delta_j\} \cdot \tilde{\rho}_j^{bm,(b)}$ is an unbiased estimator for ρ_j^{bm}.

In MCF [Dai et al., 2019] (see Section 5.1), a Poisson distribution (GPE-1) was used for the distribution of κ_c for $c = 1, \ldots, C$, since this lead to a *positive, bounded* unbiased estimator, which is required for a rejection sampler. However, as we noted above, a key development in BF was to replace the rejection sampler with a SMC scheme. This motivates the use of GPE-2 since it has been shown empirically to outperform GPE-1 in terms of having a lower variance [Fearnhead et al., 2010, Section 5]. Dai et al. [2021, Section 3.5] also showed empirically that GPE-2 outperforms GPE-1 with a number of simulation studies. A summary of the full construction of (5.23) can be found in Dai et al. [2021, Appendix B], but it is essentially a combination of arguments and algorithms that are presented in Chapter 4. We can simulate $\tilde{\rho}_j^{bm}$ as per Algorithm 5.2.1.

Algorithm 5.2.1 Simulating $\tilde{\rho}_j^{bm}$ (5.23) [Dai et al., 2021, Algorithm 4].

1. For $c = 1, \ldots, C$,
   (a) R_c: Simulate $R_c \sim \mathcal{R}$ as per Algorithm 4.2.4.
   (b) $L_j^{(c)}, U_j^{(c)}$: Compute lower and upper bounds, $L_j^{(c)}$ and $U_j^{(c)}$, of $\phi_c^{dl}(\boldsymbol{x})$ for $\boldsymbol{x} \in R_c$.
   (c) p_c: Choose $p(\cdot|R_c)$ using either GPE-1 (Condition 5.2.1) or GPE-2 (Condition 5.2.2).
   (d) κ_c, ξ: Simulate $\kappa_c \sim p(\cdot|R_c)$, and simulate $\xi_{c,1}, \ldots, \xi_{c,\kappa_c} \sim \mathcal{U}[t_{j-1}, t_j]$.
   (e) $\boldsymbol{X}^{(c)}$: Simulate $\boldsymbol{X}_{\xi_{c,1}}^{(c)}, \ldots, \boldsymbol{X}_{\xi_{c,\kappa_c}}^{(c)} \sim \mathbb{W}_c|R_c$ as per Algorithm 4.2.5.
2. Output $\tilde{\rho}_j^{bm,(\cdot)}$ (5.23).

Given unbiased estimators for ρ_j^{bm} for each $j = 1, \ldots, n$, then it is possible to implement a SMC algorithm for sampling from from the extended target g^{bf} in (5.21) with proposal h^{bf} in (5.20). This algorithm is termed *Bayesian Fusion* and is presented in Algorithm 5.2.2.

The algorithm is initialised by simulating N particles from the time 0 marginal of h^{bf} (5.20), denoted $\{\vec{\boldsymbol{x}}_{0,i}\}_{i=1}^N$ (recalling that $\vec{\boldsymbol{x}}_{0,i} = \boldsymbol{x}_{0,i}^{(1:C)}$, where $\boldsymbol{x}_{0,i}^{(c)} \sim f_c$ for $c = 1, \ldots, C$). Each particle is then assigned an un-normalised importance weight $w'_{0,i} := \rho_0^{bm}(\vec{\boldsymbol{x}}_{0,i})$ for $i = 1, \ldots, N$ and this initial particle set can be used to approximate the time 0 marginal of g^{bf}. The particles are then iteratively propagated n times using Theorem 5.2.2, and the weights are updated by a factor of $a_j \tilde{\rho}_j^{bm}(\vec{\boldsymbol{x}}_{j-1,i}, \vec{\boldsymbol{x}}_{j,i})$ for $i = 1, \ldots, N$ at each iteration $j = 1, \ldots, n$. The weighted particle set obtained at the final nth iteration of the algorithm gives an approximation of the time T marginal of g^{bf}, which as noted above can be used as an approximation to the target fusion density f (established in Theorem 5.2.1). At the end of each iteration, the importance weights are *normalised*. As such, we can adopt the common approach for monitoring and combating weight degeneracy (see Section 3.2 and Section 3.3) by computing an estimate of the *effective sample size (ESS)* (given in (3.18)) and introduce *resampling steps* if the estimated ESS falls below a user-specified threshold. Also note that the normalisation of the importance weights removes the contributions from $\boldsymbol{\Phi}_1^{bm}, \ldots, \boldsymbol{\Phi}_C^{bm}$ from $a_j \tilde{\rho}_j^{bm}(\vec{\boldsymbol{x}}_{j-1,\cdot}, \vec{\boldsymbol{x}}_{j,\cdot})$, since a_j is a common constant which is cancelled out in normalisation. As such, it is possible to avoid the computation of $\boldsymbol{\Phi}_c^{bm}$ for all $c = 1, \ldots, C$, and we only need to evaluate $\tilde{\rho}_j^{bm}$ as per Algorithm 5.2.1.

Algorithm 5.2.2 Bayesian Fusion [Dai et al., 2021, Algorithm 1].

---

1. **Initialisation** ($j = 0$):
   (a) **Input:** Sub-posteriors $f_1, \ldots, f_C$, number of particles N, time horizon T, and temporal partition $\mathcal{P} := \{t_0, t_1, \ldots, t_n : 0 := t_0 < t_1 < \cdots < t_n := T\}$.
   (b) For i in 1 to N,
      i. $\vec{\boldsymbol{x}}_{0,i}$: For $c = 1, \ldots, C$, simulate $\boldsymbol{x}_{0,i}^{(c)} \sim f_c$. Set $\vec{\boldsymbol{x}}_{0,i} := \boldsymbol{x}_{0,i}^{(1:C)}$.
      ii. Compute un-normalised weight $w'_{0,i} := \rho_0^{bm}(\vec{\boldsymbol{x}}_{0,i})$ as per (5.8).
   (c) $w_{0,i}$**:** For i in 1 to N, compute normalised weight $w_{0,i} = w'_{0,i} / \sum_{k=1}^N w'_{0,k}$.
   (d) g_0^N: Set $g_0^N(\mathrm{d}\vec{\boldsymbol{x}}) := \sum_{i=1}^N w_{0,i} \cdot \delta_{\vec{\boldsymbol{x}}_{0,i}}(\mathrm{d}\vec{\boldsymbol{x}})$.
2. **Iterative updates**. For $j = 1, \ldots, n$:
   (a) **Resample:** If the ESS $:= \left(\sum_{i=1}^N w_{j-1,i}^2\right)^{-1}$ breaches the lower user-specified threshold, then for $i = 1, \ldots, N$, resample $\vec{\boldsymbol{x}}_{j-1,i} \sim g_{j-1}^N$ and set $w_{j-1,i} = \frac{1}{N}$.
   (b) For i in 1 to N,
      i. $\vec{\boldsymbol{x}}_{j,i}$**:** Simulate $\vec{\boldsymbol{x}}_{j,i} \sim \mathcal{N}_d\left(\vec{\boldsymbol{M}}_{j,i}, \boldsymbol{V}_j\right)$ as per Theorem 5.2.2.
      ii. $w'_{j,i}$**:** Compute un-normalised weight $w'_{j,i} = w_{j-1,i} \cdot \tilde{\rho}_j^{bm}(\vec{\boldsymbol{x}}_{j-1,i}, \vec{\boldsymbol{x}}_{j,i})$ as per Algorithm 5.2.1.
   (c) $w_{j,i}$**:** For i in 1 to N, compute normalised weight $w_{j,i} = w'_{0,i} / \sum_{k=1}^N w'_{j,k}$.
   (d) g_j^N: Set $g_j^N(\mathrm{d}\vec{\boldsymbol{x}}_j) := \sum_{i=1}^N w_{j,i} \cdot \delta_{\vec{\boldsymbol{x}}_{j,i}}(\mathrm{d}\vec{\boldsymbol{x}}_j)$.
3. **Output:** $\left\{\vec{\boldsymbol{x}}_{0,i}, \ldots, \vec{\boldsymbol{x}}_{n-1,i}, \boldsymbol{y}_i, w_{n,i}\right\}_{i=1}^N$, where $\hat{f}(\mathrm{d}\boldsymbol{y}) := \sum_{i=1}^N w_{n,i} \cdot \delta_{\boldsymbol{y}_i}(\mathrm{d}\boldsymbol{y}) \approx f(\mathrm{d}\boldsymbol{y})$.

5.2.3 Implementational guidance for Bayesian Fusion

One of the key reasons why BF can be applied to a wider range of applications (when compared to MCF) is that there is more flexibility in the hyperparameters. As noted in Section 5.1, with MCF (Algorithm 5.1.3), if T is chosen to be small, then the first accept/reject acceptance probability, ρ^{bm}, is smaller, while the second acceptance probability Q^{bm} is larger. For larger values of T, the opposite is true. A key drawback with the MCF approach is that in many practical settings, it is difficult to choose a value of T which leads to sufficiently large acceptance probabilities. For many practical problems, it is unrealistic to be able to find the optimal tuning parameter easily and even if the optimal T is found, it is likely that the acceptance probabilities ρ^{bm} and Q^{bm} are very small. In contrast, BF introduces a temporal partition of time T, denoted $\mathcal{P}$. This means that to achieve a good initialisation of the algorithm, we have the flexibility to make T sufficiently large such that the initial importance weights $\{\rho_{0,i}^{bm}\}_{i=1}^{N}$ have low variance. Further, to ensure that the subsequent iterative steps of the algorithm are stable and the incremental importance weights $\{\tilde{\rho}_{j,i}^{bm}\}_{i=1}^{N}$ have low variance, we can simply impose a finer temporal mesh in $\mathcal{P}$. Of course, this comes at the cost of increasing the number of iterations in the algorithm, n, but this ultimately results in a more scalable and robust approach over the MCF approach of Dai et al. [2019].

It is clear to see that the efficiency of Algorithm 5.2.2 depends on the user-specified time $T > 0$ and the temporal partition $\mathcal{P}$. Dai et al. [2021, Section 3.1–3.2] provides guidance on selecting these hyperparameters along with additional practical guidance on implementation. However, we will not discuss the guidance for selecting T and $\mathcal{P}$ for the BF approach provided in Dai et al. [2021, Section 3.1–3.2] since in Chapter 7, we will develop a *Generalised Bayesian Fusion (GBF)* approach which admits the BF as a special case. We subsequently further develop the implementational guidance for the GBF approach (see Section 7.3) which can be applied to the BF setting too.

The main motivation in the development of the BF approach was to develop a practical SMC approach for inference in the fusion problem (i.e. providing a sample approximation for (1.1)). The methodology discussed in Section 5.2 attempts to improve on the scalability of the MCF approach (see Section 5.1) by helping to alleviate the problems of robustness with regards to number of sub-posteriors and sub-posterior heterogeneity. However, as Dai et al. [2021, Section 3.6–3.7] notes, there could be applications of Fusion which come with several additional problem-specific constraints that require some modification of Algorithm 5.2.2. For example, as noted in Section 1.2, the development of approximate methods for the fusion problem have typically been motivated by developing methodology for performing Bayesian inference with large datasets. In this setting, *latency* in communication between cores may be of particular concern [Scott et al., 2016; Dai et al., 2021]. Furthermore, we may also have the problem where there is a large amount of data associated to each of the individual sub-posteriors, so it may not be computationally efficient to evaluate certain quantities in Algorithm 5.2.2 (such as computing the weights in Step 2(b)ii) which would need to be modified appropriately. In this section, we will summarise several modifications of the BF methodology proposed by Dai et al. [2021, Section 3.6–3.7].

5.2.3.1 Using approximate samples from the sub-posteriors

The Fusion methodologies typically assume that we have access to independent realisations from each sub-posterior, however in many cases we will not be able to simulate i.i.d. draws from each sub-posterior. More realistically, we may only have sample approximations of each sub-posterior obtained by another Monte Carlo scheme (for instance Markov chain Monte Carlo). Dai et al. [2021, Section 3.6] analyses the impact of using approximate sub-posteriors by denoting ψ_c the cth *normalised* sub-posterior density and $\psi_c^{(N)}$ denote the approximation of ψ_c using a Monte Carlo sample of size N. Here, we naturally assume that $\left\|\psi_c(\boldsymbol{x}^{(c)}) - \psi_c^{(N)}(\boldsymbol{x}^{(c)})\right\| \to 0$ as $N \to \infty$ and Dai et al. [2021, Theorem 6] show that by substituting $\psi_c^{(N)}$ for ψ_c in Algorithm 5.2.2 for large enough N, this would still output $\boldsymbol{y}$ arbitrarily close to the target fusion density f. For notational simplicity, Dai et al. [2021] take $n = 1$ and notes that in this setting, we simulate $\vec{\boldsymbol{x}}_0$ and $\boldsymbol{y}$ from

$$g^{bf,(N)}(\vec{\boldsymbol{x}}_0, \boldsymbol{y}) \propto \prod_{c=1}^{C} \left[\psi_c^{(N)}\left(\boldsymbol{x}_0^{(c)}\right)\right] \cdot \mathcal{N}_d\left(\vec{\boldsymbol{M}}_1, \boldsymbol{V}_1\right) \cdot \rho_0^{bm} \cdot \rho_1^{bm}, \tag{5.28}$$

whilst noting that for general values of n, the proof is similar.

Theorem 5.2.4. *[Dai et al., 2021, Theorem 6]. Suppose for $\varepsilon > 0$, there exists a N_0 such that for $N > N_0$,*

$$\left\|\psi_c\left(\boldsymbol{x}^{(c)}\right) - \psi_c^{(N)}\left(\boldsymbol{x}^{(c)}\right)\right\| \leq \varepsilon,$$

for all $c = 1, \ldots, C$. Then for any ε^*, we can find a N' such that when $N > N'$, we have

$$\int \left| \int \left(g^{bf}(\vec{\boldsymbol{x}}_0, \boldsymbol{y}) - g^{bf,(N)}(\vec{\boldsymbol{x}}_0, \boldsymbol{y})\right) \mathrm{d}\vec{\boldsymbol{x}}_0 \right| \mathrm{d}\vec{\boldsymbol{x}}_1 \leq \varepsilon^*.$$

Proof. See Dai et al. [2021, Appendix F]. ■

If $\psi_c^{(N)}$ for $c = 1, \ldots, C$, are obtained by some Monte Carlo approach (e.g. MCMC), then some care must be taken if the approximate sub-posterior samples are serially correlated. Dai et al. [2021, Section 3.6] notes that analysing theoretically the impact of such approximations is challenging, but in practice, one could thin the MCMC output for each sub-posterior, or randomly sample from the MCMC trajectories when initialising the particle set in Algorithm 5.2.2 Step 1(b)i.

5.2.3.2 Reducing communication between the cores

Dai et al. [2021, Section 3.7.1–3.7.2] identified two steps in Algorithm 5.2.2 in which communication between cores could be reduced; during initialisation of the particle set (in Step 1b) and when propagating the particle set in the iterative steps of the algorithm (in Step 2(b)i).

In the initialisation of the particle set, we first compose particles $\{\vec{\boldsymbol{x}}_{0,i} := \boldsymbol{x}_{0,i}^{(1:C)}\}_{i=1}^N$ in Step 1(b)i where $\boldsymbol{x}_{0,i}^{(c)} \sim f_c$ for $c = 1, \ldots, C$. Composing $\{\vec{\boldsymbol{x}}_{0,i}\}_{i=1}^N$ requires communications between the

cores, and there is a further communication back to the cores for the computation of the proposal importance weight $\rho_0^{bm}(\vec{\boldsymbol{x}}_{0,\cdot})$ in Step 1(b)ii (since the mean $\bar{\boldsymbol{x}}_{0,i}$ must be sent back to the individual machines). Lastly, there is a third communication between the cores to compute $\rho_0^{bm}(\vec{\boldsymbol{x}}_{0,\cdot})$ since it can be trivially decomposed into a product of C terms corresponding to the contribution from each core separately. To reduce the amount of communication between the cores, Dai et al. [2021, Section 3.7.1] suggest to choose $\tilde{\boldsymbol{\theta}} \in \mathbb{R}^d$ to be a weighted average of approximate modes of each sub-posterior (which can be preformed in a single pre-processing step) and then modify the proposal distribution for the initial draw to be from

$$\tilde{f}_c\left(\boldsymbol{x}_0^{(c)}\right) \propto \exp\left\{-\frac{\left\|\boldsymbol{x}_0^{(c)} - \tilde{\boldsymbol{\theta}}\right\|^2}{2T}\right\} \cdot f_c\left(\boldsymbol{x}_0^{(c)}\right). \tag{5.29}$$

We must compensate for this modification by replacing the importance weight $\rho_0^{bm}(\cdot)$ with

$$\tilde{\varrho}_0^{bm} := \exp\left\{\frac{C\left\|\bar{\boldsymbol{x}}_0 - \tilde{\boldsymbol{\theta}}\right\|^2}{2T}\right\}, \tag{5.30}$$

where $\bar{\boldsymbol{x}}_0 = \frac{1}{C}\sum_{c=1}^{C} \boldsymbol{x}_0^{(c)}$. We can see that

$$\tilde{\varrho}_0^{bm}(\vec{\boldsymbol{x}}_0) \cdot \prod_{c=1}^{C} \tilde{f}_c\left(\boldsymbol{x}_0^{(c)}\right) \propto \rho_0^{bm}(\vec{\boldsymbol{x}}_0) \cdot \prod_{c=1}^{C} f_c\left(\boldsymbol{x}_0^{(c)}\right),$$

and since we re-normalise the weights, this removes the need to compute the constant of proportionality for $\tilde{\varrho}_0^{bm}$. This modification by Dai et al. [2021] results in an approach whereby we can sample from $\tilde{f}_c$ on each core in isolation (using a rejection sampler with f_c as the proposal density). More critically, the evaluation of the modified importance weights $\tilde{\varrho}_0^{bm}$ does not require any further communication between the cores (since the computation can be done at the central server). This means that we have removed two of the three communications required in the original formulation of the initialisation of the Bayesian Fusion algorithm. However, this approach does require a communication to compute $\tilde{\boldsymbol{\theta}}$. This modified initialisation is summarised in Algorithm 5.2.3.

Algorithm 5.2.3 Particle set initialisation modification (to replace Algorithm 5.2.2 Step 1b). [Dai et al., 2021, Algorithm 2].

1(b) For i in 1 to N,

(i) $\vec{\boldsymbol{x}}_{0,i}$: For $c = 1, \ldots, C$, simulate $\boldsymbol{x}_{0,i}^{(c)} \sim \tilde{f}_c$ (5.29). Set $\vec{\boldsymbol{x}}_{0,i} := \boldsymbol{x}_{0,i}^{(1:C)}$.

(ii) Compute un-normalised weight $w'_{0,i} := \tilde{\varrho}_0^{bm}(\vec{\boldsymbol{x}}_{0,i})$ as per (5.30).

We now focus our attention on the iterative propagation of the particle set in Algorithm 5.2.2 Step 2(b)i. For this step, we must compute $\vec{\boldsymbol{M}}_j := \vec{\boldsymbol{M}}_{t_{j-1},t_j}$ as per (5.17), which requires a communication

between the cores since it requires the position of every trajectory over all cores. Furthermore, after propagating the particle set, there is a second communication to compute the updated importance weights of the particle set. Instead, Dai et al. [2021, Section 3.7.2] suggest an alternative approach where each of the C parallel processes propagate their own individual particles to compose $\vec{\boldsymbol{x}}_j$, which can be achieved by exploiting Corollary 5.2.1.

Corollary 5.2.1. *[Dai et al., 2021, Corollary 2]. Simulating* $\vec{\boldsymbol{x}}_j \sim \mathcal{N}_d\left(\vec{\boldsymbol{M}}_j, \boldsymbol{V}_j\right)$*, the required transition from* $\vec{\boldsymbol{x}}_{j-1}$ *to* $\vec{\boldsymbol{x}}_j$ *in Algorithm 5.2.2 Step 2(b)i, can be expressed as*

$$\boldsymbol{x}_j^{(c)} = \left(\frac{\Delta_j^2}{C(T-t_{j-1})}\right)^{\frac{1}{2}} \boldsymbol{\xi}_j + \left(\frac{T-t_j}{T-t_{j-1}}\right)^{\frac{1}{2}} \boldsymbol{\eta}_j^{(c)} + \boldsymbol{M}_j^{(c)}, \tag{5.31}$$

where $\boldsymbol{\xi}_j$*,* $\boldsymbol{\eta}_j$ *are standard Gaussian vectors and* $\boldsymbol{M}_j^{(c)}$ *is the* c*th sub-vector of* $\vec{\boldsymbol{M}}_j := \vec{\boldsymbol{M}}_{t_{j-1},t_j}$ (5.17)*.*

Proof. See Dai et al. [2021, Appendix D]. This also follows directly from Corollary 7.3.2 by setting $\boldsymbol{\Lambda}_c = \mathbb{I}_d$ for $c \in \mathcal{C} := \{1, \ldots, C\}$. ■

Dai et al. [2021] notes that the interaction between the trajectories only occurs through the mean of the trajectories at the previous iteration. Computation of $\bar{\boldsymbol{x}}_{j-1}$ can be computed at the previous iteration at the same time the trajectory positions are communicated to compute the updated importance weight. Furthermore, the common Gaussian vector $\boldsymbol{\xi}_j$ can also be simulated at the previous iteration and communicated at the same time. This results in an approach where each of the C parallel cores can update the individual components of $\vec{\boldsymbol{x}}_j$ and consequently removes an unnecessary additional communication between the cores. In particular, the trajectories can be propagated separately using the mean of the trajectories which is computed at the previous iteration. To summarise, we can modify Step 2(b)i by substituting in Algorithm 5.2.4.

Algorithm 5.2.4 Particle set propagation modification (to replace Algorithm 5.2.2 Step 2(b)i). [Dai et al., 2021, Algorithm 3].

2(b)i.

(A) For $c = 1, \ldots, C$, simulate $\boldsymbol{x}_{j,i}^{(c)} | (\bar{\boldsymbol{x}}_{j-1,i}, \boldsymbol{x}_{j-1,i}^{(c)})$ in (5.31).

(B) Set $\vec{\boldsymbol{x}}_{j,i} := \boldsymbol{x}_{j,i}^{(1:C)}$, compute $\bar{\boldsymbol{x}}_{j,i} := \frac{1}{C}\sum_{c=1}^{C} \boldsymbol{x}_{j,i}^{(c)}$.

5.2.3.3 Alternative methods for updating the particle set weights

So far, we have assumed that we have been able to compute functionals of each sub-posterior f_c for $c = 1, \ldots, C$, but in many settings it may be impractical or infeasible to do so. This may be due to some form of intractability of the sub-posteriors (e.g. settings considered in Andrieu and Roberts [2009]), or its evaluation may be simply too computationally expensive which could be the case in large data settings [Pollock et al., 2020; Bouchard-Côté et al., 2018; Bierkens et al., 2019; Dai et al.,

2021]. In such scenarios, we no longer are able to evaluate ϕ_c^{dl} which is necessary to update the particle weights in the iterative steps of the BF algorithm. However, Dai et al. [2021, Section 3.7.3] note that it is possible to consider a suitable unbiased estimator for ϕ_c^{dl} and subsequently replacing the unbiased estimator $\tilde{\rho}_j^{bm}$ in Step 2(b)ii. This idea is summarised in the following corollary:

Corollary 5.2.2. *[Dai et al., 2021, Corollary 3] The estimator*

$$\tilde{\varrho}_j^{bm}\left(\vec{\boldsymbol{x}}_{j-1}, \vec{\boldsymbol{x}}_j\right) := \prod_{c=1}^{C}\left(\frac{\Delta_j^{\kappa_c} \cdot e^{-\bar{U}_X^{(c)}\Delta_j}}{\kappa_c! \cdot p\left(\kappa_c | R_c\right)} \cdot \prod_{k_c=1}^{\kappa_c}\left(\bar{U}_X^{(c)} - \tilde{\phi}_c^{dl}\left(\boldsymbol{X}_{\xi_{c,k_c}}^{(c)}\right)\right)\right), \tag{5.32}$$

where $\tilde{\phi}_c^{dl}$ is an unbiased estimator of ϕ_c^{dl} and $\bar{U}_j^{(c)}$ is a constant such that $\tilde{\phi}_c^{dl}(\boldsymbol{x}) \leq \bar{U}_j^{(c)}$ for $\boldsymbol{x} \in R_c$.

Proof. This follows directly from Theorem 5.2.3. ■

The estimator $\tilde{\varrho}_j^{bm}$ in Corollary 5.2.2 can therefore be used as a direct substitute for $\tilde{\rho}_j^{bm}$ in Algorithm 5.2.2 Step 2(b)ii. Dai et al. [2021] highlight that there is a penalty for introducing $\tilde{\varrho}_j^{bm}$ as it typically increases the variance of the estimator, which will result in higher variance in the particle set weights in the BF algorithm. However, as noted, introducing an alternative unbiased estimator may be necessary to apply the BF approach to some settings.

As an example (following the example provided in Dai et al. [2021, Appendix E]), an important application of the Fusion methodology is to perform Bayesian inference with large datasets (see for example Section 1.2). In such scenarios, we will have a large number of data points associated to each sub-posterior. For instance, suppose we have $m_c \gg 1$ data points for core $c = 1, \ldots, C$, then computing ϕ_c^{dl} in (5.6) is an expensive $\mathcal{O}(m_c)$ operation. To utilise Corollary 5.2.2, we simply need to find a suitable unbiased estimator for ϕ_c^{dl}, which in many settings, will be simple since ϕ_c^{dl} is linear in terms terms of $\nabla \log f_c(\boldsymbol{x})$ and $\Delta \log f_c(\boldsymbol{x})$. In particular, consider the case where the sub-posteriors admit a structure with conditional independence and can be factorised as follows:

$$f_c(\boldsymbol{x}) \propto \prod_{i=1}^{m_c} l_{i,c}(\boldsymbol{x}). \tag{5.33}$$

This assumption of this factorisation is fairly weak and many classes of models exhibits such a conditional independence structure [Pollock et al., 2020]. Since ϕ_c^{dl} is linear in terms of $\nabla \log l_{i,c}(\boldsymbol{x})$ and $\Delta \log l_{i,c}(\boldsymbol{x})$, then we can use the following naive unbiased estimator for ϕ_c^{dl} [Dai et al., 2021, Appendix E]:

$$\tilde{\phi}_c^{dl}(\boldsymbol{x}) = \frac{m_c}{2}\left[(\nabla \log l_{I,c}(\boldsymbol{x}))^{\intercal}(\nabla \log l_{J,c}(\boldsymbol{x})) + \Delta \log l_{I,c}(\boldsymbol{x})\right], \tag{5.34}$$

where $I, J \overset{\text{iid}}{\sim} \mathcal{U}\{1, \ldots, m_c\}$. The advantage of using such an estimator is that evaluating this estimator has $\mathcal{O}(1)$ cost. However, a caveat is that when applying Corollary 5.2.2, we must consider the expected number of functional evaluations in computing the unbiased estimator. In particular,

by using Corollary 5.2.2, the number of expected functional evaluations will change from K to K' and so we must consider the growth in the ratio K'/K as $m_c \to \infty$. Dai et al. [2019] notes that whilst (5.34) has $\mathcal{O}(1)$ cost to evaluate, this comes at the cost of an $\mathcal{O}(m_c)$ inflation in the expected number of function evaluations, effectively cancelling any benefit of using (5.34).

However, it is possible to apply the approach of Pollock et al. [2020] to develop an $\mathcal{O}(1)$ unbiased estimator $\tilde{\phi}_c^{dl}$ which also has a $\mathcal{O}(1)$ scaling of the ratio K'/K. In particular, Pollock et al. [2020, Section 4] propose using *control variates* for each sub-posterior and compute $\nabla \log f_c$ and $\Delta \log f_c$ at points *close* to the mode of the sub-posterior, $\hat{\boldsymbol{x}}_c$, or a point *close* to the mode of the target posterior $\hat{\boldsymbol{x}}$ (where '*close*' means within $\mathcal{O}(m_c^{-\frac{1}{2}})$ of the true respective modes). Note that the use of such control variates within subsampling schemes have also been proposed with great success in a number of other works, for instance Bouchard-Côté et al. [2018]; Bierkens et al. [2019]; Baker et al. [2019]. Computing these control variates will typically be one-time $\mathcal{O}(m_c)$ computations. Now let

$$\tilde{\alpha}_{I,c}(\boldsymbol{x}) := n \cdot \big[\nabla \log l_{I,c}(\boldsymbol{x}) - \nabla \log l_{I,c}(\boldsymbol{x}^*)\big], \tag{5.35}$$

then since $\log f_c(\boldsymbol{x}) = \sum_{i=1}^{m_c} \log l_{i,c}(\boldsymbol{x})$ in this setting, we have

$$\mathbb{E}_{\mathcal{A}}\left[\tilde{\alpha}_{I,c}(\boldsymbol{x})\right] = \alpha_c(\boldsymbol{x}), \quad \mathbb{E}_{\mathcal{A}}\left[\operatorname{div} \tilde{\alpha}_{I,c}(\boldsymbol{x})\right] = \operatorname{div} \alpha_c(\boldsymbol{x}), \tag{5.36}$$

where $\alpha_c(\boldsymbol{x}) := \nabla \log f_c(\boldsymbol{x}) - \nabla \log f_c(\boldsymbol{x}^*)$ and $\mathcal{A}$ is the law of $I \sim \mathcal{U}\{1, \ldots, n\}$. By noting that $\phi_c^{dl}(\boldsymbol{x})$ in (5.6) can be re-expressed as

$$\phi_c^{dl}(\boldsymbol{x}) = \frac{1}{2}\big[\alpha_c(\boldsymbol{x})^\intercal(2\nabla \log f_c(\boldsymbol{x}^*) + \alpha_c(\boldsymbol{x})) + \operatorname{div} \alpha_c(\boldsymbol{x})\big] + C^*, \tag{5.37}$$

where $C^* := \frac{1}{2}\left(\|\nabla \log f_c(\boldsymbol{x}^*)\|^2 + \Delta \log f_c(\boldsymbol{x}^*)\right)$, then this leads to the following unbiased estimator for ϕ_c^{dl}:

$$\tilde{\phi}_c^{dl}(\boldsymbol{x}) := \frac{1}{2}\big[\alpha_{I,c}(\boldsymbol{x})^\intercal(2\nabla \log f_c(\boldsymbol{x}^*) + \alpha_{J,c}(\boldsymbol{x})) + \operatorname{div} \alpha_{I,c}(\boldsymbol{x})\big] + C^*, \tag{5.38}$$

where $I, J \overset{\text{iid}}{\sim} \mathcal{U}\{1, \ldots, m_c\}$, i.e. if now we let $\mathcal{A}$ be the law of $I, J \overset{\text{iid}}{\sim} \mathcal{U}\{1, \ldots, m_c\}$, we have $\mathbb{E}_{\mathcal{A}}\left[\tilde{\phi}_c^{dl}(\boldsymbol{x})\right] = \phi_c^{dl}(\boldsymbol{x})$.

The evaluations of the constants $\|\nabla \log f_c(\boldsymbol{x}^*)\|^2$, $\Delta \log f_c(\boldsymbol{x}^*)$ are of $\mathcal{O}(m_c)$ cost, but they only need to be computed once and performed in parallel. Pollock et al. [2020, Theorem 3] showed that under mild assumptions, and in the setting where sub-posteriors contract at the rate $m_c^{-\frac{1}{2}}$, then K'/K grows with data size like $\mathcal{O}(1)$. The unbiased estimator $\tilde{\phi}_c(\boldsymbol{x})$ uses only double draws from $\{1, \ldots, m_c\}$, although Pollock et al. [2020] notes that it would be possible to replace this by averaging over multiple draws (sampling from $\{1, \ldots, m_c\}$ with replacement) which could have advantages of reducing the variance of the estimator at the cost of increasing the number of data points to evaluate at.

Part II

Methodology

Chapter 6

Divide-and-Conquer Generalised Monte Carlo Fusion

We have seen in Section 1.2 and Chapter 5 that the *Fusion* approach to combine (sample approximations of) probability distributions via (1.1) is to construct a direct sample approximation of f itself, rather than seeking to obtain an ad hoc approximation to f. The *Monte Carlo Fusion* (MCF) approach of Dai et al. [2019] (outlined in Section 5.1) is a rejection sampling (see Section 2.2) approach to sample f, and so returns i.i.d. draws from f. The benefit of such approach is that it is readily parallelisable and is exact (in the sense that it avoids any form of approximation error besides Monte Carlo error). However, there are significant limitations with MCF; namely it struggles in certain scenarios such as unifying sub-posteriors which exhibit strong correlation structure, or sub-posteriors which conflict with one another (i.e. sub-posteriors which have little common support), and the approach suffers from a lack of robustness with unifying increasing numbers of sub-posteriors. These limitations are discussed more fully in Section 6.3.

In this chapter, we aim to alleviate some of these issues and begin by reformulating the theory underpinning the MCF approach, and introduce a *Generalised Monte Carlo Fusion (GMCF)* algorithm which is based upon *importance sampling* (see Section 2.3) (as opposed to rejection sampling) and exploits readily available global information about each sub-posterior. The resulting GMCF methodology has far greater robustness to increasing sub-posterior correlation. We next embed our GMCF methodology within a *divide-and-conquer* paradigm by combining the sub-posteriors in stages to recover the correct fusion density f (1.1), in an approach we term *Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF)*. We do this by appealing to the Divide-and-Conquer Sequential Monte Carlo (D&C-SMC) approach of Lindsten et al. [2017] (which we described in Section 3.4), and so as a consequence we are able to take advantage of much of what is known about SMC and this particular variant thereof. The D&C-GMCF approach addresses both the robustness of Fusion approaches with respect to increasing numbers of sub-

posteriors and by means of *tempering* methodology, provides a possible resolution to conflicting sub-posteriors. This is shown empirically by means of both illustrative and realistic benchmarks.

The remainder of this chapter is organised as follows: in Section 6.1 and Algorithm 6.1.2, we present our GMCF approach. In Section 6.2, we embed our Fusion approach within a divide-and-conquer paradigm, and by introducing a D&C-SMC algorithm, address the robustness of Fusion with increasing numbers of sub-posteriors; the use of *tree diagrams/graphs* (see for instance Figures 6.1–6.2) illustrate how the sub-posteriors can be combined in stages in a flexible problem-specific manner. We present illustrative examples to study the robustness of our methodology applied to various scenarios in Section 6.3. In Section 6.4, we present some applications of our methodology and study the performance of our approach in comparison to competing approximate methodologies for combining sub-posterior samples which were outlined Section 1.2.1. We note that the content of this chapter appears in Chan et al. [2021].

6.1 A generalisation of Monte Carlo Fusion

As noted in the introduction of this chapter, we will subsequently embed our Fusion approach within a divide-and-conquer framework whereby sub-posteriors are combined in stages to recover our target fusion density f (1.1). As such, we introduce an index set, denoted by $\mathcal{C}$, representing the sub-posteriors we want to unify. The key observation in the Fusion approach (first proposed in Dai et al. [2019]) is that if we wish to sample from the density $f^{(\mathcal{C})}(\boldsymbol{x}) \propto \prod_{c\in\mathcal{C}} f_c(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^d$ and $f_c(\boldsymbol{x})$ for $c \in \mathcal{C}$, then this is simply a marginal of a density on an extended state space ($g_{\mathcal{C}}$ given in Proposition 6.1.1). For the purposes of simplifying the notation in Proposition 6.1.1, we denote by $\vec{\boldsymbol{x}}^{(\mathcal{C})} \in \mathbb{R}^{|\mathcal{C}|\times d}$ a vector composed of $\left\{\boldsymbol{x}^{(c)}\right\}_{c\in\mathcal{C}} \in \mathbb{R}^d$ (so $\vec{\boldsymbol{x}}^{(\mathcal{C})} := (\boldsymbol{x}^{(c_1)}, \ldots, \boldsymbol{x}^{(c_{|\mathcal{C}|})})$, with c_i denoting the ith element of the index set $\mathcal{C}$). Although not required for Proposition 6.1.1, in the subsequent methodology we develop (in common with Dai et al. [2019]), we assume that for each $c = 1, \ldots, C$, we can evaluate f_c pointwise (up to its normalising constant), f_c is nowhere zero and everywhere differentiable, and that we can compute $A_c(\boldsymbol{x}) := \log f_c(\boldsymbol{x})$, $\nabla A_c(\boldsymbol{x})$, and $\nabla^2 A_c(\boldsymbol{x})$ pointwise (where ∇ is the gradient operator and ∇^2 is the Hessian). However, we note that this is not a limiting factor of the Fusion methodology (as we have seen in Section 5.2.3.3).

Proposition 6.1.1. *Let $\mathcal{C} := (c_1, \ldots, c_{|\mathcal{C}|})$ denote the index set representing the sub-posteriors we wish to unify and suppose that p_c is the transition density of a Markov chain on $\mathbb{R}^d$ with a stationary probability density proportional to f_c^2. Then the $(|\mathcal{C}| + 1)d$-dimensional probability density proportional to the integrable function*

$$g_{\mathcal{C}}\left(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}\right) := \prod_{c\in\mathcal{C}} \left[f_c^2\left(\boldsymbol{x}^{(c)}\right) \cdot p_c\left(\boldsymbol{y}^{(\mathcal{C})} \middle| \boldsymbol{x}^{(c)}\right) \cdot \frac{1}{f_c(\boldsymbol{y}^{(\mathcal{C})})} \right], \tag{6.1}$$

admits marginal density $f^{(\mathcal{C})} \propto \prod_{c\in\mathcal{C}} f_c$ for $\boldsymbol{y}^{(\mathcal{C})} \in \mathbb{R}^d$.

Proof. Following the same approach as in Proposition 5.1.1, by integrating out $\vec{\boldsymbol{x}}^{(\mathcal{C})}$, we have

$$\begin{aligned}\int_{\mathbb{R}_d} \cdots \int_{\mathbb{R}_d} g_{\mathcal{C}}\left(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}\right) \, \mathrm{d}\boldsymbol{x}^{(c_1)} \cdots \, \mathrm{d}\boldsymbol{x}^{(c_{|\mathcal{C}|})} &= \prod_{c\in\mathcal{C}} \left[\int_{\mathbb{R}_d} f_c^2\left(\boldsymbol{x}^{(1)}\right) \cdot p_c\left(\boldsymbol{y}^{(\mathcal{C})} \middle| \boldsymbol{x}^{(c)}\right) \cdot \frac{1}{f_c\left(\boldsymbol{y}^{(\mathcal{C})}\right)} \, \mathrm{d}\boldsymbol{x}^{(c)}\right] \\ &= \prod_{c\in\mathcal{C}} \left[\frac{f_c^2\left(\boldsymbol{y}^{(\mathcal{C})}\right)}{f_c\left(\boldsymbol{y}^{(\mathcal{C})}\right)}\right] \\ &= \prod_{c\in\mathcal{C}} f_c\left(\boldsymbol{y}^{(\mathcal{C})}\right) = f^{(\mathcal{C})}\left(\boldsymbol{y}^{(\mathcal{C})}\right). \end{aligned} \tag{6.2}$$

Hence, $\boldsymbol{y}^{(\mathcal{C})}$ has marginal density $f^{(\mathcal{C})}$. ■

Dai et al. [2019, Proposition 2] (see Proposition 5.1.1) exploited Proposition 6.1.1 by noting that if $\mathcal{C} := \{1, \ldots, C\}$, then we recover the target *fusion* density f (1.1). In the theory we develop in this section, we consider the abstraction to more general index sets, as this facilitates the *divide-and-conquer* approach we introduce in Section 6.2. Since $g_{\mathcal{C}}$ will not typically be accessible directly, Dai et al. [2019] proposed sampling from $g_{\mathcal{C}}$ (with $\mathcal{C} := \{1, \ldots, C\}$) by constructing a suitable $(|\mathcal{C}|+1)d$-dimensional proposal density, $h_{\mathcal{C}}$, for use within a rejection sampling algorithm (outlined in Algorithm 5.1.3), and then simply retaining the $\boldsymbol{y}^{(\mathcal{C})}$ marginal of any accepted draw as a realisation of $f^{(\mathcal{C})}$. Dai et al. [2019] showed that if p_c in Proposition 6.1.1 was chosen to be the transition density of a constant volatility Langevin diffusion (5.3) for time $T > 0$ with invariant measure f_c^2 for each $c \in \mathcal{C}$ respectively, then for a (easily accessible) proposal $h_{\mathcal{C}}$ constructed by sampling a single draw from each sub-posterior ($\boldsymbol{x}^{(c)} \sim f_c$ for $c \in \mathcal{C}$), and a single Gaussian random variable parameterised by the sub-posterior realisations (corresponding to the $\boldsymbol{y}^{(\mathcal{C})}$-marginal), the acceptance probability was readily computable. The full details of the MCF approach are provided in Section 5.1.1.

As noted earlier, one of the principle shortcomings of the rejection sampling MCF approach of Dai et al. [2019] is its lack of robustness with increasing sub-posterior correlation. In this section, we develop theory and methodology to improve upon MCF (even in the setting where $\mathcal{C} := \{1, \ldots, C\}$), by providing a principled way to weight the contribution from each sub-posterior in the construction of the proposal $h_{\mathcal{C}}$, and then embed the approach in more robust Monte Carlo methodologies (such as importance sampling (see Section 2.3) and sequential Monte Carlo (see Chapter 3)). Having the flexibility to weight the contribution from each sub-posterior allows us to incorporate within the algorithm global information for each sub-posterior which will often be readily obtainable, such as information about sub-posterior means or covariance structures ($\hat{\boldsymbol{x}}_c$ and $\hat{\boldsymbol{\Sigma}}_c$ for $c \in \mathcal{C}$ respectively).

6.1.1 Theory

Although MCF is a rejection sampler and outputs independent realisations from (1.1), the computational efficiency of the approach is limited by the quality of the proposal used. The construction of the proposal suggested by Dai et al. [2019] (which is given in (5.5)) closely matches the extended target when each sub-posterior has a covariance structure which is approximately a scaled iden-

tity (i.e. $\hat{\boldsymbol{\Sigma}}_c \approx (1/T)\mathbb{I}_d$ for $c \in \mathcal{C}$ and user chosen constant $T > 0$). However, there is a lack of robustness in the approach (in the sense that the acceptance probabilities rapidly degrade) when the sub-posteriors have non-identity covariance structure. In this section, our aim is to improve the quality of the proposal to address this lack of robustness.

As we assume we are able to simulate directly from each sub-posterior $\boldsymbol{x}^{(c)} \sim f_c$ for $c \in \mathcal{C}$, it is possible to alter the proposal of Dai et al. [2019] via the manner in which the $|\mathcal{C}|$ sub-posterior realisations paramaterise the Gaussian random variable corresponding to the proposed $\boldsymbol{y}^{(\mathcal{C})}$-marginal. Recall from Section 5.1.1, the proposal for the $\boldsymbol{y}^{(\mathcal{C})}$-marginal in the MCF was simulated from $\mathcal{N}_d(\bar{\boldsymbol{x}}, \frac{T}{C}\mathbb{I}_d)$ where $\bar{\boldsymbol{x}}$ denotes the simple average of sub-posterior average. A natural choice for the proposal parameterisation would be one which incorporated global information for each sub-posterior, e.g. using estimated sub-posterior covariance matrices. In particular, we propose the proposal density for $g_{\mathcal{C}}$ (6.1) to be proportional to the following function (where we let $\boldsymbol{\Lambda}_c$ be the user-specified matrices associated with sub-posterior f_c for $c \in \mathcal{C}$):

$$h_{\mathcal{C}}\left(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}\right) := \prod_{c\in\mathcal{C}} \left[f_c\left(\boldsymbol{x}^{(c)}\right)\right] \cdot \exp\left\{-\frac{(\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}^{(\mathcal{C})})^\intercal \boldsymbol{\Lambda}_{\mathcal{C}}^{-1} (\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}^{(\mathcal{C})})}{2T}\right\}, \tag{6.3}$$

where

$$\tilde{\boldsymbol{x}}^{(\mathcal{C})} := \left(\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}\right)^{-1} \left(\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}^{(c)}\right), \qquad \boldsymbol{\Lambda}_{\mathcal{C}}^{-1} := \sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}. \tag{6.4}$$

Simulating from the proposal $h_{\mathcal{C}}$ in (6.3) is possible by first simulating a single draw from each sub-posterior (which are assumed to be accessible) to obtain $\{\boldsymbol{x}^{(c)}\}_{c\in\mathcal{C}}$. These draws are then used to compute the *weighted* average $\tilde{\boldsymbol{x}}^{(\mathcal{C})}$, and then simulate $\boldsymbol{y}^{(\mathcal{C})} \sim \mathcal{N}_d\big(\tilde{\boldsymbol{x}}^{(\mathcal{C})}, T\boldsymbol{\Lambda}_{\mathcal{C}}\big)$ as per (6.4). This *proposal* for $\boldsymbol{y}^{(\mathcal{C})}$ is a Gaussian perturbation of the approach suggested in Consensus Monte Carlo (CMC) method [Scott et al., 2016] (see Section 1.2.1) for combining sub-posterior draws (namely $\tilde{\boldsymbol{x}}^{(\mathcal{C})}$), on a scale commensurate with that of the posterior (modulated by a user-specified parameter T). Now considering the choice of the $|\mathcal{C}|$ stochastic processes on $\mathbb{R}^d$ with stationary densities proportional to f_c^2 for $c \in \mathcal{C}$ (as required in Proposition 6.1.1), we similarly want to exploit the availability of global sub-posterior information. As such, we choose p_c in Proposition 6.1.1 to be the transition density over $[0, T]$ of the following stochastic differential equation:

$$\mathrm{d}\boldsymbol{X}_t^{(c)} = \boldsymbol{\Lambda}_c \nabla \log f_c\Big(\boldsymbol{X}_t^{(c)}\Big)\ \mathrm{d}t + \boldsymbol{\Lambda}_c^{\frac{1}{2}}\ \mathrm{d}\boldsymbol{W}_t^{(c)}, \quad \boldsymbol{X}_0^{(c)} = \boldsymbol{x}^{(c)},\ t \in [0, T], \tag{6.5}$$

where in this case $\boldsymbol{\Lambda}_c$ can be interpreted as a *preconditioning* matrix with $\boldsymbol{\Lambda}_c^{\frac{1}{2}}$ being the (positive semi-definite) square root of $\boldsymbol{\Lambda}_c$ where $\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{\Lambda}_c^{\frac{1}{2}} = \boldsymbol{\Lambda}_c$, and $\boldsymbol{W}_t^{(c)}$ is a d-dimensional Brownian motion. Note that for the purposes of our numerical simulations we used the Schur decomposition.

Having constructed a more informative proposal $h_{\mathcal{C}}$, we now require an expression for the ratio of the target to proposal densities.

Proposition 6.1.2. *Letting* $p_c\left(\boldsymbol{y}^{(\mathcal{C})}|\boldsymbol{x}^{(c)}\right)$ *be the transition density of the diffusion given in* (6.5) *we have*

$$\frac{g_{\mathcal{C}}(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})})}{h_{\mathcal{C}}(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})})} \propto \rho_0\left(\vec{\boldsymbol{x}}^{(\mathcal{C})}\right) \cdot \rho_1\left(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}\right), \tag{6.6}$$

where

$$\rho_0\left(\vec{\boldsymbol{x}}^{(\mathcal{C})}\right) := \exp\left\{-\sum_{c\in\mathcal{C}} \frac{(\tilde{\boldsymbol{x}}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})^\intercal \boldsymbol{\Lambda}_c^{-1} (\tilde{\boldsymbol{x}}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})}{2T}\right\}, \tag{6.7}$$

$$\rho_1\left(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}\right) := \prod_{c\in\mathcal{C}} \mathbb{E}_{\mathbb{W}_{\boldsymbol{\Lambda}_c}}\left[\exp\left\{-\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right) \mathrm{d}t\right\}\right], \tag{6.8}$$

and

$$\phi_c(\boldsymbol{x}) := \frac{1}{2}\left(\nabla \log f_c(\boldsymbol{x})^\intercal \boldsymbol{\Lambda}_c \nabla \log f_c(\boldsymbol{x}) + \mathrm{Tr}\big(\boldsymbol{\Lambda}_c \nabla^2 \log f_c(\boldsymbol{x})\big)\right), \tag{6.9}$$

where $\mathrm{Tr}(\cdot)$ *denotes the trace of a matrix, and* $\mathbb{W}_{\boldsymbol{\Lambda}_c}$ *denotes the law of a Brownian bridge* $\{\boldsymbol{X}_t^{(c)}, t \in [0,T]\}$ *with* $\boldsymbol{X}_0^{(c)} := \boldsymbol{x}^{(c)}$, $\boldsymbol{X}_T^{(c)} := \boldsymbol{y}^{(\mathcal{C})}$ *and covariance matrix* $\boldsymbol{\Lambda}_c$.

Proof. Let $p_c\left(\boldsymbol{y}^{(\mathcal{C})}\middle|\boldsymbol{x}^{(c)}\right)$ be the transition density of a Langevin diffusion with covariance matrix $\boldsymbol{\Lambda}_c$ and invariant measure f_c^2 in the interval $[0,T]$ (as given in (6.5)). From the Dacunha-Castelle representation [Dacunha-Castelle and Florens-Zmirou, 1986, Lemma 1] (see Section 4.3.4), we have

$$\begin{aligned} p_c\left(\boldsymbol{y}^{(\mathcal{C})}\middle|\boldsymbol{x}^{(c)}\right) \propto \frac{f_c(\boldsymbol{y}^{(\mathcal{C})})}{f_c\left(\boldsymbol{x}^{(c)}\right)} \cdot \exp\left\{-\frac{(\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})^\intercal \boldsymbol{\Lambda}_c^{-1} (\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})}{2T}\right\} \\ \times \mathbb{E}_{\mathbb{W}_{\boldsymbol{\Lambda}_c}}\left[\exp\left\{-\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right) \mathrm{d}t\right\}\right]. \end{aligned}$$

The extended fusion target density of (6.1) is proportional to the integrable function

$$\begin{aligned} g_{\mathcal{C}}(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}) = \prod_{c\in\mathcal{C}}\left[f_c\left(\boldsymbol{x}^{(c)}\right)\right] \cdot \exp\left\{-\sum_{c\in\mathcal{C}} \frac{(\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})^\intercal \boldsymbol{\Lambda}_c^{-1} (\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})}{2T}\right\} \\ \cdot \prod_{c\in\mathcal{C}} \mathbb{E}_{\mathbb{W}_{\boldsymbol{\Lambda}_c}}\left[\exp\left\{-\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right) \mathrm{d}t\right\}\right]. \end{aligned}$$

We have

$$\begin{aligned} \sum_{c\in\mathcal{C}} \frac{(\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})^\intercal \boldsymbol{\Lambda}_c^{-1} (\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})}{2T} = \sum_{c\in\mathcal{C}} \frac{(\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}^{(\mathcal{C})})^\intercal \boldsymbol{\Lambda}_c^{-1} (\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}^{(\mathcal{C})})}{2T} \\ + \sum_{c\in\mathcal{C}} \frac{(\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}^{(\mathcal{C})})^\intercal \boldsymbol{\Lambda}_c^{-1} (\tilde{\boldsymbol{x}}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})}{T} + \sum_{c\in\mathcal{C}} \frac{(\tilde{\boldsymbol{x}}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})^\intercal \boldsymbol{\Lambda}_c^{-1} (\tilde{\boldsymbol{x}}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})}{2T}. \end{aligned}$$

We can simplify this further considering the middle term and by noting

$$\sum_{c\in\mathcal{C}} \frac{(\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}^{(\mathcal{C})})^{\intercal}\boldsymbol{\Lambda}_c^{-1}(\tilde{\boldsymbol{x}}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})}{T} = \sum_{c\in\mathcal{C}} \frac{\boldsymbol{y}^{(\mathcal{C})\intercal}(\boldsymbol{\Lambda}_c^{-1}\tilde{\boldsymbol{x}}^{(\mathcal{C})} - \boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}^{(c)}) - \tilde{\boldsymbol{x}}^{(\mathcal{C})\intercal}(\boldsymbol{\Lambda}_c^{-1}\tilde{\boldsymbol{x}}^{(\mathcal{C})} - \boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}^{(c)})}{T}. \tag{6.10}$$

Recalling $\tilde{\boldsymbol{x}}^{(\mathcal{C})} = \left(\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}\right)^{-1}\left(\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}^{(c)}\right)$ and noting that $\left(\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}\right)\tilde{\boldsymbol{x}}^{(\mathcal{C})} = \sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}^{(c)}$, we can see that (6.10) cancels to 0.

Now, considering the proposal density $h_{\mathcal{C}}$ given by (6.3), the ratio of $g_{\mathcal{C}}$ and $h_{\mathcal{C}}$ is given by,

$$\frac{g_{\mathcal{C}}(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})})}{h_{\mathcal{C}}(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})})} \propto \exp\left\{-\sum_{c\in\mathcal{C}} \frac{(\tilde{\boldsymbol{x}}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})^{\intercal}\boldsymbol{\Lambda}_c^{-1}(\tilde{\boldsymbol{x}}^{(\mathcal{C})} - \boldsymbol{x}^{(c)})}{2T}\right\} \cdot \prod_{c\in\mathcal{C}} \mathbb{E}_{\mathbb{W}_{\boldsymbol{\Lambda}_c}}\left[\exp\left\{-\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right)\mathrm{d}t\right\}\right],$$

and so defining ρ_0 and ρ_1 as in the statement of Proposition 6.1.2, we arrive at the result. ■

6.1.2 Methodology

It would be possible to develop Monte Carlo methodology based directly upon Propositions 6.1.1–6.1.2 provided we could simulate from $h_{\mathcal{C}}$ and compute the quantities ρ_0 and ρ_1. Viewed from an importance sampling perspective (see Section 2.3), Proposition 6.1.2 allows us to use ρ_0 and ρ_1 as importance weights. The quantity ρ_0 in (6.7) in essence penalises proposals based on the distance the weighted average of the sub-posterior draws are from one another, whereas ρ_1 penalises how far the proposal $\boldsymbol{y}^{(\mathcal{C})}$ is from each sub-posterior draw under the corresponding sub-posterior. Considering the un-normalised importance weight, the computation of ρ_0 is direct, whereas computing ρ_1 in (6.8) is not direct as it involves the evaluation of path integrals of functionals of Brownian bridges. As we have seen in Chapter 5, this is the key complication in implementing Fusion methodologies.

We have seen that it is possible to unbiasedly estimate quantities similar to ρ_1 by application of the path-space rejection sampling methodology introduced in Section 4.4; in particular, we utilise the Poisson estimators discussed in Section 4.4.3 to simulate positive unbiased estimators for functions of the form $\exp\{-\int_0^T \phi_c(\boldsymbol{X}_t^{(c)})\,\mathrm{d}t\}$ where $\boldsymbol{X}_t^{(c)} \sim \mathbb{W}_{\boldsymbol{\Lambda}_c}$ for $c \in \mathcal{C}$. However, note that unlike the unbiased estimators constructed in Chapter 5 (recall we constructed unbiased estimators for Q^{bm} (5.9) in Algorithm 5.1.2, and ρ_j^{bm} (5.22) in Algorithm 5.2.1), ρ_1 in (6.8) involves evaluation of path integrals of functionals of Brownian bridges *with covariance matrices* $\boldsymbol{\Lambda}_c$ for $c \in \mathcal{C}$. In contrast, in the unbiased estimators introduced in Chapter 5, we had $\boldsymbol{\Lambda}_c = \mathbb{I}_d$. The benefit of having $\boldsymbol{\Lambda}_c \neq \mathbb{I}_d$ is that when applying the methodology from Section 4.4, the proposal paths from $\mathbb{W}_{\boldsymbol{\Lambda}_c}$ have a covariance structure which is more closely matched to that of f_c^2, the invariant distribution of the

target diffusion measure given by (6.5). This means that we have far more efficient simulation of ρ_1, especially in cases when the sub-posteriors exhibit high correlation, since our proposal paths will more likely move in directions of high probability density. This approach of using Brownian bridges with covariance $\mathbf{\Lambda}_c$ for $c \in \mathcal{C}$ is equivalent to employing a preconditioning transformation to roughly equate the scales for different components (see e.g. Pollock et al. [2016, Section 4.3]).

For the purposes of the methodology we will subsequently develop, it is sufficient to find a non-negative unbiased estimator of ρ_1 with finite variance which can be obtained with finite cost. To do so, we can apply the methodology outlined in Section 4.4.1 and Section 4.4.3. To utilise this methodology, we require for a given sample path $\boldsymbol{X}_{[0,T]}^{(c)} \sim \mathbb{W}_{\mathbf{\Lambda}_c}$ that we have bounds on ϕ_c for each $c \in \mathcal{C}$. In general, it is not possible to find global bounds for ϕ_c, and so following the approach of Beskos et al. [2006a] and Pollock et al. [2016], we simulate Brownian bridges in such a way that we can determine a compact region which almost surely constrains a given path (the details of which appear in Pollock et al. [2016, Sections 8.1 and 8.5] and are discussed in Section 4.2.2), which enables us to instead find *local bounds* on ϕ_c. In particular, we let $R_c := R_c\big(\boldsymbol{X}_{[0,T]}^{(c)}\big)$ denote a compact subset of $\mathbb{R}^d$ for which $\boldsymbol{X}_t^{(c)}$ is constrained in time $[0, T]$. We note that it is possible to partition the sample space into disjoint sets and simulate from an associated distribution function (without having to sample the underlying path), $R_c \sim \mathcal{R}_c$ conditional on the user specified partitioning of the space. Although it is possible to find tight local bounds for ϕ_c in a problem specific manner, it is helpful for practitioners to note that it is possible to find generic (less tight) bounds (which we denote by $L_X^{(c)}$ and $U_X^{(c)}$ respectively):

Proposition 6.1.3. *For all $c \in \mathcal{C}$ and $\boldsymbol{x} \in R_c$, we have $\phi_c(\boldsymbol{x}) \in \left[L_X^{(c)}, U_X^{(c)}\right]$, where*

$$L_X^{(c)} := -\frac{1}{2}\left(d \cdot P^{\mathbf{\Lambda}_c}\right), \tag{6.11}$$

$$U_X^{(c)} := \frac{1}{2}\left[\left(\left\|\mathbf{\Lambda}_c^{\frac{1}{2}} \nabla \log f_c\left(\hat{\boldsymbol{x}}^{(c)}\right)\right\| + \max_{\boldsymbol{x} \in R_c}\left\|\mathbf{\Lambda}_c^{-\frac{1}{2}}\left(\boldsymbol{x} - \hat{\boldsymbol{x}}^{(c)}\right)\right\| \cdot P^{\mathbf{\Lambda}_c}\right)^2 + d \cdot P^{\mathbf{\Lambda}_c}\right], \tag{6.12}$$

where d denotes the dimension of $\boldsymbol{x}$, $\|\cdot\|$ is the Euclidean norm, $\hat{\boldsymbol{x}}^{(c)}$ is a user-specified point central to R_c, and where $P^{\mathbf{\Lambda}_c}$ is a quantity such that

$$P^{\mathbf{\Lambda}_c} \geq \max_{\boldsymbol{x} \in R_c} \gamma\left(\mathbf{\Lambda}_c \nabla^2 \log f_c(\boldsymbol{x})\right), \tag{6.13}$$

where γ denotes the matrix norm*, defined as*

$$\gamma(A) := \max_{\|\boldsymbol{x}\| \neq 0} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|}. \tag{6.14}$$

Proof. First note that we can rewrite (6.9) as follows,

$$\phi_c(\boldsymbol{x}) = \frac{1}{2}\left(\left\|\boldsymbol{\Lambda}_c^{\frac{1}{2}}\nabla \log f_c(\boldsymbol{x})\right\|^2 + \operatorname{Tr}\big(\boldsymbol{\Lambda}_c\nabla^2 \log f_c(\boldsymbol{x})\big)\right). \tag{6.15}$$

Let $R_c := R_c\big(\boldsymbol{X}^{(c)}_{[0,T]}\big)$ denote a compact subset of $\mathbb{R}^d$ for which $\boldsymbol{X}^{(c)}_t$ is constrained in time $[0,T]$ for $c \in \mathcal{C}$, then to bound the first term in (6.15), we first use the triangle inequality by noting

$$\begin{aligned}\max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{\frac{1}{2}}\nabla \log f_c(\boldsymbol{x})\right\| &= \max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{\frac{1}{2}}\nabla \log f_c(\hat{\boldsymbol{x}}^{(c)}) + \boldsymbol{\Lambda}_c^{\frac{1}{2}}\left(\nabla \log f_c(\boldsymbol{x}) - \nabla \log f_c(\hat{\boldsymbol{x}}^{(c)})\right)\right\| \\ &\leq \left\|\boldsymbol{\Lambda}_c^{\frac{1}{2}}\nabla \log f_c(\hat{\boldsymbol{x}}^{(c)})\right\| + \max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{\frac{1}{2}}\left(\nabla \log f_c(\boldsymbol{x}) - \nabla \log f_c(\hat{\boldsymbol{x}}^{(c)})\right)\right\|, \end{aligned} \tag{6.16}$$

where $\hat{\boldsymbol{x}}^{(c)}$ is a user-specified point in $\mathbb{R}^d$. Focusing now on bounding the second term of (6.16), then we express this as a line integral between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}^{(c)}$ so

$$\max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{\frac{1}{2}}\left(\nabla \log f_c(\boldsymbol{x}) - \nabla \log f_c(\hat{\boldsymbol{x}}^{(c)})\right)\right\| = \max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{-\frac{1}{2}}\int_{\mathbf{0}}^{\left\|\boldsymbol{x}-\hat{\boldsymbol{x}}^{(c)}\right\|}\boldsymbol{\Lambda}_c\nabla^2 \log f(\boldsymbol{x}+u\boldsymbol{n})\boldsymbol{n}\,\mathrm{d}u\right\|,$$

where $\boldsymbol{u} = \boldsymbol{x} + u\boldsymbol{n}$, where $\boldsymbol{n}$ is a unit-vector with $\|\boldsymbol{n}\| = 1$. We have

$$\begin{aligned}\max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{-\frac{1}{2}}\int_{\mathbf{0}}^{\left\|\boldsymbol{x}-\hat{\boldsymbol{x}}^{(c)}\right\|}\boldsymbol{\Lambda}_c\nabla \log f(\boldsymbol{x}+u\boldsymbol{n})\boldsymbol{n}\,\mathrm{d}u\right\| &\leq \max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{-\frac{1}{2}}\left(\boldsymbol{x}-\hat{\boldsymbol{x}}^{(c)}\right)\right\| \\ &\quad\cdot \sup_{\boldsymbol{n};\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c\nabla^2 \log f(\boldsymbol{x}+u\boldsymbol{n})\boldsymbol{n}\right\| \\ &\leq \max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{-\frac{1}{2}}\left(\boldsymbol{x}-\hat{\boldsymbol{x}}^{(c)}\right)\right\|\cdot P^{\boldsymbol{\Lambda}_c}, \end{aligned} \tag{6.17}$$

where $P^{\boldsymbol{\Lambda}_c}$ is defined in (6.13). Putting together (6.16) and (6.17), we have

$$\max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{\frac{1}{2}}\nabla \log f_c(\boldsymbol{x})\right\| \leq \left\|\boldsymbol{\Lambda}_c^{\frac{1}{2}}\nabla \log f_c\left(\hat{\boldsymbol{x}}^{(c)}\right)\right\| + \max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{-\frac{1}{2}}\left(\boldsymbol{x}-\hat{\boldsymbol{x}}^{(c)}\right)\right\|\cdot P^{\boldsymbol{\Lambda}_c}.$$

Since for a matrix $A \in \mathbb{R}^d$, $\operatorname{Tr}(A) \leq d\cdot\gamma(A)$, we can bound the second term in (6.15) as follows:

$$\max_{\boldsymbol{x}\in R_c}\left|\operatorname{Tr}\big(\boldsymbol{\Lambda}_c\nabla^2 \log f_c(\boldsymbol{x})\big)\right| \leq d\cdot P^{\boldsymbol{\Lambda}_c},$$

and hence we can bound ϕ_c as follows:

$$\max_{\boldsymbol{x}\in R_c}|\phi_c(\boldsymbol{x})| \leq \frac{1}{2}\left[\left(\left\|\boldsymbol{\Lambda}_c^{\frac{1}{2}}\nabla \log f_c\left(\hat{\boldsymbol{x}}^{(c)}\right)\right\| + \max_{\boldsymbol{x}\in R_c}\left\|\boldsymbol{\Lambda}_c^{-\frac{1}{2}}\left(\boldsymbol{x}-\hat{\boldsymbol{x}}^{(c)}\right)\right\|\cdot P^{\boldsymbol{\Lambda}_c}\right)^2 + d\cdot P^{\boldsymbol{\Lambda}_c}\right].$$

Noting that in (6.15) that the first term is squared, then the lower and upper bounds of $\phi_c(\boldsymbol{x})$ for $\boldsymbol{x} \in R_c$ are given by (6.11) and (6.12) respectively. ∎

Once local bounds for ϕ_c are obtained, it is possible to unbiasedly estimate ρ_1 using auxiliary diffusion bridge path-space samplers developed in (for instance) Beskos et al. [2006b,a]; Fearnhead et al. [2008]; Pollock et al. [2016]. In particular,

$$\begin{aligned}\rho_1 &= \mathbb{E}\left[\mathbb{E}\left[\mathbb{E}\left[\mathbb{E}\left[\tilde{\rho}_1 \mid \{\mathcal{R}_c, \boldsymbol{X}^{(c)}_{[0,T]}, \kappa_c\}_{c\in\mathcal{C}}\right] \mid \{\mathcal{R}_c, \boldsymbol{X}^{(c)}_{[0,T]}\}_{c\in\mathcal{C}},\right] \mid \{\mathcal{R}_c\}_{c\in\mathcal{C}}\right]\right] \\ &= \mathbb{E}_{\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{W}}|\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{K}}}\mathbb{E}_{\bar{\mathbb{U}}}\left[\tilde{\rho}_1\right], \end{aligned} \tag{6.18}$$

where the subscript for each expectation denotes the law with which we are taking the expectation, $\bar{\mathcal{R}}$ denotes the law of $\{R_c \sim \mathcal{R}_c : c \in \mathcal{C}\}$, $\bar{\mathbb{W}}$ denotes the law of the $|\mathcal{C}|$ Brownian bridges (conditional on the $|\mathcal{C}|$ distinct starting points, but with common endpoint $\boldsymbol{y}^{(\mathcal{C})}$) of $\{\boldsymbol{X}^{(c)}_{[0,T]} \sim \mathbb{W}_{\boldsymbol{\Lambda}_c} : c \in \mathcal{C}\}$, $\bar{\mathbb{K}}$ denotes the law of $\{\kappa_c : c \in \mathcal{C}\}$ where κ_c is a discrete random variable with conditional probabilities $\mathbb{P}[\kappa_c = k_c|R_c] := p(\kappa_c|R_c)$, $\bar{\mathbb{U}}$ denotes the law of $\{\xi_{c,1}, \ldots, \xi_{c,\kappa_c} : c \in \mathcal{C}\} \overset{\text{iid}}{\sim} \mathcal{U}[0,T]$, and with

$$\tilde{\rho}_1(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}) = \prod_{c\in\mathcal{C}}\left(\frac{T^{\kappa_c}\cdot e^{-U_X^{(c)}T}}{\kappa_c!\cdot p\left(\kappa_c|R_c\right)}\cdot\prod_{k_c=1}^{\kappa_c}\left(U_X^{(c)} - \phi_c\left(\boldsymbol{X}^{(c)}_{\xi_{c,k_c}}\right)\right)\right). \tag{6.19}$$

Corollary 6.1.1. *$\tilde{\rho}_1$ is a positive unbiased estimator of ρ_1.*

Proof. Follows directly from Theorem 7.1.3 by setting $j = 1$ and $\Delta_j = T$. ■

Given $L_X^{(c)}$ and $U_X^{(c)}$ which bound ϕ_c as per Proposition 6.1.3, as discussed in Section 4.4.3, there are two natural choices of unbiased estimator for ρ_1 which we denote $\tilde{\rho}_1^{(a)}$ and $\tilde{\rho}_1^{(b)}$ (based, respectively, upon the GPE-1 and GPE-2 estimators of Fearnhead et al. [2008] (see Section 4.4.3)):

Definition 6.1.1. (GPE-1 for ρ_1 (6.8)): Choosing the law of $\kappa_c \sim \text{Poi}\big((U_X^{(c)} - L_X^{(c)})T\big)$ for $c \in \mathcal{C}$ leads to the following unbiased estimator of ρ_1:

$$\tilde{\rho}_1^{(a)}(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}) := \prod_{c\in\mathcal{C}}\left(e^{-L_X^{(c)}T}\cdot\prod_{k_c=1}^{\kappa_c}\left[\frac{U_X^{(c)} - \phi_c\left(\boldsymbol{X}^{(c)}_{\xi_{c,k_c}}\right)}{U_X^{(c)} - L_X^{(c)}}\right]\right). \tag{6.20}$$

Definition 6.1.2. (GPE-2 for ρ_1 (6.8)): Choosing the law of $\kappa_c \sim \text{NB}(\gamma_c, \beta_c)$ for $c \in \mathcal{C}$ with

$$\gamma_c := U_X^{(c)}T - \int_0^T \phi_c\left(\boldsymbol{x}^{(c)}\frac{T-s}{T} + \boldsymbol{y}^{(\mathcal{C})}\frac{s}{T}\right)\mathrm{d}s, \tag{6.21}$$

leads to the following unbiased estimator of ρ_1:

$$\tilde{\rho}_1^{(b)}(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}) := \prod_{c\in\mathcal{C}}\left(e^{-U_X^{(c)}T}\cdot\frac{T^{\kappa_c}\cdot\Gamma(\beta_c)\cdot(\beta_c+\gamma_c)^{\beta_c+\kappa_c}}{\Gamma(\beta_c+\kappa_c)\beta_c^{\beta_c}\gamma_c^{\kappa_c}}\cdot\prod_{k_c=1}^{\kappa_c}\left[U_X^{(c)} - \phi_c\left(\boldsymbol{X}^{(c)}_{\xi_{c,k_c}}\right)\right]\right). \tag{6.22}$$

Computing $\tilde{\rho}_1^{(a)}$ and $\tilde{\rho}_1^{(b)}$ in the case where $\boldsymbol{\Lambda}_c = \mathbb{I}_d$ is detailed explicitly in Dai et al. [2021, Appendix B] and in Chapter 4. In the case where $\boldsymbol{\Lambda}_c \neq \mathbb{I}_d$, we simulate layers by appealing to a suitable transformation. In particular, we transform the start and end points of the Brownian bridge with transformation matrix $\boldsymbol{\Lambda}_c^{-\frac{1}{2}}$, letting $\boldsymbol{z}_0^{(c)} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{x}^{(c)}$ and $\boldsymbol{z}_T^{(c)} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{y}^{(\mathcal{C})}$. The resulting Brownian bridge sample path, $\boldsymbol{z}_t^{(c)} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{X}_t^{(c)}$, has identity covariance structure and thus we can use existing methods for simulating *layered* Brownian bridge sample paths $\boldsymbol{z}_t^{(c)}$ with law $\mathbb{W}_{\mathbb{I}_d}$ from $\boldsymbol{z}_0^{(c)}$ to $\boldsymbol{z}_T^{(c)}$ (see for instance Section 4.2.2). We are then able with minimal modification to apply the approach of Dai et al. [2021] to simulate an unbiased estimator for ρ_1, as given in Algorithm 6.1.1.

Algorithm 6.1.1 Simulating $\tilde{\rho}_1$ [Chan et al., 2021, Algorithm 3].

1. For $c \in \mathcal{C}$
   (a) $\boldsymbol{z}_0^{(c)}, \boldsymbol{z}_0^{(c)}$: Transform the path, setting $\boldsymbol{z}_0^{(c)} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{x}^{(c)}$, and $\boldsymbol{z}_T^{(c)} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{y}^{(\mathcal{C})}$.
   (b) R_c: Set $R_c := \boldsymbol{\Lambda}_c^{\frac{1}{2}} R_c^{(z)}$, where $R_c^{(z)} \sim \mathcal{R}_c^{(z)}$ as per Algorithm 4.2.4.
   (c) $L_X^{(c)}, U_X^{(c)}$: Compute lower and upper bounds, $L_X^{(c)}$ and $U_X^{(c)}$, of $\phi_c(\boldsymbol{x})$ for $\boldsymbol{x} \in R_c$ (as per (6.11) and (6.12), or otherwise).
   (d) p_c: Choose $p(\cdot|R_c)$ using either GPE-1 (Condition 6.1.1) or GPE-2 (Condition 6.1.2).
   (e) κ_c, ξ: Simulate $\kappa_c \sim p(\cdot|R_c)$, and simulate $\xi_{c,1}, \ldots, \xi_{c,\kappa_c} \sim \mathcal{U}[0,T]$.
   (f) $\boldsymbol{z}^{(c)}$: Simulate $\boldsymbol{z}_{\xi_{c,1}}^{(c)}, \ldots, \boldsymbol{z}_{\xi_{c,\kappa_c}}^{(c)} \sim \mathbb{W}_{\mathbb{I}_d} | R_c^{(z)}$ as per Algorithm 4.2.5.
   (g) $\boldsymbol{X}^{(c)}$: Reverse transform the path, setting $\boldsymbol{X}_{\xi_{c,k_c}}^{(c)} = \boldsymbol{\Lambda}_c^{\frac{1}{2}} \boldsymbol{z}_{\xi_{c,k_c}}^{(c)}$ for $k_c = 1, \ldots, \kappa_c$.
2. Output $\tilde{\rho}_1^{(\cdot)}$ (6.19).

In Algorithm 6.1.1 Step 1c, we compute the lower and upper bounds on $\phi_c(\boldsymbol{x})$ for $\boldsymbol{x} \in R_c$ which can be computed as per Proposition 6.1.3. To use these general bounds, we must find a upper bound on the matrix norm of $\boldsymbol{\Lambda}_c \nabla^2 \log f_c(\boldsymbol{x})$ for $\boldsymbol{x} \in R_c$ (i.e. find $P^{\boldsymbol{\Lambda}_c}$ given in (6.13)), which can be done by computing the matrix norm of the matrix which bounds the matrix $\boldsymbol{\Lambda}_c \nabla^2 \log f_c(\boldsymbol{x})$ element-wise. We note that in some cases, it may be easier to upper bound the matrix norm of the Hessian of the *transformed* sub-posterior, $f_c^{(z)}(\boldsymbol{z})$ where $\boldsymbol{z} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{x}$. In particular, we can focus on

$$P^{\boldsymbol{\Lambda}_c} \geq \max_{\boldsymbol{z} \in R_c^{(z)}} \gamma \left(\nabla^2 \log f_c^{(z)}(\boldsymbol{z}) \right), \tag{6.23}$$

which is equivalent to finding the bound in (6.13). In Algorithm 6.1.1 Step 1b, we compute the layer information $R_c^{(z)}$ and so we can directly use this to find local element-wise bounds $\nabla^2 \log f_c^{(z)}(\boldsymbol{z})$ for $\boldsymbol{z} \in R_c^{(z)}$. Therefore, to find $P^{\boldsymbol{\Lambda}_c}$, we just need to find bounds on the second order derivatives of the log-sub-posterior in the transformed space $\boldsymbol{z} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{x}$ so that we can compute the matrix norm of the matrix which bounds $\nabla^2 \log f_c^{(z)}(\boldsymbol{z})$ element-wise. Some example calculations of using the general bounds found in Proposition 6.1.3 can be found in Appendices B.5–B.7.

In developing a rejection sampling approach to sampling from $f^{(\mathcal{C})}$ by means of the extended density $g_{\mathcal{C}}$ with our proposal $h_{\mathcal{C}}$ (as suggested by Proposition 6.1.1), it is natural to select the estimator $\tilde{\rho}_1^{(a)}$ as it is bounded. Indeed, the theory developed in Section 6.1.1 admits the earlier work of Dai et al.

[2019] as a special case, as established in the following corollary. Although there is an advantage in using the broader theory of Section 6.1.1 (with $\boldsymbol{\Lambda}_c \neq \mathbb{I}_d$ for $c = 1, \ldots, C$) to support an improved rejection sampling approach. We omit details of this as it is a minor modification of MCF.

Corollary 6.1.2. *Setting $\boldsymbol{\Lambda}_c = \mathbb{I}_d$ for $c \in \mathcal{C} := \{1, \ldots, C\}$, where $\mathbb{I}_d$ is the identity matrix of dimension d and accepting a proposal $\boldsymbol{y}^{(\mathcal{C})}$ as a sample from (1.1) with probability $(\rho_0 \cdot \tilde{\rho}_1^{(a)})(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}) \cdot \exp\{\sum_{c \in \mathcal{C}} \boldsymbol{\Phi}_c T\}$, we recover the Monte Carlo Fusion approach of Dai et al. [2019].*

In the subsequent section, we embed our approach within the Divide-and-Conquer Sequential Monte Carlo (D&C-SMC) framework of Lindsten et al. [2017] to address the robustness of Fusion with increasing numbers of sub-posteriors. In this setting, we are not restricted to obtaining independent realisations of $f^{(\mathcal{C})}$ in (1.1), and so we can instead employ the estimator $\tilde{\rho}_1^{(b)}$ (as we no longer require our estimator to be bounded above by a constant), which has better asymptotic properties and still has finite variance [Fearnhead et al., 2008]. A (self-normalised) importance sampling approach would proceed as follows: approximate $g_{\mathcal{C}}$ in Proposition 6.1.1 by sampling from the proposal density $h_{\mathcal{C}}$ (i.e. $(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}) \sim h_{\mathcal{C}}$) as given in (6.3). Thus far, we have assumed we have access to i.i.d. draws from each sub-posterior, but this is not necessary for our methodology. In particular, in many settings (including that in Section 6.2) we will instead have access to importance weighted draws from each sub-posterior (with weights $w^{(c)}$ for $c \in \mathcal{C}$ for instance, although in the simplest setting $w^{(c)} \propto 1$ for $c \in \mathcal{C}$). As such, the proposals are then assigned an un-normalised importance weight $w'(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}) := (\prod_{c \in \mathcal{C}} w^{(c)}) \cdot (\rho_0 \cdot \tilde{\rho}_1)(\vec{\boldsymbol{x}}^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})})$.

In practice, and to better relate to our methodology developed later in Section 6.2, we adopt a variant of the importance sampling approach described above. We term the following approach *Generalised Monte Carlo Fusion (GMCF)*, and summarise it in Algorithm 6.1.2. We assume for simplicity in presenting our algorithm that we have access to M importance weighted samples from each sub-posterior (each of which form a Monte Carlo representation for their respective sub-posterior). In addition, we note that the marginal importance sampling weight ρ_0 in (6.7) only depends upon the sub-posterior realisations (and not $\boldsymbol{y}^{(\mathcal{C})}$). To exploit this fact, we compose M *partial proposals* by pairing the draws from each sub-posterior $\{\vec{\boldsymbol{x}}_{0,j}^{(\mathcal{C})}\}_{j=1}^M$, and compute the associated partial weights $\{\vec{w}_j^{(\mathcal{C})}\}_{j=1}^M$ where $\vec{w}_j^{(\mathcal{C})} := (\prod_{c \in \mathcal{C}} w_j^{(c)}) \cdot \rho_0(\vec{\boldsymbol{x}}_{0,j}^{(\mathcal{C})})$ for $j = 1, \ldots, M$. We then sample with replacement N times from this collection of M draws with associated partial weights $\vec{w}_j^{(\mathcal{C})}$. For each of the N partial proposals, we then *complete the proposal* by simulating a corresponding endpoint $\boldsymbol{y}_i^{(\mathcal{C})}$. Taking account of our earlier sampling, each of the N particles will be given (an appropriately normalised) weight proportional to $\rho_1(\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})}, \boldsymbol{y}_i^{(\mathcal{C})})$. By retaining for each of the N weighted particles the marginal for $\boldsymbol{y}^{(\mathcal{C})}$ (i.e. $\{\boldsymbol{y}_i^{(\mathcal{C})}, w_i^{(\mathcal{C})}\}_{i=1}^N$), we simply have an approximation to the desired distribution,

$$f^{(\mathcal{C})}(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \approx \sum_{i=1}^{N} w_i^{(\mathcal{C})} \cdot \delta_{\boldsymbol{y}_i^{(\mathcal{C})}}(\mathrm{d}\boldsymbol{y}). \tag{6.24}$$

Algorithm 6.1.2 `gmcf`$(\mathcal{C}, \{\{\boldsymbol{x}_{0,i}^{(c)}, w_i^{(c)}\}_{i=1}^M, \boldsymbol{\Lambda}_c\}_{c\in\mathcal{C}}, N, T)$: Generalised Monte Carlo Fusion (GMCF) [Chan et al., 2021, Algorithm 1].

1. **Input:** Importance weighted realisations $\{\boldsymbol{x}_{0,i}^{(c)}, w_i^{(c)}\}_{i=1}^M$ for $c \in \mathcal{C}$, the user-specified matrices, $\{\boldsymbol{\Lambda}_c : c \in \mathcal{C}\}$, the number of particles required, N, and time horizon $T > 0$.
2. **Partial proposal:** Compose the importance weighted realisations $\{\vec{\boldsymbol{x}}_{0,j}^{(\mathcal{C})}, \vec{w}_j^{(\mathcal{C})}\}_{j=1}^M$ where $\vec{w}_j^{(\mathcal{C})} := (\prod_{c\in\mathcal{C}} w_j^{(c)}) \cdot \rho_0(\vec{\boldsymbol{x}}_{0,j}^{(\mathcal{C})})$ for $j = 1, \ldots, M$, as per (6.7).
3. For i in 1 to N,
   (a) $\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})}$**:** Sample $I \sim$ categorical$(\vec{w}_1, \ldots, \vec{w}_M)$ and set $\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})} := \vec{\boldsymbol{x}}_{0,I}^{(\mathcal{C})}$.
   (b) **Complete proposal:** Simulate $\boldsymbol{y}_i^{(\mathcal{C})} \sim \mathcal{N}_d\big(\tilde{\boldsymbol{x}}_i^{(\mathcal{C})}, T\boldsymbol{\Lambda}_{\mathcal{C}}\big)$ as per (6.4).
   (c) $\tilde{\rho}_{1,i}^{(\mathcal{C})}$**:** Compute importance weight $\tilde{\rho}_{1,i}^{(\mathcal{C})} := \tilde{\rho}_1^{(b)}\big(\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})}, \boldsymbol{y}_i^{(\mathcal{C})}\big)$ as per Definition 6.1.2.
4. $w_i^{(\mathcal{C})}$**:** For i in 1 to N compute normalised weight $w_i^{(\mathcal{C})} = \tilde{\rho}_{1,i}^{(\mathcal{C})} / \sum_{k=1}^N \tilde{\rho}_{1,k}^{(\mathcal{C})}$.
5. **Output:** $\left\{\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})}, \boldsymbol{y}_i^{(\mathcal{C})}, w_i^{(\mathcal{C})}\right\}_{i=1}^N$, where $\hat{f}^{(\mathcal{C})}(\mathrm{d}\boldsymbol{y}) := \sum_{i=1}^N w_i^{(\mathcal{C})} \cdot \delta_{\boldsymbol{y}_i^{(\mathcal{C})}}(\mathrm{d}\boldsymbol{y}) \approx f^{(\mathcal{C})}(\boldsymbol{y})\mathrm{d}\boldsymbol{y}$.

Of course, in general in the Input step of Algorithm 6.1.2, we may have access to different numbers of samples from each sub-posterior: say M_c importance weighted samples for sub-posterior f_c (for $c \in \mathcal{C}$). In order to compose our M partial proposals in Step 2, there are a number of approaches we could take. As presented above, if $M_c = M$ for $c \in \mathcal{C}$, we simply pair the sub-posterior draws index-wise. This is a basic merging strategy of the sub-posterior realisations and has the advantage that it can be implemented in $O(M)$ cost (and if $M_c \neq M$ for every $c \in \mathcal{C}$ one could simply sub-sample to obtain a common number of samples from each sub-posterior). However, as noted in Lindsten et al. [2017], while this approach has a low computational cost, it can lead to high variance when the product $\prod_{c\in\mathcal{C}} f_c(\boldsymbol{x}^{(c)})$ differs substantially from the corresponding marginal of $f^{(\mathcal{C})}$ — which one might expect to be the case if the sub-posteriors disagree.

We found this simple approach more than adequate in our simulations, but there are more sophisticated options available should they be required in still more challenging settings. In particular, as described in Lindsten et al. [2017, Section 4.1], at the expense of a computational cost $O(\prod_{c\in\mathcal{C}} M_c)$, one could instead compose all possible permutations of the samples from each sub-posterior before weighting and then resampling to reduce the number of points in the approximation back to a pre-specified number, arriving at a better approximation at a greater cost. They termed this approach *mixture resampling* and also detailed a *lightweight mixture resampling* approach in which more than one permutation, but not all possible permutations, are used and found it to work well; as noted by Kuntz et al. [2021a] such a strategy can be connected directly with the theory of incomplete U-statistics and consequently one might hope to realise much of the benefit of mixture resampling at a much reduced cost (see e.g. Kong and Zheng [2021]). Although Step 3a corresponds to a multinomial resampling of the partially composed proposals, an approach we followed in the interest of simplicity, we can of course use other resampling methods and might expect better performance by choosing a lower-variance approach (see for instance Doucet et al. [2001]; Douc et al. [2005]; Gerber et al. [2019] for an investigation of the properties of many such resampling schemes).

6.2 A divide-and-conquer approach to Fusion

A key drawback of the Monte Carlo Fusion approach of Dai et al. [2019] is that it lacks robustness with increasing number of sub-posteriors, $|\mathcal{C}|$. This is unsurprising as the extended target and proposal densities ($g_{\mathcal{C}}$ and $h_{\mathcal{C}}$) of Proposition 6.1.1 are $(|\mathcal{C}|+1)d$-dimensional, and these become increasingly mismatched with increasing dimension. In particular, as a consequence of the definition of ρ_1 in (6.8) of Proposition 6.1.2, the acceptance probability of any rejection-based scheme will decrease geometrically with increasing $|\mathcal{C}|$.

As presented both in Chapter 5 and Section 6.1, the Fusion methodologies discussed so far are examples of a *fork-and-join* approach which unifies all of the sub-posteriors in a single step (similarly, the approximate methods discussed in Sections 1.2.1–1.2.2 are also fork-and-join approaches). In particular, within the GMCF framework Section 6.1, we set $\mathcal{C} := \{1, \ldots, C\}$. This is illustrated in the *tree* diagram of Figure 6.1, where the *leaves* of the tree represent the available sub-posterior densities, the directed edges are used to illustrate the computational flow of Monte Carlo Fusion, and the *root* vertex of the tree is the desired fusion density, f (as given in (1.1)).

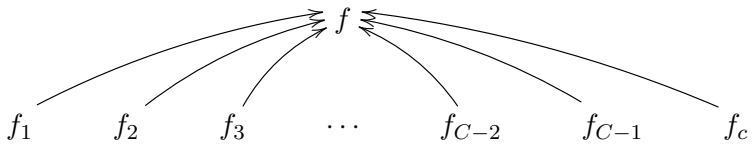
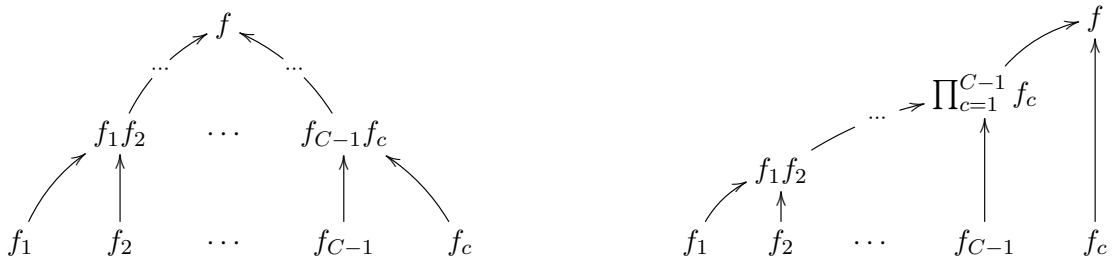


Figure 6.1: A tree representation of the fork-and-join approach for the fusion problem of (1.1).

As the goal of the methodology is to obtain a Monte Carlo approximation of f in (1.1), one could envision a recursive *divide-and-conquer* approach in which the sub-posteriors are combined in stages to recover f. There are a number of possible orderings in which we could combine sub-posteriors, and so we represent these orderings in *tree diagrams*, and term these *hierarchies* (see e.g. Figure 6.2). For instance as illustrated in Figure 6.2a, one approach would be to combine two sub-posteriors at a time (we term this a *balanced-binary tree* approach). In Figure 6.2a, the intermediate vertices represent intermediate (*auxiliary*) densities up to proportionality. The approximation of the distribution associated with any non-leaf vertex is obtained by an application of Fusion methodology to the densities of the children of that vertex. A balanced-binary tree approach is perhaps the most natural way to combine sub-posteriors in a *truly distributed setting* (where the simulation of each sub-posterior has been conducted separately, and so the inferences we wish to combine are distributed). Another approach one might employ is given in Figure 6.2b, whereby sub-posteriors are fused one at a time (and so we term this the *progressive tree* approach). This is perhaps the most natural approach for an *online setting*. We focus on these two natural hierarchies for the remainder of this chapter, although other hierarchies are certainly possible within our framework, and there is no limitation in unifying more than two vertices at any level of a tree (as suggested by both Section 6.1 and Figure 6.1).



(a) A balanced-binary tree.

(b) A progressive tree.

Figure 6.2: Illustrative hierarchies for the fusion problem of (1.1).

From this recursive perspective, sample approximations of auxiliary densities obtained at one level of any tree are themselves treated as sub-posteriors at the next level up. As such, one can iteratively apply the Fusion methodology of Section 6.1, working through the levels of the tree from the leaves to the root, using at each stage the output of one step as the input for the subsequent step. An advantage of our divide-and-conquer approach is that as fewer sub-posteriors are combined at each stage, we avoid (at each stage) the rapidly diminishing and variable importance weights. By utilising the importance sampling approach to Fusion we developed in Section 6.1.2, we can embed Fusion within sequential Monte Carlo to address the robustness of Fusion with increasing $|\mathcal{C}|$, albeit this being a trade-off with the cost of the repeated application of the methodology.

A divide-and-conquer variant of sequential Monte Carlo (D&C-SMC) was recently introduced in Lindsten et al. [2017] and is discussed in Section 3.4. D&C-SMC generalises the classical SMC framework from sequences (or chains) to trees, such as those in Figures 6.1 and 6.2. In our recursive setting, we unify distributed sample approximations by operating on a tree of *auxiliary* Fusion densities. Let $\mathbb{T} = (\mathcal{V}, \mathcal{E})$ denote a tree with vertices $\mathcal{V}$ and (directed) edge set $\mathcal{E}$. Let $\text{Leaf}(\mathbb{T})$ denote the leaves of the tree (which represent the sub-posteriors $f_1, \ldots, f_c$), $\text{Root}(\mathbb{T})$ denote the root of the tree (which represents f in (1.1)) and $\text{Ch}(v)$ denote the children of vertex $v \in \mathcal{V}$ where $\text{Ch}(t) = \emptyset$ if t is a leaf. Let $\mathcal{V} = \{v_0, v_1, \ldots, v_C, \ldots\}$ be the set of vertices, with $v_0 = \text{Root}(\mathbb{T})$, $\{v_1, \ldots, v_C\} = \text{Leaf}(\mathbb{T})$ and as many intermediate vertices as are required to specify the tree.

For the purposes of utilising the methodology developed in Section 6.1.2, we define the following notation for non-leaf vertices $v \notin \text{Leaf}(\mathbb{T})$: let $\mathcal{C}_v := \cup_{u \in \text{Ch}(v)} \mathcal{C}_u$ denote the index set representing the sub-posteriors that we want to unify for vertex $v \notin \text{Leaf}(\mathbb{T})$. To simplify the notation and avoid an unnecessary level of subscripts, we index densities and other quantities by v rather than $\mathcal{C}_v$ when it is clear what is intended. In particular, let $\boldsymbol{\Lambda}_v := \boldsymbol{\Lambda}_{\mathcal{C}_v}$, $\tilde{\boldsymbol{x}}^{(v)} := \tilde{\boldsymbol{x}}^{(\mathcal{C}_v)}$ as per (6.4), $\vec{\boldsymbol{x}}^{(v)} := \vec{\boldsymbol{x}}^{(\mathcal{C}_v)}$, $\boldsymbol{y}^{(v)} := \boldsymbol{y}^{(\mathcal{C}_v)}$ where $\boldsymbol{y}^{(v)} \sim f_v := f^{(\mathcal{C}_v)}$ for $v \notin \text{Leaf}(\mathbb{T})$. Let $\mathbb{W}_{\boldsymbol{\Lambda}_v}$ denote the law of a Brownian bridge $\{\boldsymbol{X}_t^{(v)}, t \in [0, T]\}$ with $\boldsymbol{X}_0^{(v)} := \boldsymbol{x}^{(v)}$ and $\boldsymbol{X}_T^{(v)} := \boldsymbol{y}^{(v)}$ with covariance $\boldsymbol{\Lambda}_v$. The extended target and proposal densities for vertex $v \notin \text{Leaf}(\mathbb{T})$ are denoted $g_v := g_{\mathcal{C}_v}$ and $h_v := h_{\mathcal{C}_v}$, respectively. Lastly, the importance sampling weights for $v \notin \text{Leaf}(\mathbb{T})$ are given by $\rho_0^{(v)}(\vec{\boldsymbol{x}}^{(v)}, \boldsymbol{y}^{(v)}) := \rho_0(\vec{\boldsymbol{x}}^{(\mathcal{C}_v)}, \boldsymbol{y}^{(\mathcal{C}_v)})$ and $\rho_1^{(v)}(\vec{\boldsymbol{x}}^{(v)}, \boldsymbol{y}^{(v)}) := \rho_1(\vec{\boldsymbol{x}}^{(\mathcal{C}_v)}, \boldsymbol{y}^{(\mathcal{C}_v)})$.

To describe our *Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF)* approach, we specify an algorithm that is carried out at each vertex $v \in \mathcal{V}$ which leads to a recursive procedure; an initial call to `d&c.gmcf`$(\text{Root}(\mathcal{V}), \ldots)$ carries out the overall approach. For $v \in \mathcal{V}$, we define a procedure (as given in Algorithm 6.2.1), which returns a weighted particle set $\{\vec{\boldsymbol{x}}_{0,i}^{(v)}, \boldsymbol{y}_i^{(v)}, w_i^{(v)}\}_{i=1}^N$ where $w_i^{(v)}$ denotes the normalised importance weight of particle i at vertex $v \in \mathcal{V}$. From this particle set, we can take the marginal weighted samples for $\boldsymbol{y}^{(v)}$ to approximate the fusion density $f_v \propto \prod_{u \in \text{Ch}(v)} f_u$ for vertex $v \in \mathcal{V}$. Recall that the leaf vertices, v_c for $c = 1, \ldots, C$, represent each of the sub-posteriors. It is straightforwardly possible to additionally incorporate importance sampling for the leaf vertices but for simplicity we assume that we have access to unweighted samples for the sub-posteriors. Therefore, at these leaf vertices, we simply sample from the sub-posteriors. If independent sampling is not feasible, one could use MCMC to obtain unweighted sample approximations at the leaves. Formal arguments (under appropriate regularity conditions) could in principle follow an approach analogous to that in Finke et al. [2020]. If v is a non-leaf vertex, we simply call Algorithm 6.1.2 by inputting the importance weighted samples $\{\boldsymbol{y}_i^{(u)}, w_i^{(u)}\}_{i=1}^N$ for $u \in \text{Ch}(v)$. As in standard SMC, although the auxiliary distributions are defined on larger spaces we do not need to retain sampled values which are not subsequently used; to obtain a more computationally manageable algorithm, we can choose to retain only the final parameter space marginal at each vertex (i.e. only returning $\{\boldsymbol{y}_i^{(v)}, w_i^{(v)}\}_{i=1}^N$) since the computation of the importance weights $\rho_0^{(v)}$ and $\tilde{\rho}_1^{(v)}$ at each vertex $v \notin \text{Leaf}(\mathbb{T})$ only requires $\{\boldsymbol{y}^{(u)}\}_{u \in \text{Ch}(v)}$.

Algorithm 6.2.1 `d&c.gmcf`(v, N, T): Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF) [Chan et al., 2021, Algorithm 2].

Given: Sub-posteriors, $\{f_u\}_{u \in \text{Leaf}(\mathbb{T})}$, and preconditioning matrices $\{\boldsymbol{\Lambda}_u\}_{u \in \mathbb{T}}$.
Input: Node in tree, v, the number of particles N, and time horizon $T > 0$.

1. For $u \in \text{Ch}(v)$,
   (a) $\left\{\boldsymbol{x}_i^{(u)}, \boldsymbol{y}_i^{(u)}, w_i^{(u)}\right\}_{i=1}^N \leftarrow$ `d&c.gmcf`(u, N, T).
2. If $v \in \text{Leaf}(\mathbb{T})$,
   (a) For $i = 1, \ldots, N$, sample $\boldsymbol{y}_i^{(v)} \sim f_v(\boldsymbol{y})$.
   (b) **Output:** $\{\emptyset, \boldsymbol{y}_i^{(v)}, \frac{1}{N}\}_{i=1}^N$.
3. If $v \notin \text{Leaf}(\mathbb{T})$,
   (a) **Output:** Call `gmcf`$(\text{Ch}(v), \{\{\boldsymbol{y}_i^{(u)}, w_i^{(u)}\}_{i=1}^N, \boldsymbol{\Lambda}_u\}_{u \in \text{Ch}(v)}, N, T)$.

Note when calling Algorithm 6.1.2 (in Algorithm 6.2.1 Step 3), we input N importance samples from each sub-posterior (i.e. $M_u = M$ for $u \in \text{Ch}(v)$). As noted in Section 6.1.2, for simplicity, we use these to obtain partial proposals by pairing the sub-posterior realisations index-wise. If we use this simple merging scheme then $M = N$, and so resampling in Algorithm 6.1.2 Step 3a might not be necessary. As standard within the SMC literature, in practice we choose to re-sample only when we observe *weight degeneracy* [Kong et al., 1994]. We monitor weight degeneracy by computing the *effective sample size (ESS)* (3.18) of the particle set, and if it falls below some user-specified threshold then resampling is performed (see Sections 3.2–3.3 for a short introduction on resampling). There are two steps where we can compute the ESS (3.18) (and if necessary

resample) in Algorithm 6.1.2: (i) after computing the weights for the partially composed proposals, $\vec{w}_j^{(\mathcal{C})} := \left(\prod_{c\in\mathcal{C}} w_j^{(c)}\right) \cdot \rho_0(\vec{\boldsymbol{x}}_{0,j}^{(\mathcal{C})})$ for $j = 1, \ldots, M$ (in Step 2); and (ii) after computing the weights $\tilde{\rho}_{1,i}^{(\mathcal{C})} := \tilde{\rho}_1^{(b)}(\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})}, \boldsymbol{y}_i^{(\mathcal{C})})$ (in Step 3c).

6.3 Illustrative comparisons with Monte Carlo Fusion

As discussed in the introduction of this chapter, although *Monte Carlo Fusion (MCF)* of Dai et al. [2019] (see Section 5.1) offers some compelling advantages (such as providing a direct Monte Carlo approximation to (1.1) without approximating the individual sub-posteriors), the existing methodology lack robustness in various key practical settings. In this section, we revisit these settings, and with the aid of illustrative examples, show that the *Generalised Monte Carlo Fusion (GMCF)* and *Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF)* approaches we introduced in Sections 6.1 and 6.2 respectively, address these key bottlenecks when contrasted with MCF. In particular, in Section 6.3.1 we consider the effect of increasing sub-posterior correlation, in Section 6.3.2 we consider the robustness with increasing numbers of sub-posteriors, and in Section 6.3.3 we consider how to address conflicting sub-posteriors. When applying GMCF (or calling it as an embedded algorithm in D&C-GMCF), in all cases we use the GPE-2 variant of Algorithm 6.1.2 discussed in Definition 6.1.2 of Section 6.1, and follow the direction of Fearnhead et al. [2008] by estimating the mean of the Negative Binomial distribution in (6.21) by using the Trapezoidal rule and setting $\beta_c = 10$. We note that different choices for the parameters of the Negative Binomial distribution in Definition 6.1.2 may affect the efficiency and variance of our algorithm, but does not introduce any bias to the estimator. Details on how to find the corresponding code is given in Appendix A, and necessary calculations to implement these examples are given in Appendix B (in particular Appendix B.3 and Appendix B.4).

In order to compare the MCF, GMCF and D&C-GMCF approaches, we compute both the computational run-times of each methodology and a metric which we term the *Integrated Absolute Distance (IAD)*. To compute the IAD we average across each dimension the difference between the true target (*fusion*) density (f), and a kernel density estimate of the draws realised using a given methodology ($\hat{f}$). In particular,

$$\text{IAD} = \frac{1}{2d}\sum_{j=1}^{d}\int \left|\hat{f}(\theta_j) - f(\theta_j)\right| \mathrm{d}\theta_j \in [0,1]. \tag{6.25}$$

In the case where the true marginal density is not available analytically, we take as a proxy for the target f a kernel density estimate of f (for instance, obtained using realisations from an MCMC run).

6.3.1 Effect of correlation

In this example, we consider the illustrative case in which we wish to recover a bivariate Gaussian target distribution, $f \propto f_1 f_2$, where $f_c \sim \mathcal{N}_2(\mathbf{0}, \mathbf{\Sigma})$ with $\mathbf{\Sigma} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. As we are only considering combining two sub-posteriors in this section, we in effect consider only the GMCF approach of Section 6.1. To study the impact of sub-posterior correlation on the robustness of MCF and GMCF we can simply consider varying the single parameter ρ, and (in this case) compute the effective sample size (ESS) per second averaged across 50 runs in order to compare the efficiency of each methodology. For simplicity, we assume we are able to sample directly from each sub-posterior, and for both methodologies we set $T = 1$. For the purposes of the GMCF approach of Algorithm 6.1.2, we simply set $\mathbf{\Lambda}_c = \hat{\mathbf{\Sigma}}_c$, where $\hat{\mathbf{\Sigma}}_c$ is the *estimated* covariance matrix from the sub-posterior samples for $c = 1, 2$ (and so in effect we have incorporated global information into our proposals), and use a particle set size of $N = 10000$. The results are presented in Figure 6.3, which clearly show that GMCF is robust to increasing sub-posterior correlation, and offers a significant computational advantage over MCF (which in this case exhibits a strong degradation in efficiency and performance as we increase the correlation between the two components in this two-dimensional example).

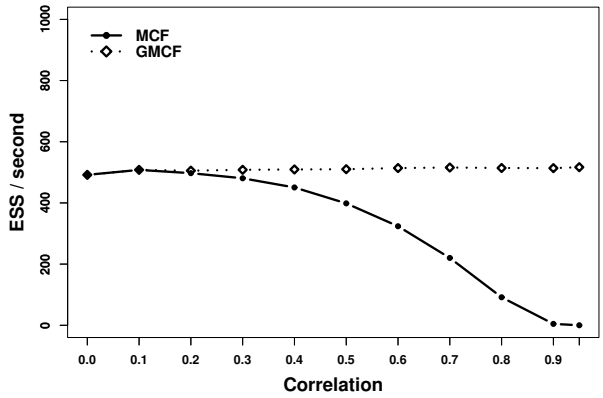
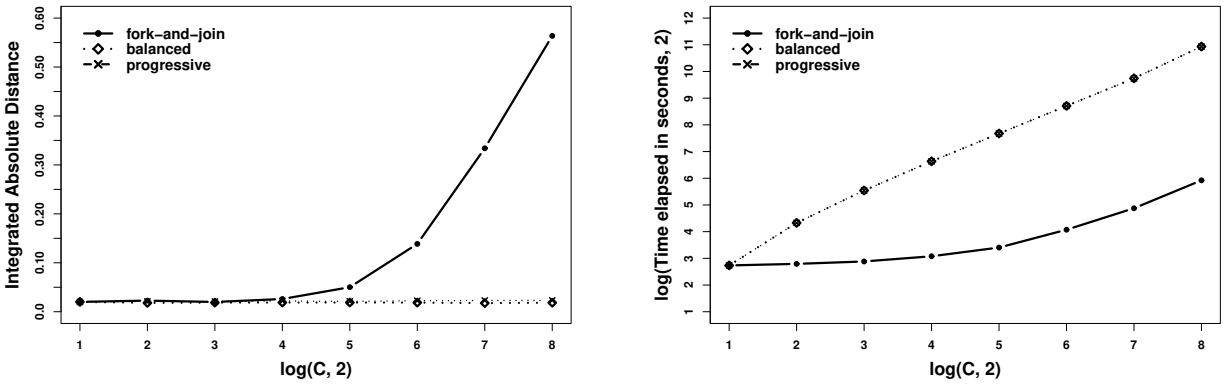


Figure 6.3: ESS per second (averaged over 50 runs) when contrasting Monte Carlo Fusion and Generalised Monte Carlo Fusion, along with increasing sub-posterior correlation, as per the example in Section 6.3.1.

6.3.2 Effect of hierarchy

We consider the illustrative case of attempting to recover a univariate standard Gaussian target distribution. In particular, we have $f \propto \prod_{c=1}^{C} f_c$, where $f_c \sim \mathcal{N}_1(0, C)$ for $c = 1, \ldots, C$. By simply varying C, we can study the robustness with increasing numbers of sub-posteriors of MCF (in effect the fork-and-join approach illustrated in Figure 6.1), and both our suggested versions of D&C-GMCF (the balanced-binary tree approach illustrated in Figure 6.2a, and the progressive tree approach illustrated in Figure 6.2b). Note that in our chosen idealised setting, there is no

advantage conferred with our embedded GMCF methodology of Section 6.1, and so we are simply contrasting hierarchies. In all cases we use a particle set of size $N = 10000$ with resampling if ESS $< N/2$, set $T = 1$, use an appropriately scaled identity as the preconditioning (scalar) matrix, and average across 50 runs. The results are presented in Figure 6.4, which clearly show that, in contrast to the fork-and-join tree approach, both the balanced-binary tree and progressive tree approaches are robust in recovering the correct posterior distribution in the case of increasing C at the cost of modestly increased computational cost.



(a) Integrated Absolute Distance.

(b) Computational cost.

Figure 6.4: Illustrative comparison of the effect of using different hierarchies in Section 6.3.2 (averaged over 50 runs).

6.3.3 Dealing with conflicting sub-posteriors

Directly unifying C *conflicting* sub-posteriors (sub-posteriors which have little common support and have high total-variation distance) using a fork-and-join approach as in Monte Carlo Fusion [Dai et al., 2019] and Figure 6.1 is impractical. This can be understood with reference to (6.7) and (6.8), which indicates that importance weights will degrade rapidly in this setting.

An approach to deal with conflicting sub-posteriors is to temper the sub-posteriors (to an inverse temperature $\beta \in (0, 1]$ such that there is sufficient sub-posterior overlap), and then propose a suitable tree for which the recursive Divide-and-Conquer Generalised Monte Carlo Fusion approach we introduced in Section 6.2 could then be applied to recover (1.1). In particular,

$$f(\boldsymbol{x}) \propto \prod_{i=1}^{1/\beta} \left[\prod_{c=1}^{C} f_c^{\beta}(\boldsymbol{x}) \right], \qquad \text{for } \frac{1}{\beta} \in \mathbb{N}. \tag{6.26}$$

One such generic tree is provided in Figure 6.5, in which the tempered sub-posteriors are first unified into $1/\beta \in \mathbb{N}$ tempered posteriors, which are then again unified into f.

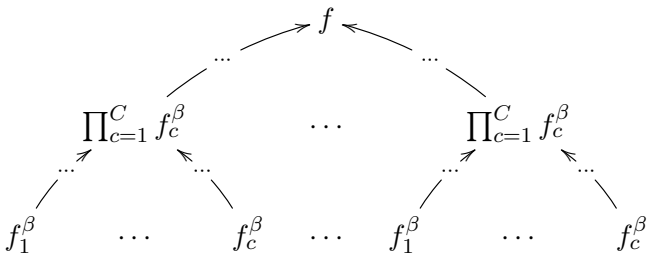
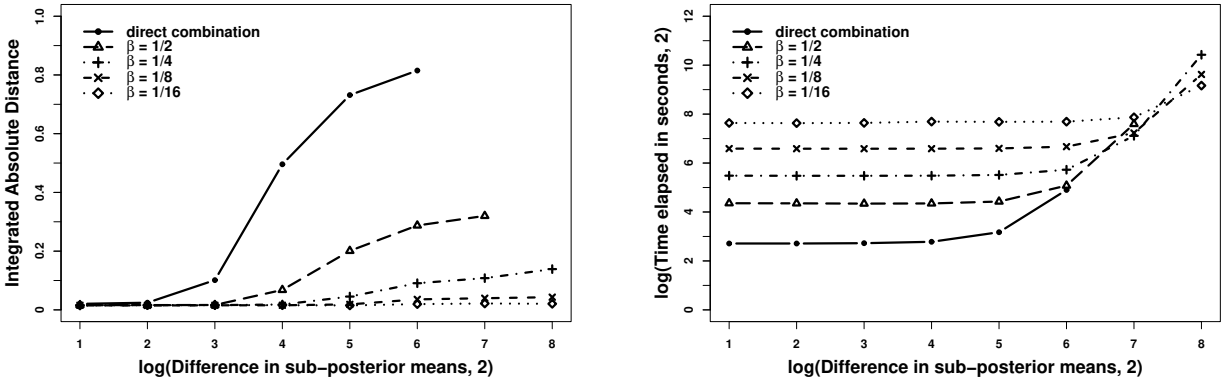


Figure 6.5: Illustrative tree approach for the fusion problem in the case of conflicting sub-posteriors as in Section 6.3.3. $1/\beta$ copies of the C tempered (and over-lapping) sub-posteriors represent the leaves of the tree, which are unified into $1/\beta$ tempered versions of f (using a suitable tree and D&C-GMCF as in Section 6.2), and then unified again (using another tree, and D&C-GMCF) to recover f.

To illustrate the advantage of our D&C-GMCF and tempering approach in the case of conflicting sub-posteriors, we consider the scenario of unifying two Gaussian sub-posteriors with the same variance (1), but with different mean ($\pm\mu$). In particular, we have $f \propto f_1 f_2$ where $f_1 \sim \mathcal{N}_1(-\mu, 1)$ and $f_2 \sim \mathcal{N}_1(\mu, 1)$. By simply increasing μ we can emulate increasingly conflicting sub-posteriors and study how MCF (which is equivalent to the fork-and-join approach of Figure 6.1), behaves in terms of the IAD metric and computational time. We contrast this with our tempering approach, considering a range of temperatures $1/\beta \in \{2, 4, 8, 16\}$, and then following the guidance of Figure 6.5. In particular, we use our D&C-GMCF approach to unify the tempered sub-posteriors with the balanced-binary approach of Figure 6.2a for both the first and second stage in Figure 6.5. In all cases, we use a particle set size of $N = 10000$ with resampling if ESS $< N/2$, set $T = 1$, and average across 50 runs. The results are presented in Figure 6.6, and show clearly that our D&C-GMCF approach is significantly more robust to conflicting sub-posteriors than the MCF approach where no tempering is applied. A natural trade-off arises when applying the tempering approach suggested, in that decreasing β results in tempered sub-posteriors which are less conflicting and are easier to combine, but there is an increased computational cost in recovering f as an increased number of levels are added to the resulting tree.

6.4 Examples

In this section, we consider a logistic regression model applied to a simulated dataset (Section 6.4.1), and a real credit card dataset (Section 6.4.2). For each dataset, we compare the performance (in terms of computational run-time and the *Integrated Absolute Distance (IAD)* as defined in (6.25)) of the Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF) method introduced in Section 6.2, against the other established approximate methodologies we discussed in Section 1.2.1. As in Section 6.3, we use the GPE-2 variant of Algorithm 6.1.2 discussed in Definition 6.1.2 with the same guidance on parameterisation. We consider the setting where the data is first split into C disjoint subsets and the simulation of each sub-posterior is conducted separately. For this



(a) Integrated Absolute Distance.

(b) Computational cost.

Figure 6.6: Illustrative comparison of using no tempering (solid line), and tempering at 4 different levels together with D&C-GMCF, to combat conflicting sub-posteriors as per Section 6.3.3 (averaged over 50 runs).

reason, we focus on the balanced-binary tree approach (Figure 6.2a), as for a fixed dataset this is the most natural hierarchy (whereas the *progressive tree* approach in Figure 6.2b is more naturally suited to an *online* setting). The established methodologies we contrast our implementation against are *Consensus Monte Carlo (CMC)* [Scott et al., 2016], the kernel density averaging approach of Neiswanger et al. [2014] (which we term *KDEMC*), and the *Weierstrass Sampler (WRS)* [Wang and Dunson, 2013] (see Appendix A for details of where to find the corresponding code). For each example, as a benchmark for f (in terms of a reference in computing IAD), we use `Stan` [Carpenter et al., 2017] on the entire dataset, together with an appropriate choice of prior, to find a reference sample approximation of the desired f. For full details on where to find the corresponding code/scripts to implement these examples, see Appendix A. Furthermore, in Appendix B, we supply details of calculations required to implement these examples.

6.4.1 Simulated data example

In this section, we consider an idealised *small data size* scenario ($m = 1000$) to which we applied a logistic regression model:

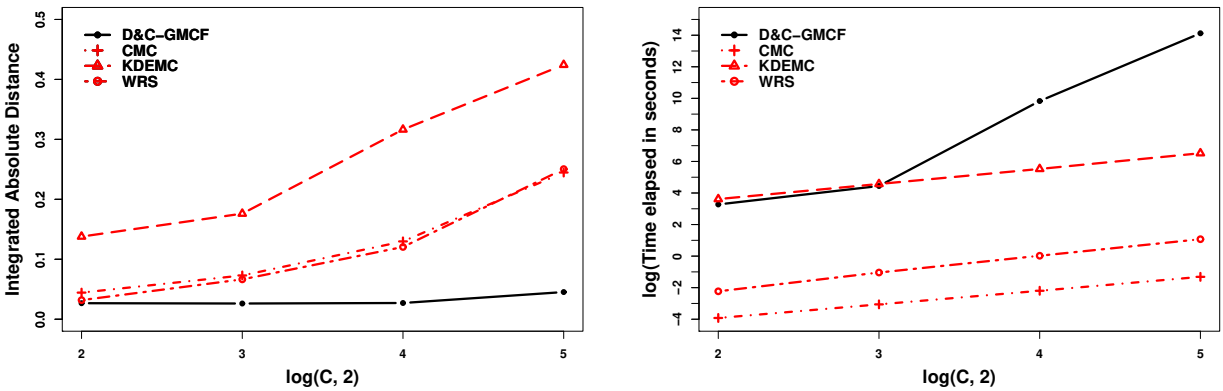
$$y_i = \begin{cases} 1 & \text{with probability } \frac{\exp\{\mathbf{x}_i^\intercal \boldsymbol{\beta}\}}{1+\exp\{\mathbf{x}_i^\intercal \boldsymbol{\beta}\}}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.27}$$

This is a variant of Scott et al. [2016, Section 4.3], and is of specific interest because when such a dataset is split among a large number of cores, C, both *exact* and *approximate* methodologies for unifying sub-posterior samples are challenged. In particular, the resulting sub-posteriors will typically conflict with one another and have little overlapping support.

Each record of the simulated design matrix contained four covariates in addition to an intercept. The ith entry of the design matrix is given by $\mathbf{x}_i = [1, \zeta_{i,1}, \zeta_{i,2}, \zeta_{i,3}, \zeta_{i,4}]^\intercal$, where $\zeta_{i,1}, \zeta_{i,2}, \zeta_{i,3}, \zeta_{i,4}$ are random variables generated from a mixture density with a point-mass at zero (and so are either *activated* or not). In particular, we have for $j = 1, \ldots, 4$, that $\zeta_{i,j} \sim p_j \mathcal{N}_1(1,1) + (1-p_j)\delta_0$. For this example we chose $p_1 = 0.2$, $p_2 = 0.3$, $p_3 = 0.5$ and $p_4 = 0.01$ (corresponding to a rarely activated covariate). Upon simulating the design matrix, binary observations were obtained by simulation using the parameters $\beta = [-3, 1.2, -0.5, 0.8, 3]^\intercal$. In total there were a relatively small number of positive responses ($\sum_i y_i = 129$).

To study our methodology, we begin by splitting the dataset of size $m = 1000$ equally between $C \in \{4, 8, 16, 32\}$ cores. On each dataset on each core, we fit the logistic regression model using MCMC with `Stan` using a Gaussian prior distribution with mean 0 and variance C on each of the parameters, resulting in the required C sub-posteriors. The resulting sub-posteriors in the case of $C = 32$ naturally resulted in conflicting sub-posteriors (see Section 6.3.3) due to the small amount of data on each core. We then applied our D&C-GMCF approach using a balanced-binary tree with $N = 10000$ and $T = 0.5$, and compared our approach to CMC, KDEMC and WRS. The results are shown in Figure 6.7, with reference to the `Stan` benchmark.

It is clear from Figure 6.7a that D&C-GMCF achieves the best sample approximation in terms of IAD of any of the approaches considered. Furthermore, the quality of the sample approximation is robust to the increasing sub-posteriors that we consider. In terms of computational cost, CMC outperforms all other methodologies, but has poor performance in terms of IAD. WRS offers a similar performance to CMC in this example but at a slightly higher computational cost whereas KDEMC has by far the poorest performance.



(a) Integrated Absolute Distance.

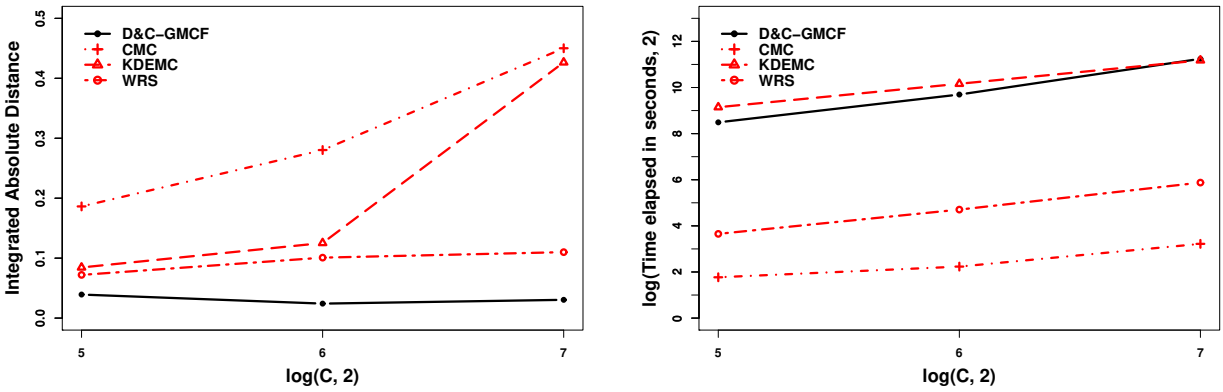
(b) Computational cost.

Figure 6.7: Comparison of competing methodologies to Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF) applied to a logistic regression problem with simulated data (in the setting of Section 6.4.1).

6.4.2 Credit-card data example

In this example, we consider the *'Default of credit card clients'* dataset available from the *UCI Machine Learning Repository* [Yeh and Lien, 2009] and again fit the logistic regression model of (6.27). The dataset comprised $m = 30000$ records, each of which contained response variable of whether a default had occurred (which we treated as $y_i = 1$), or not (in which case $y_i = 0$). From the dataset we used the **X2: Gender** attribute, treating it as a binary covariate with 1 being male and 0 female. In addition we used the **X3: Education** attribute to create three further binary covariates: a binary corresponding to whether the individual had completed high school; a binary corresponding to completion of university; and a further binary corresponding to completion of graduate school. These three education covariates were chosen to induce strong correlation in the resulting inference.

We again split the dataset of size $m = 30000$ equally between C cores, and again used `Stan` together with a Gaussian prior distributions with mean 0 and variance C on each of the parameters, to arrive at our C sub-posteriors. In this example, we consider $C \in \{32, 64, 128\}$ subsets, which we then attempt to unify. This example is particularly challenging as the data and sub-posteriors exhibit large correlation (due to the covariates related to **X3: Education**), especially for large C. D&C-GMCF using a balanced-binary tree with $N = 30000$ and $T = 0.5$, together with CMC, KDEMC and WRS were then used to unify the C sub-posteriors. The results are shown in Figure 6.8 with reference to the `Stan` benchmark. The results in Figure 6.8 are comparable to those of Section 6.4.1: D&C-GMCF achieves the best IAD of all methodologies, and is robust to increasing C. This comes at a moderate fixed computational cost, which scales no worse than any other methodology (including CMC), and indeed in this example has a computational cost which is lower than KDEMC.



(a) Integrated Absolute Distance.

(b) Computational cost.

Figure 6.8: Comparison of competing methodologies to Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF) applied to a logistic regression problem with real data (in the setting of Section 6.4.2).

Chapter 7

Divide-and-Conquer Generalised Bayesian Fusion

We have seen that although the Monte Carlo Fusion (MCF) approach [Dai et al., 2019] (see Section 5.1) provides a theoretical framework for sampling independent draws from the fusion target density f in (1.1), it has several computational drawbacks (which we discussed in detail in Section 6.3). To alleviate some of these problems, we introduced the Generalised Monte Carlo Fusion (GMCF) approach in Chapter 6 which reformulates the theory underpinning the MCF approach. In Section 6.2, we embedded the GMCF approach within a Divide-and-Conquer Sequential Monte Carlo (D&C-SMC) [Lindsten et al., 2017] (see Section 3.4) to arrive at the Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF) approach, whereby sub-posterior sample approximations were combined in stages to recover the target fusion density f in (1.1). We saw through various simulation studies and examples that D&C-GMCF offered a much greater robustness to a number of practical scenarios. However, as discussed in Section 5.2, the Bayesian Fusion (BF) approach of Dai et al. [2021] is an alternative sequential Monte Carlo (SMC) (see Chapter 3) approach which also aimed to develop a methodology which shares the consistency properties of MCF while addressing some of the scalability issues of MCF. While the BF and D&C-GMCF approaches allow Fusion to be applied to more practical settings, both approaches still lack the scalability with regards to dimensionality of the underlying fusion target density. In this chapter, we will generalise the theory and methodology of BF by applying many of the ideas outlined in Chapter 6; namely adjusting the proposal distribution of the extended target density to incorporate global information of the sub-posteriors. We subsequently embed this *Generalised Bayesian Fusion (GBF)* approach within a D&C-SMC paradigm. We will see that the resulting *Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF)* approach is the most scalable Fusion methodology developed so far and is applicable to much higher dimensional problems.

In the next section and with Algorithm 7.1.2, we present the theory and methodology of our GBF approach. In Section 7.2, we follow the same approach as in Section 6.2 to embed the algorithm with a divide-and-conquer approach to combine the sub-posterior is stages and present our D&C-GBF approach in Algorithm 7.2.1. This is accompanied with practical guidance for implementing GBF which includes full details for selecting the user-specified parameters in the algorithm and is provided in Section 7.3. In Section 7.4, we present illustrative simulation studies to investigate the robustness of our approach when applied to various scenarios; we investigate the applicability of the implementational guidance developed in Sections 7.4.1–7.4.2 and study the impact of dimension on the robustness of the Fusion approaches in Section 7.4.3. In Section 7.5, we study the performance of our D&C-GBF approach in a number of real data applications. The content of this chapter is joint work with my supervisors, Dr. Murray Pollock and Professor Gareth Roberts.

7.1 A generalisation of Bayesian Fusion

In this section, we develop theory and methodology to generalise and improve upon Bayesian Fusion (BF) Dai et al. [2021], by incorporating information about the covariance of the sub-posteriors within the SDE formulation of the algorithm. As in Section 6.1, we consider the more general fusion density $f^{(\mathcal{C})} \propto \prod_{c\in\mathcal{C}} f_c$, where $\mathcal{C}$ is an index set representing the sub-posteriors to unify.

7.1.1 Theory

To begin generalising the BF approach, we first need to derive tractable dynamics of our *proposal measure*, denoted $\mathbb{P}$, of $|\mathcal{C}|$ *scaled* Brownian motion processes which are conditioned to coalesce at some time $T > 0$. We correct this to find the target *fusion measure*, denoted $\mathbb{F}$, by finding appropriate importance weights. In particular, let $\mathbb{P}$ be the *proposal measure* given by the probability law induced by $|\mathcal{C}|$ interacting d-dimensional parallel continuous-time Markov processes in $[0,T]$ where each process is given by the stochastic differential equation,

$$\mathrm{d}\boldsymbol{X}_t^{(c)} = \frac{\tilde{\boldsymbol{X}}_t - \boldsymbol{X}_t^{(c)}}{T-t}\mathrm{d}t + \boldsymbol{\Lambda}_c^{\frac{1}{2}}\mathrm{d}\boldsymbol{W}_t^{(c)}, \quad \boldsymbol{X}_0^{(c)} := \boldsymbol{x}_0^{(c)} \sim f_c, \quad t \in [0,T], \tag{7.1}$$

where $\boldsymbol{\Lambda}_c$ are (positive semi-definite) user-specified matrices associated to sub-posterior f_c for $c \in \mathcal{C}$ with $\boldsymbol{\Lambda}_c^{1/2}$ being the (positive semi-definite) square root of $\boldsymbol{\Lambda}_c$ where $\boldsymbol{\Lambda}_c^{1/2}\boldsymbol{\Lambda}_c^{1/2} = \boldsymbol{\Lambda}_c$. Note that for the purposes of our numerical simulations later we use the Schur decomposition. Furthermore, $\{\boldsymbol{W}_t^{(c)}\}_{c\in\mathcal{C}}$ denotes independent Brownian motions, and $\tilde{\boldsymbol{X}}_t^{(c)}$ denoting the weighted average of the processes at time t with weights $\{\boldsymbol{\Lambda}_c\}_{c\in\mathcal{C}}$. We denote by $\vec{\boldsymbol{x}}_t^{(\mathcal{C})} \in \mathbb{R}^{|\mathcal{C}|\times d}$ a vector composed of $\{\boldsymbol{x}_t^{(c)}\}_{c\in\mathcal{C}} \in \mathbb{R}^d$ (in particular, we have $\vec{\boldsymbol{x}}_t^{(\mathcal{C})} := (\boldsymbol{x}_t^{(c_1)}, \ldots, \boldsymbol{x}_t^{(c_{|\mathcal{C}|})})$, with c_i denoting the ith element of the index set $\mathcal{C}$). Realisations of the proposal measure are denoted as $\mathfrak{X} := \{\vec{\boldsymbol{x}}_t^{(\mathcal{C})}, t \in [0,T]\}$. We note that the initialisation of the proposal measure given by (7.1) at time $t = 0$ only requires independent draws from the $|\mathcal{C}|$ sub-posteriors that we wish to unify. For the purposes of exposition, we defer discussion on the practical simulation of $\mathbb{P}$ (without discretisation error) to Section 7.1.2.1.

The *Fusion measure* $\mathbb{F}$ is the measure induced by the Radon-Nikodým derivative,

$$\frac{\mathrm{d}\mathbb{F}}{\mathrm{d}\mathbb{P}}(\mathfrak{X}) \propto \rho_0\left(\vec{\boldsymbol{x}}_0^{(\mathcal{C})}\right) \cdot \prod_{c\in\mathcal{C}}\left[\exp\left\{-\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right)\mathrm{d}t\right\}\right], \tag{7.2}$$

where $\{\boldsymbol{X}_t^{(c)}, t\in[0,T]\}$ is a Brownian bridge from $\boldsymbol{X}_0^{(c)} := \boldsymbol{x}_0^{(c)} \sim f_c$ to $\boldsymbol{X}_T^{(c)} := \boldsymbol{x}_T^{(c)}$ with covariance matrix $\boldsymbol{\Lambda}_c$, ρ_0 is given in (6.7) and ϕ_c is given in (6.9). Recall with the Fusion methodologies, we typically we assume that we can evaluate each sub-posterior pointwise (up to its normalising constant) and for $c = 1,\ldots,C$, f_c is nowhere zero and everywhere differentiable, and that we can compute $A_c(\boldsymbol{x}) := \log f_c(\boldsymbol{x})$, $\nabla A_c(\boldsymbol{x})$, and $\nabla^2 A_c(\boldsymbol{x})$ pointwise (where ∇ is the gradient operator and ∇^2 is the Hessian), since we will need to be able to evaluate $\phi_c(\boldsymbol{x})$ defined in (6.9). However, we will see later (in Section 7.3.3.2) that this is not a limiting factor of the methodology.

Having defined the proposal measure $\mathbb{P}$ and the Fusion measure $\mathbb{F}$, we can now establish how we can access the target fusion density $f^{(\mathcal{C})}$ by means of the temporal marginal of $\mathbb{F}$ corresponding to value of the $|\mathcal{C}|$ trajectories at the time of coalescence T.

Theorem 7.1.1. *Under the fusion measure $\mathbb{F}$, the ending points of the $|\mathcal{C}|$ interacting, parallel processes have a common value at time T, $\boldsymbol{y}^{(\mathcal{C})}$ which has density $f^{(\mathcal{C})}$ and $\boldsymbol{y}^{(\mathcal{C})} = \boldsymbol{x}_T^{(c_1)} = \cdots = \boldsymbol{x}_T^{(c_{|\mathcal{C}|})}$ almost surely.*

Proof. Following the approach of Dai et al. [2019, Appendix A], we begin by proving that the law of $|\mathcal{C}|$ independent Brownian motions initialised at $\boldsymbol{x}_0^{(c)} \sim f_c$ for $c\in\mathcal{C}$ and conditioned to coalesce at time T satisfies (7.1). Here, we use Doob h-transforms [Rogers and Williams, 2000, Chapter IV, Section 6.39] and define the following space-time harmonic function

$$h\left(t, \vec{\boldsymbol{x}}_t^{(\mathcal{C})}\right) = \int \prod_{c\in\mathcal{C}} \frac{1}{\sqrt{2\pi(T-t)|\boldsymbol{\Lambda}_c|}} \exp\left\{-\frac{(\boldsymbol{y}-\boldsymbol{x}_t^{(c)})^\intercal \boldsymbol{\Lambda}_c^{-1}(\boldsymbol{y}-\boldsymbol{x}_t^{(c)})}{2(T-t)}\right\} \mathrm{d}\boldsymbol{y}, \tag{7.3}$$

which represents the integrated density of coalescence at time T given the current state $\vec{\boldsymbol{x}}_t^{(\mathcal{C})}$. Then the $|\mathcal{C}|$ conditioned processes satisfy a SDE of the form,

$$\mathrm{d}\vec{\boldsymbol{X}}_t^{(\mathcal{C})} = \vec{\boldsymbol{\Lambda}}^{\frac{1}{2}}\ \mathrm{d}\vec{\boldsymbol{W}}_t^{(\mathcal{C})} + \vec{\boldsymbol{\Lambda}}\nabla \log\left(h(t, \vec{\boldsymbol{X}}_t^{(\mathcal{C})})\right)\ \mathrm{d}t, \tag{7.4}$$

where $\nabla \log\left(h(t, \vec{\boldsymbol{x}}_t^{(\mathcal{C})})\right) =: \left(\boldsymbol{v}_t^{(c_1)}, \ldots, \boldsymbol{v}_t^{(c_{|\mathcal{C}|})}\right)$ is a collection of $|\mathcal{C}|d$-dimensional vectors and

$$\vec{\boldsymbol{\Lambda}}^{\frac{1}{2}} = \begin{pmatrix} \boldsymbol{\Lambda}_{c_1}^{\frac{1}{2}} & \mathbf{0}_{d\times d} & \ldots & \mathbf{0}_{d\times d} \\ \mathbf{0}_{d\times d} & \boldsymbol{\Lambda}_{c_2}^{\frac{1}{2}} & \ldots & \mathbf{0}_{d\times d} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \ldots & \boldsymbol{\Lambda}_{c_{|\mathcal{C}|}}^{\frac{1}{2}} \end{pmatrix}, \qquad \vec{\boldsymbol{\Lambda}} = \begin{pmatrix} \boldsymbol{\Lambda}_{c_1} & \mathbf{0}_{d\times d} & \ldots & \mathbf{0}_{d\times d} \\ \mathbf{0}_{d\times d} & \boldsymbol{\Lambda}_{c_2} & \ldots & \mathbf{0}_{d\times d} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \ldots & \boldsymbol{\Lambda}_{c_{|\mathcal{C}|}} \end{pmatrix},$$

where $\mathbf{\Lambda}_c^{\frac{1}{2}}$ is the (positive semi-definite) square root of $\mathbf{\Lambda}_c$ where $\mathbf{\Lambda}_c^{\frac{1}{2}}\mathbf{\Lambda}_c^{\frac{1}{2}} = \mathbf{\Lambda}_c$ for $c \in \mathcal{C}$, and $\mathbf{0}_{d\times d}$ denotes the $d \times d$ matrix with all elements equal to 0.

Considering the cth term and letting $\mathbf{\Lambda}_{\mathcal{C}}^{-1} = \sum_{c\in\mathcal{C}} \mathbf{\Lambda}_c^{-1}$, then

$$\begin{aligned}
\boldsymbol{v}_t^{(c)} &= \frac{\int \left(\frac{\mathbf{\Lambda}_c^{-1}(\boldsymbol{y}-\boldsymbol{x}_t^{(c)})}{T-t}\right) \prod_{c\in\mathcal{C}} \frac{1}{\sqrt{2\pi(T-t)|\mathbf{\Lambda}_c|}} \exp\left\{-\frac{\left(\boldsymbol{y}-\boldsymbol{x}_t^{(c)}\right)^\intercal \mathbf{\Lambda}_c^{-1}\left(\boldsymbol{y}-\boldsymbol{x}_t^{(c)}\right)}{2(T-t)}\right\} \mathrm{d}\boldsymbol{y}}{\int \prod_{c\in\mathcal{C}} \frac{1}{\sqrt{2\pi(T-t)|\mathbf{\Lambda}_c|}} \exp\left\{-\frac{\left(\boldsymbol{y}-\boldsymbol{x}_t^{(c)}\right)^\intercal \mathbf{\Lambda}_c^{-1}\left(\boldsymbol{y}-\boldsymbol{x}_t^{(c)}\right)}{2(T-t)}\right\} \mathrm{d}\boldsymbol{y}} \\
&= \frac{\int \left(\frac{\mathbf{\Lambda}_c^{-1}\boldsymbol{y}}{T-t}\right) \exp\left\{-\frac{(\boldsymbol{y}-\tilde{\boldsymbol{x}}_t)^\intercal \mathbf{\Lambda}_{\mathcal{C}}^{-1}(\boldsymbol{y}-\tilde{\boldsymbol{x}}_t)}{2(T-t)}\right\} \mathrm{d}\boldsymbol{y}}{\int \exp\left\{-\frac{(\boldsymbol{y}-\tilde{\boldsymbol{x}}_t)^\intercal \mathbf{\Lambda}_{\mathcal{C}}^{-1}(\boldsymbol{y}-\tilde{\boldsymbol{x}}_t)}{2(T-t)}\right\} \mathrm{d}\boldsymbol{y}} - \frac{\mathbf{\Lambda}_c^{-1}\boldsymbol{x}_t^{(c)}}{T-t} \\
&= \frac{\mathbf{\Lambda}_c^{-1}\left(\tilde{\boldsymbol{x}}_t - \boldsymbol{x}_t^{(c)}\right)}{T-t}.
\end{aligned}$$

Consequently, we have

$$\nabla \log\left(h(t, \vec{\boldsymbol{x}}_t^{(\mathcal{C})})\right) = \left(\frac{\mathbf{\Lambda}_{c_1}^{-1}\left(\tilde{\boldsymbol{x}}_t - \boldsymbol{x}_t^{(c_1)}\right)}{T-t}, \ldots, \frac{\mathbf{\Lambda}_{c_{|\mathcal{C}|}}^{-1}\left(\tilde{\boldsymbol{x}}_t - \boldsymbol{x}_t^{(c_{|\mathcal{C}|})}\right)}{T-t}\right), \tag{7.5}$$

and (7.1) holds.

Next, we show that under $\mathbb{F}$ this common value has density f. Since $\mathbb{P}$ is the measure for $|\mathcal{C}|$ coalesced Brownian motions (shown above), from (7.2), we can write $\mathbb{F}$ as

$$\begin{aligned}
\mathrm{d}\mathbb{F}(\mathfrak{X}) &\propto \mathrm{d}\mathbb{P}(\mathfrak{X}) \cdot \rho_0\left(\vec{\boldsymbol{x}}_0^{(\mathcal{C})}\right) \cdot \prod_{c\in\mathcal{C}} \left[\exp\left\{-\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right) \mathrm{d}t\right\}\right] \\
&\propto \left[\prod_{c\in\mathcal{C}} f_c\left(\boldsymbol{x}_0^{(c)}\right)\right] \cdot \exp\left\{-\frac{(\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}_0^{(\mathcal{C})})^\intercal \mathbf{\Lambda}_{\mathcal{C}}^{-1}(\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}_0^{(\mathcal{C})})}{2T}\right\} \cdot \mathrm{d}\bar{\mathbb{W}}_{\mathbf{\Lambda}}(\mathfrak{X}) \\
&\quad \cdot \exp\left\{-\sum_{c\in\mathcal{C}} \frac{(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \boldsymbol{x}_0^{(c)})^\intercal \mathbf{\Lambda}_c^{-1}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \boldsymbol{x}_0^{(c)})}{2T}\right\} \cdot \prod_{c\in\mathcal{C}} \left[\exp\left\{-\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right) \mathrm{d}t\right\}\right] \\
&= \left[\prod_{c\in\mathcal{C}} f_c\left(\boldsymbol{x}_0^{(c)}\right)\right] \cdot \exp\left\{-\sum_{c\in\mathcal{C}} \frac{(\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}_0^{(c)})^\intercal \mathbf{\Lambda}_c^{-1}(\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}_0^{(c)})}{2T}\right\} \cdot \mathrm{d}\bar{\mathbb{W}}_{\mathbf{\Lambda}}(\mathfrak{X}) \\
&\quad \cdot \prod_{c\in\mathcal{C}} \left[\exp\left\{-\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right) \mathrm{d}t\right\}\right],
\end{aligned} \tag{7.6}$$

where $\bar{\mathbb{W}}_{\mathbf{\Lambda}}$ denotes the law of $|\mathcal{C}|$ independent Brownian bridges $\{\boldsymbol{X}_t^{(c)}, t \in [0,T]\}_{c\in\mathcal{C}}$ starting at $\boldsymbol{X}_0^{(c)} := \boldsymbol{x}_0^{(c)}$ and ending at $\boldsymbol{X}_T^{(c)} := \boldsymbol{y}^{(\mathcal{C})}$ (with covariance $\mathbf{\Lambda}_c$). Let $g_{\mathcal{C}}(\vec{\boldsymbol{x}}_0^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})})$ denote the

marginal distribution of $\mathbb{F}$ at $\vec{\boldsymbol{x}}_0^{(\mathcal{C})}$ and $\vec{\boldsymbol{x}}_T^{(\mathcal{C})} =: \boldsymbol{y}^{(\mathcal{C})}$, then we have

$$\begin{aligned} g_{\mathcal{C}}(\vec{\boldsymbol{x}}_0^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})}) &\propto \prod_{c\in\mathcal{C}} \left[f_c\left(\boldsymbol{x}_0^{(c)}\right)\right] \cdot \exp\left\{ -\sum_{c\in\mathcal{C}} \frac{(\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}_0^{(c)})^\intercal \boldsymbol{\Lambda}_c^{-1} (\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}_0^{(c)})}{2T} \right\} \\ &\quad \cdot \prod_{c\in\mathcal{C}} \left[\exp\left\{ -\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right) \mathrm{d}t \right\}\right], \\ &= \prod_{c\in\mathcal{C}} \left[f_c^2\left(\boldsymbol{x}_0^{(c)}\right) \cdot p_c\left(\boldsymbol{y}^{(\mathcal{C})} \middle| \boldsymbol{x}_0^{(c)}\right) \cdot \frac{1}{f_c\left(\boldsymbol{y}^{(\mathcal{C})}\right)} \right], \end{aligned} \tag{7.7}$$

where

$$\begin{aligned} p_c\left(\boldsymbol{y}^{(\mathcal{C})} \middle| \boldsymbol{x}_0^{(c)}\right) &\propto \frac{f_c\left(\boldsymbol{y}^{(\mathcal{C})}\right)}{f_c\left(\boldsymbol{x}_0^{(c)}\right)} \cdot \exp\left\{ -\frac{(\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}_0^{(c)})^\intercal \boldsymbol{\Lambda}_c^{-1} (\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}_0^{(c)})}{2T} \right\} \\ &\quad \cdot \mathbb{E}_{\mathbb{W}_{\boldsymbol{\Lambda}_c}} \left[\exp\left\{ -\int_0^T \phi_c\left(\boldsymbol{X}_t^{(c)}\right) \mathrm{d}t \right\}\right]. \end{aligned} \tag{7.8}$$

Recall from Proposition 6.1.1, using the Dacunha-Castelle representation [Dacunha-Castelle and Florens-Zmirou, 1986, Lemma 1] (see Section 4.3.4), this is the transition density density of a Langevin diffusion with covariance matrix $\boldsymbol{\Lambda}_c$ over time $t \in [0, T]$ (as given in (6.5)). Critically, this diffusion process has invariant density proportional to f_c^2, so

$$\int p\left(\boldsymbol{y}^{(\mathcal{C})} \middle| \boldsymbol{x}_0^{(c)}\right) f_c^2\left(\boldsymbol{x}_0^{(c)}\right) \, \mathrm{d}\boldsymbol{x}_0^{(c)} = f_c^2\left(\boldsymbol{y}^{(\mathcal{C})}\right).$$

By integrating out $\vec{\boldsymbol{x}}_0^{(\mathcal{C})}$ in (7.7), we can see that $g_{\mathcal{C}}(\vec{\boldsymbol{x}}_0^{(\mathcal{C})}, \boldsymbol{y}^{(\mathcal{C})})$ admits $f^{(\mathcal{C})}$ as a marginal. ■

7.1.2 Methodology

Theorem 7.1.1 suggests that we can simulate from the fusion target density $f^{(\mathcal{C})}$ by simulating $\mathfrak{X} \sim \mathbb{F}$ and retaining the T time marginal, $\boldsymbol{y}^{(\mathcal{C})}$. As suggested by the theory, we do so by means of simulating a number of proposals $\mathfrak{X} \sim \mathbb{P}$ and accepting (or importance weighting) the terminal time marginal $\boldsymbol{y}^{(\mathcal{C})}$ with probability (or weight) proportional to the Radon-Nikodým derivative in (7.2). As such, we need to consider: (i) how to simulate proposals from $\mathfrak{X} \sim \mathbb{P}$ (outlined in Section 7.1.2.1); and (ii) how to compute the Radon-Nikodým correction (7.2) (outlined in Section 7.1.2.2). We then present our proposed complete methodology in Section 7.1.2.3.

7.1.2.1 Simulating from the proposal measure

First, we consider how to simulate proposals from $\mathfrak{X} \sim \mathbb{P}$. We begin by noting that the initialisation of the proposal measure given by (7.1) at time $t = 0$ only requires independent draws from the $|\mathcal{C}|$ sub-posteriors that we wish to unify, which in this thesis, we assume we have access to. If independent sampling is not feasible, it is possible to obtain approximate sub-posterior samples

using MCMC (see Section 5.2.3.1 for a discussion on the impacts of using approximate sub-posterior samples for Fusion). Further, although paths $\mathfrak{X} \sim \mathbb{P}$ are infinite dimensional random variables (and so we cannot draw entire sample paths from $\mathbb{P}$), it is sufficient for our needs to simulate (exactly) the paths at a finite collection of times provided we can ensure that we are able to simulate the path (*exactly*) at time T. For clarity, we only consider simulating $\mathfrak{X}$ at times given by the following auxiliary temporal partition,

$$\mathcal{P} = \{t_0, t_1, \ldots, t_n : 0 =: t_0 < t_1 < \cdots < t_n := T\}. \tag{7.9}$$

We let $\Delta_j := t_j - t_{j-1}$ and for notational simplicity, subscripts are suppressed when considering the processes at times given in the temporal partition. In particular, let $\boldsymbol{x}_j^{(c)}$ denote $\boldsymbol{x}_{t_j}^{(c)}$, and let $\vec{\boldsymbol{x}}_j^{(\mathcal{C})}$ denote $\vec{\boldsymbol{x}}_{t_j}^{(\mathcal{C})}$. The following theorem tells us how to simulate from $\mathbb{P}$ without discretisation error.

Theorem 7.1.2. *Let $\mathcal{C} := (c_1, \ldots, c_{|\mathcal{C}|})$ denote the index set representing the sub-posteriors we wish to unify, then if $\mathfrak{X}$ satisfies* (7.1)*, then under the proposal measure, $\mathbb{P}$, we have*

(a) For $s < t$,

$$\vec{\boldsymbol{X}}_t^{(\mathcal{C})} \middle| \left(\vec{\boldsymbol{X}}_s^{(\mathcal{C})} = \vec{\boldsymbol{x}}_s^{(\mathcal{C})} \right) \sim \mathcal{N}_{|\mathcal{C}|d} \left(\vec{\boldsymbol{M}}_{s,t}^{(\mathcal{C})}, \boldsymbol{V}_{s,t} \right), \tag{7.10}$$

where $\vec{\boldsymbol{M}}_{s,t}^{(\mathcal{C})} \in \mathbb{R}^{|\mathcal{C}| \times d} := \left(\boldsymbol{M}_{s,t}^{(c_1)}, \ldots, \boldsymbol{M}_{s,t}^{(c_{|\mathcal{C}|})} \right)$ with

$$\boldsymbol{M}_{s,t}^{(c)} = \frac{T-t}{T-s} \boldsymbol{x}_s^{(c)} + \frac{t-s}{T-s} \tilde{\boldsymbol{x}}_s, \tag{7.11}$$

and

$$\boldsymbol{V}_{s,t} = \begin{pmatrix} \boldsymbol{\Gamma}_{11} & \boldsymbol{\Gamma}_{12} & \ldots & \boldsymbol{\Gamma}_{1|\mathcal{C}|} \\ \boldsymbol{\Gamma}_{21} & \boldsymbol{\Gamma}_{22} & \ldots & \boldsymbol{\Gamma}_{2|\mathcal{C}|} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Gamma}_{|\mathcal{C}|1} & \boldsymbol{\Gamma}_{|\mathcal{C}|2} & \ldots & \boldsymbol{\Gamma}_{|\mathcal{C}||\mathcal{C}|} \end{pmatrix} \in \mathbb{R}^{|\mathcal{C}|d \times |\mathcal{C}|d}, \tag{7.12}$$

where for $i, j = 1, \ldots, |\mathcal{C}|$,

$$\boldsymbol{\Gamma}_{ii} = \frac{(t-s)(T-t)}{T-s} \boldsymbol{\Lambda}_{c_i} + \frac{(t-s)^2}{T-s} \boldsymbol{\Lambda}_{\mathcal{C}} \in \mathbb{R}^{d \times d}, \tag{7.13}$$

$$\boldsymbol{\Gamma}_{ij} = \frac{(t-s)^2}{T-s} \boldsymbol{\Lambda}_{\mathcal{C}} \in \mathbb{R}^{d \times d}. \tag{7.14}$$

(b) For each $c \in \mathcal{C}$, the distribution of $\{\boldsymbol{X}_q^{(c)}, s \leq q \leq t\}$ given endpoints $\boldsymbol{X}_s^{(c)} = \boldsymbol{x}_s^{(c)}$ and $\boldsymbol{X}_t^{(c)} = \boldsymbol{x}_t^{(c)}$ is a Brownian bridge with covariance matrix $\boldsymbol{\Lambda}_c$, so

$$\boldsymbol{X}_u^{(c)} \middle| \left(\boldsymbol{x}_s^{(c)}, \boldsymbol{x}_t^{(c)} \right) \sim \mathcal{N}_d \left(\frac{(t-q)\boldsymbol{x}_s^{(c)} + (q-s)\boldsymbol{x}_t^{(c)}}{t-s}, \frac{(t-q)(q-s)}{t-s} \boldsymbol{\Lambda}_c \right). \tag{7.15}$$

Proof. Theorem (a): We begin by deriving the joint density of $\vec{\boldsymbol{X}}_t^{(\mathcal{C})}$ conditional on the state at time s, $\vec{\boldsymbol{x}}_s^{(\mathcal{C})}$. Firstly, consider the $d(|\mathcal{C}|+1)$ dimensional joint density of $\vec{\boldsymbol{X}}_t^{(\mathcal{C})}$ *and* end-point $\boldsymbol{y}^{(\mathcal{C})}$ conditional on $\vec{\boldsymbol{x}}_s^{(\mathcal{C})}$, which we denote as p_1, then

$$-2\log p_1 = D_1 + D_2,$$

where D_1 is the log-density of $\boldsymbol{y}^{(\mathcal{C})}$ conditional on $\vec{\boldsymbol{x}}_s^{(\mathcal{C})}$ and given by

$$D_1 = \sum_{c\in\mathcal{C}} \frac{(\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}_s^{(c)})^\intercal \boldsymbol{\Lambda}_c^{-1} (\boldsymbol{y}^{(\mathcal{C})} - \boldsymbol{x}_s^{(c)})}{T-s} + k_1$$

where k_1 is a constant; D_2 is the log-density of $\vec{\boldsymbol{X}}_t^{(\mathcal{C})}$ conditional on $\vec{\boldsymbol{x}}_s^{(\mathcal{C})}$ and $\boldsymbol{y}^{(\mathcal{C})}$ (which is simply the log-density of $|\mathcal{C}|$ Brownian bridges with respective covariance matrices $\boldsymbol{\Lambda}_c$ for $c\in\mathcal{C}$), given by

$$D_2 = \sum_{c\in\mathcal{C}} \frac{T-s}{(t-s)(T-t)} \left[\boldsymbol{x}_t^{(c)} - \frac{t-s}{T-s}\boldsymbol{y}^{(\mathcal{C})} - \frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right]^\intercal \boldsymbol{\Lambda}_c^{-1} \left[\boldsymbol{x}_t^{(c)} - \frac{t-s}{T-s}\boldsymbol{y}^{(\mathcal{C})} - \frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right] + k_2,$$

where k_2 is a constant. We therefore have

$$\begin{aligned}
-2\log p_1 &= \frac{(\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}_s^{(\mathcal{C})})^\intercal \boldsymbol{\Lambda}_{\mathcal{C}}^{-1} (\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}_s^{(\mathcal{C})})}{T-s} \\
&\quad + \sum_{c\in\mathcal{C}} \left[\frac{t-s}{(T-t)(T-s)} \boldsymbol{y}^{(\mathcal{C})\intercal} \boldsymbol{\Lambda}_c^{-1} \boldsymbol{y}^{(\mathcal{C})} - \frac{2}{T-t}\boldsymbol{y}^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_t^{(c)} + \frac{2}{T-s}\boldsymbol{y}^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_s^{(c)}\right] \\
&\quad + \sum_{c\in\mathcal{C}} \frac{T-s}{(t-s)(T-t)} \left[\boldsymbol{x}_t^{(c)} - \frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right]^\intercal \boldsymbol{\Lambda}_c^{-1} \left[\boldsymbol{x}_t^{(c)} - \frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right] + k_3 \\
&= \frac{1}{T-s}\left[\boldsymbol{y}^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\boldsymbol{y}^{(\mathcal{C})} - 2\boldsymbol{y}^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\tilde{\boldsymbol{x}}_s^{(\mathcal{C})}\right] \\
&\quad + \left[\frac{t-s}{(T-t)(T-s)} \boldsymbol{y}^{(\mathcal{C})\intercal} \boldsymbol{\Lambda}_{\mathcal{C}}^{-1} \boldsymbol{y}^{(\mathcal{C})} - \frac{2}{T-t}\boldsymbol{y}^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})} + \frac{2}{T-s}\boldsymbol{y}^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\tilde{\boldsymbol{x}}_s^{(\mathcal{C})}\right] \\
&\quad + \sum_{c\in\mathcal{C}} \frac{T-s}{(t-s)(T-t)} \left[\boldsymbol{x}_t^{(c)} - \frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right]^\intercal \boldsymbol{\Lambda}_c^{-1} \left[\boldsymbol{x}_t^{(c)} - \frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right] + k_4 \\
&= \left[\frac{1}{T-s} + \frac{t-s}{(T-t)(T-s)}\right] \boldsymbol{y}^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\boldsymbol{y}^{(\mathcal{C})} - \frac{2}{T-t}\boldsymbol{y}^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})} \\
&\quad + \sum_{c\in\mathcal{C}} \frac{T-s}{(t-s)(T-t)} \left[\boldsymbol{x}_t^{(c)} - \frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right]^\intercal \boldsymbol{\Lambda}_c^{-1} \left[\boldsymbol{x}_t^{(c)} - \frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right] + k_4 \\
&= \frac{1}{T-t}\left[\boldsymbol{y}^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\boldsymbol{y}^{(\mathcal{C})} - 2\boldsymbol{y}^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})}\right] \\
&\quad + \sum_{c\in\mathcal{C}} \frac{T-s}{(t-s)(T-t)} \left[\boldsymbol{x}_t^{(c)} - \frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right]^\intercal \boldsymbol{\Lambda}_c^{-1} \left[\boldsymbol{x}_t^{(c)} - \frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right] + k_4 \\
&= \frac{1}{T-t}\left[(\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}_t^{(\mathcal{C})})^\intercal \boldsymbol{\Lambda}_{\mathcal{C}}^{-1} (\boldsymbol{y}^{(\mathcal{C})} - \tilde{\boldsymbol{x}}_t^{(\mathcal{C})})\right] - \frac{1}{T-t}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})}
\end{aligned}$$

$$+\sum_{c\in\mathcal{C}}\frac{T-s}{(t-s)(T-t)}\left[\boldsymbol{x}_t^{(c)}-\frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right]^\intercal\boldsymbol{\Lambda}_c^{-1}\left[\boldsymbol{x}_t^{(c)}-\frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right]+k_4,$$

where k_3 and k_4 are constants, and $\boldsymbol{\Lambda}_{\mathcal{C}}:=\left(\sum_{c\in\mathcal{C}}\boldsymbol{\Lambda}_c^{-1}\right)^{-1}$.

Next, we integrate out $\boldsymbol{y}^{(\mathcal{C})}$ to obtain the $d|\mathcal{C}|$-dimensional density of $\vec{\boldsymbol{X}}_t$ conditional on $\vec{\boldsymbol{x}}_s^{(\mathcal{C})}$, which we denote p_2:

$$\begin{aligned}-2\log p_2&=-\frac{1}{T-t}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})}+\sum_{c\in\mathcal{C}}\frac{T-s}{(t-s)(T-t)}\left[\boldsymbol{x}_t^{(c)}-\frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right]^\intercal\boldsymbol{\Lambda}_c^{-1}\left[\boldsymbol{x}_t^{(c)}-\frac{T-t}{T-s}\boldsymbol{x}_s^{(c)}\right]+k_5\\&=-\frac{1}{T-t}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})}+\sum_{c\in\mathcal{C}}\frac{T-s}{(t-s)(T-t)}\left[\boldsymbol{x}_t^{(c)\intercal}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_t^{(c)}-2\left(\frac{T-t}{T-s}\right)\boldsymbol{x}_t^{(c)\intercal}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_s^{(c)}\right]+k_6,\end{aligned}$$

where k_5 and k_6 are constants. Noting that

$$\begin{aligned}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})\intercal}\boldsymbol{\Lambda}_{\mathcal{C}}^{-1}\tilde{\boldsymbol{x}}_t^{(\mathcal{C})}&=\left[\left(\sum_{c\in\mathcal{C}}\boldsymbol{\Lambda}_c^{-1}\right)^{-1}\left(\sum_{c\in\mathcal{C}}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_t^{(c)}\right)\right]^\intercal\left(\sum_{c\in\mathcal{C}}\boldsymbol{\Lambda}_c^{-1}\right)\left[\left(\sum_{c\in\mathcal{C}}\boldsymbol{\Lambda}_c^{-1}\right)^{-1}\left(\sum_{c\in\mathcal{C}}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_t^{(c)}\right)\right]\\&=\left(\sum_{c\in\mathcal{C}}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_t^{(c)}\right)^\intercal\boldsymbol{\Lambda}_{\mathcal{C}}\left(\sum_{c\in\mathcal{C}}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_t^{(c)}\right)\\&=\sum_{i,j\in\mathcal{C}}\boldsymbol{x}_t^{(i)\intercal}\left(\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\Lambda}_{\mathcal{C}}\boldsymbol{\Lambda}_j^{-1}\right)\boldsymbol{x}_t^{(j)}.\end{aligned}$$

So we have,

$$\begin{aligned}-2\log p_2&=-\frac{1}{T-t}\sum_{i,j\in\mathcal{C}}\boldsymbol{x}_t^{(i)\intercal}\left(\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\Lambda}_{\mathcal{C}}\boldsymbol{\Lambda}_j^{-1}\right)\boldsymbol{x}_t^{(j)}+\frac{T-s}{(t-s)(T-t)}\sum_{c\in\mathcal{C}}\boldsymbol{x}_t^{(c)\intercal}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_t^{(c)}\\&\quad-\frac{2}{t-s}\sum_{c\in\mathcal{C}}\boldsymbol{x}_t^{(c)\intercal}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_s^{(c)}+k_6\\&=\frac{T-s}{(t-s)(T-t)}\sum_{c\in\mathcal{C}}\boldsymbol{x}_t^{(c)\intercal}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_t^{(c)}-\frac{1}{T-t}\sum_{c\in\mathcal{C}}\boldsymbol{x}_t^{(c)\intercal}\left(\boldsymbol{\Lambda}_c^{-1}\boldsymbol{\Lambda}_{\mathcal{C}}\boldsymbol{\Lambda}_c^{-1}\right)\boldsymbol{x}_t^{(c)}\\&\quad-\frac{1}{T-t}\sum_{\substack{i,j\in\mathcal{C}\\i\neq j}}\boldsymbol{x}_t^{(i)\intercal}\left(\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\Lambda}_{\mathcal{C}}\boldsymbol{\Lambda}_j^{-1}\right)\boldsymbol{x}_t^{(j)}-\frac{2}{t-s}\sum_{c\in\mathcal{C}}\boldsymbol{x}_t^{(c)\intercal}\boldsymbol{\Lambda}_c^{-1}\boldsymbol{x}_s^{(c)}+k_6\\&=\vec{\boldsymbol{x}}_t^\intercal\boldsymbol{V}_{s,t}^{-1}\vec{\boldsymbol{x}}_t^{(\mathcal{C})}-\frac{2}{t-s}\vec{\boldsymbol{x}}_t^\intercal\boldsymbol{L}^{-1}\vec{\boldsymbol{x}}_s^{(\mathcal{C})}+k_6\end{aligned}$$

where

$$\boldsymbol{V}_{s,t}^{-1}=\begin{pmatrix}\boldsymbol{\Sigma}_{11}^{-1}&\boldsymbol{\Sigma}_{12}^{-1}&\cdots&\boldsymbol{\Sigma}_{1|\mathcal{C}|}^{-1}\\\boldsymbol{\Sigma}_{21}^{-1}&\boldsymbol{\Sigma}_{22}^{-1}&\cdots&\boldsymbol{\Sigma}_{2|\mathcal{C}|}^{-1}\\\vdots&\vdots&\ddots&\vdots\\\boldsymbol{\Sigma}_{|\mathcal{C}|1}^{-1}&\boldsymbol{\Sigma}_{|\mathcal{C}|2}^{-1}&\cdots&\boldsymbol{\Sigma}_{|\mathcal{C}||\mathcal{C}|}^{-1}\end{pmatrix}\in\mathbb{R}^{|\mathcal{C}|d\times|\mathcal{C}|d},\tag{7.16}$$

with

$$\boldsymbol{\Sigma}_{ii}^{-1} = \frac{T-s}{(t-s)(T-t)}\boldsymbol{\Lambda}_{c_i}^{-1} - \frac{1}{T-t}\left(\boldsymbol{\Lambda}_{c_i}^{-1}\boldsymbol{\Lambda}_{\mathcal{C}}\boldsymbol{\Lambda}_{c_i}^{-1}\right) \in \mathbb{R}^{d\times d},$$
$$\boldsymbol{\Sigma}_{ij}^{-1} = -\frac{1}{T-t}\left(\boldsymbol{\Lambda}_{c_i}^{-1}\boldsymbol{\Lambda}_{\mathcal{C}}\boldsymbol{\Lambda}_{c_j}^{-1}\right) \in \mathbb{R}^{d\times d},$$

for $i, j = 1, \ldots, |\mathcal{C}|$, and

$$\boldsymbol{L}^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}_{c_1}^{-1} & \mathbf{0}_{d\times d} & \ldots & \mathbf{0}_{d\times d} \\ \mathbf{0}_{d\times d} & \boldsymbol{\Lambda}_{c_2}^{-1} & \ldots & \mathbf{0}_{d\times d} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \ldots & \boldsymbol{\Lambda}_{c_{|\mathcal{C}|}}^{-1} \end{pmatrix} \in \mathbb{R}^{|\mathcal{C}|d\times|\mathcal{C}|d},$$

where $\mathbf{0}_{d\times d}$ is the $d \times d$ with all elements zero. We finally complete the square to get

$$-2\log p_2 = \vec{\boldsymbol{x}}_t^{(\mathcal{C})}\boldsymbol{V}_{s,t}^{-1}\vec{\boldsymbol{x}}_t^{(\mathcal{C})} - 2\vec{\boldsymbol{x}}_t^{(\mathcal{C})}\boldsymbol{V}_{s,t}^{-1}\vec{\boldsymbol{M}}_{s,t}^{(\mathcal{C})} + k_6,$$

where

$$\vec{\boldsymbol{M}}_{s,t}^{(\mathcal{C})} = \frac{\boldsymbol{V}_{s,t}\boldsymbol{L}^{-1}\vec{\boldsymbol{x}}_s^{(\mathcal{C})}}{t-s}.$$

Inverting $\boldsymbol{V}_{s,t}^{-1}$ in (7.16), we obtain (7.12) and subsequently we can get the expression for $\boldsymbol{M}_{s,t}^{(c)}$ in (7.11) to prove the statement in part (a) of Theorem 7.1.1.

For part (b), for $c \in \mathcal{C}$, the law of $\{\boldsymbol{X}_t^{(c)}, t \in (0,T)\}$ conditional on endpoints $\boldsymbol{x}_0^{(c)}$ and $\boldsymbol{y}^{(\mathcal{C})}$ is that of a Brownian bridge (see Section 4.1.2). This statement in the theorem holds from the standard properties of Brownian bridges (with covariance matrix $\boldsymbol{\Lambda}_c$). In particular, considering the distribution of $\boldsymbol{X}_q^{(c)}$ at an intermediate point $q \in (s,t)$ given the positions $\boldsymbol{X}_s^{(c)} = \boldsymbol{x}_s^{(c)}$ and $\boldsymbol{X}_t^{(c)} = \boldsymbol{x}_t^{(c)}$ at times s and t respectively, then we have

$$\begin{aligned}
&\mathbb{P}\left(\boldsymbol{X}_q = \boldsymbol{w} \middle| \boldsymbol{X}_s^{(c)} = \boldsymbol{x}_s^{(c)}, \boldsymbol{X}_t^{(c)} = \boldsymbol{x}_t^{(c)}\right) \\
&\quad\propto \mathbb{P}\left(\boldsymbol{X}_t^{(c)} = \boldsymbol{x}_t^{(c)} \middle| \boldsymbol{X}_s^{(c)} = \boldsymbol{x}_s^{(c)}, \boldsymbol{X}_q = \boldsymbol{w}\right) \cdot \mathbb{P}\left(\boldsymbol{X}_q = \boldsymbol{w} \middle| \boldsymbol{X}_s^{(c)} = \boldsymbol{x}_s^{(c)}\right) \\
&\quad\propto \mathbb{P}\left(\boldsymbol{X}_t^{(c)} = \boldsymbol{x}_t^{(c)} \middle| \boldsymbol{X}_q = \boldsymbol{w}\right) \cdot \mathbb{P}\left(\boldsymbol{X}_q = \boldsymbol{w} \middle| \boldsymbol{X}_s^{(c)} = \boldsymbol{x}_s^{(c)}\right) \\
&\quad\propto \exp\left(-\frac{(\boldsymbol{x}_t^{(c)} - \boldsymbol{w})^\intercal \boldsymbol{\Lambda}_c^{-1}(\boldsymbol{x}_t^{(c)} - \boldsymbol{w})}{2(t-q)}\right) \cdot \exp\left(-\frac{(\boldsymbol{w} - \boldsymbol{x}_s^{(c)})^\intercal \boldsymbol{\Lambda}_c^{-1}(\boldsymbol{w} - \boldsymbol{x}_s^{(c)})}{2(q-s)}\right),
\end{aligned}$$

and hence we arrive at the result in the statement. ■

As we can initialise a draw from $\mathbb{P}$, and from Theorem 7.1.2 we can simulate from its transition density, we can now explicitly express the $d(n|\mathcal{C}|+1)$-dimensional density of the $|\mathcal{C}|d$-dimensional Markov process at the $(n+1)$ time marginals given by the temporal partition under $\mathbb{P}$ by iterative

simulation from the transition density:

$$h_{\mathcal{C}}\left(\vec{\boldsymbol{x}}_0^{(\mathcal{C})},\ldots,\vec{\boldsymbol{x}}_{n-1}^{(\mathcal{C})},\boldsymbol{y}^{(\mathcal{C})}\right) \propto \prod_{c\in\mathcal{C}}\left[f_c\left(\boldsymbol{x}_0^{(c)}\right)\right]\cdot\prod_{j=1}^{n}\mathcal{N}_{|\mathcal{C}|d}\left(\vec{\boldsymbol{x}}_j^{(\mathcal{C})}\middle|\vec{\boldsymbol{M}}_j^{(\mathcal{C})},\boldsymbol{V}_j\right), \tag{7.17}$$

where $\mathcal{N}_d(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ denotes the density of a d-dimensional Normal distribution (evaluated at $\boldsymbol{x}$) with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ and for notational convenience we let $\vec{\boldsymbol{M}}_j^{(\mathcal{C})}=\vec{\boldsymbol{M}}_{t_{j-1},t_j}^{(\mathcal{C})}$ and $\boldsymbol{V}_j=\boldsymbol{V}_{t_{j-1},t_j}$. Recall that $\vec{\boldsymbol{x}}_0^{(\mathcal{C})}:=(\boldsymbol{x}_0^{(c_1)},\ldots,\boldsymbol{x}_0^{(c_{|\mathcal{C}|})})$ where $\mathcal{C}:=\{c_1,\ldots,c_{|\mathcal{C}|}\}$ and $\boldsymbol{y}^{(\mathcal{C})}:=\boldsymbol{x}_n^{(c)}$ for each $c\in\mathcal{C}$.

7.1.2.2 Radon-Nikodým correction of the proposal

Now, we draw our consideration to the second step of how to compute the Radon-Nikodým correction of (7.2), given we have drawn our proposal from $\mathbb{P}$ restricted to the times given by the partition $\mathcal{P}$. Factorising the Radon-Nikodým derivative in (7.2) according to the temporal partition $\mathcal{P}$, the $d(n|\mathcal{C}|+1)$-dimensional density under $\mathbb{F}$ is

$$g_{\mathcal{C}}\left(\vec{\boldsymbol{x}}_0^{(\mathcal{C})},\ldots,\vec{\boldsymbol{x}}_{n-1}^{(\mathcal{C})},\boldsymbol{y}^{(\mathcal{C})}\right)\propto h_{\mathcal{C}}\left(\vec{\boldsymbol{x}}_0^{(\mathcal{C})},\ldots,\vec{\boldsymbol{x}}_{n-1}^{(\mathcal{C})},\boldsymbol{y}^{(\mathcal{C})}\right)\cdot\prod_{j=0}^{n}\rho_j, \tag{7.18}$$

where ρ_0 is given in (6.7) and for $j=1,\ldots,n$,

$$\rho_j\left(\vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})},\vec{\boldsymbol{x}}_j^{(\mathcal{C})}\right)=\prod_{c\in\mathcal{C}}\mathbb{E}_{\mathbb{W}_{\boldsymbol{\Lambda}_c,j}}\left[\exp\left\{-\int_{t_{j-1}}^{t_j}\left(\phi_c\left(\boldsymbol{X}_t^{(c)}\right)-\boldsymbol{\Phi}_c\right)\right\}\right]\in(0,1], \tag{7.19}$$

where $\mathbb{W}_{\boldsymbol{\Lambda}_c,j}$ is the law of a Brownian bridge $\{\boldsymbol{X}_t^{(c)},t\in(t_{j-1},t_j)\}$ from $\boldsymbol{X}_{t_{j-1}}:=\boldsymbol{x}_{j-1}^{(c)}$ to $\boldsymbol{X}_{t_j}:=\boldsymbol{x}_j^{(c)}$ with covariance $\boldsymbol{\Lambda}_c$, and $\boldsymbol{\Phi}_c<\infty$ is a constant such that $\phi_c(\boldsymbol{x})\geq\boldsymbol{\Phi}_c$ for all $\boldsymbol{x}$ and each $c\in\mathcal{C}$. As with Dai et al. [2021], we note that we can avoid computation of the global lower bounds $\boldsymbol{\Phi}_c$ of ϕ_c, by considering a sequential Monte Carlo approach, since these are simply constants that will be cancelled during normalisation of the importance weights.

Although we cannot directly compute ρ_j for $j=1,\ldots,n$, we have seen several times in this thesis that we can construct unbiased estimators in a similar fashion to Beskos et al. [2008]; Fearnhead et al. [2008]. To recap, to find an unbiased estimator, we need to find upper and lower bounds for $\phi_c(\boldsymbol{X}_t^{(c)})$ for $t\in[t_{j-1},t_j]$. Beskos et al. [2008] noted that if we can bound a sample path $\boldsymbol{X}_{[t_{j-1},t_j]}^{(c)}\sim\mathbb{W}_{\boldsymbol{\Lambda}_c,j}$, then conditional on these *layers* (or bounds) of the sample path, we can find upper and lower bounds of ϕ_c denoted $U_j^{(c)}$ and $L_j^{(c)}$, respectively, such that $\phi_c(\boldsymbol{X}_t^{(c)})\in[L_j^{(c)},U_j^{(c)}]$ for $t\in[t_{j-1},t_j]$. To achieve this, let $R_c:=R_c(\boldsymbol{X}_{[t_{j-1},t_j]})$ denote the compact region (or *layer*) in which $\boldsymbol{X}_t^{(c)}$ is constrained in time $[t_{j-1},t_j]$. If $\boldsymbol{\Lambda}_c=\mathbb{I}_d$, then we can simulate a layer to which $\boldsymbol{X}_t^{(c)}\in R_c$ for $t\in[t_{j-1},t_j]$ by using algorithms outlined in Pollock et al. [2016, Section 7] and summarised in Section 4.2.2 (for instance Algorithm 4.2.4). In the case where $\boldsymbol{\Lambda}_c\neq\mathbb{I}_d$, we can still simulate R_c by appealing to a suitable transformation as we did in Section 6.1.2. Once we have

simulated layer information for $\boldsymbol{X}_t^{(c)}$ for $t \in [t_{j-1}, t_j]$, we can simulate the path at any required time marginals conditional on the simulated layer, $\boldsymbol{X}_t^{(c)} \sim \mathbb{W}_{\boldsymbol{\Lambda}_c,j}|R_c$ (via a transformation and applying Algorithm 4.2.5). Although it is possible to find bounds for ϕ_c given the compact set R_c in a problem specific manner, we can also find general (less tight) bounds which are provided in Section 6.1.2 in Proposition 6.1.3.

Once local bounds for ϕ_c are obtained, we can unbiasedly estimate ρ_j in (7.19) for $j = 1, \ldots, n$ by letting $\Delta_j := t_j - t_{j-1}$ and computing $a_j\tilde{\rho}_j$, where $a_j := \exp\{\sum_{c\in\mathcal{C}} \boldsymbol{\Phi}_c\Delta_j\}$ and

$$\tilde{\rho}_j\left(\vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})}, \vec{\boldsymbol{x}}_j^{(\mathcal{C})}\right) := \prod_{c\in\mathcal{C}}\left(\frac{\Delta_j^{\kappa_c}\cdot e^{-U_j^{(c)}\Delta_j}}{\kappa_c!\cdot p\left(\kappa_c|R_c\right)}\cdot\prod_{k_c=1}^{\kappa_c}\left[U_j^{(c)} - \phi_c\left(\boldsymbol{X}_{\xi_{c,k_c}}^{(c)}\right)\right]\right), \tag{7.20}$$

where $L_j^{(c)}$ and $U_j^{(c)}$ are constants such that $L_j^{(c)} \leq \phi\left(\boldsymbol{X}_t^{(c)}\right) \leq U_j^{(c)}$ for all $\boldsymbol{X}_t^{(c)} \sim \mathbb{W}_{\boldsymbol{\Lambda}_c,j}|R_c$, κ_c is a discrete random variable with conditional probabilities $\mathbb{P}[\kappa_c = k_c|R_c] := p(\kappa_c|R_c)$ and $\xi_{c,1}, \ldots, \xi_{c,\kappa_c} \overset{\text{iid}}{\sim} \mathcal{U}[t_{j-1}, t_j]$ for all $c \in \mathcal{C}$.

Theorem 7.1.3. *Let $a_j := \exp\{\sum_{c=1}^{C} \boldsymbol{\Phi}_c\Delta_j\}$, then for every $j = 1, \ldots, n$, $a_j\tilde{\rho}_j$ is an unbiased estimator of ρ_j. In particular, we have*

$$\begin{aligned}\rho_j &= \mathbb{E}\left[\mathbb{E}\left[\mathbb{E}\left[a_j\tilde{\rho}_j \mid \{\mathcal{R}_c, \boldsymbol{X}_{[t_{j-1},t_j]}^{(c)}, \kappa_c\}_{c\in\mathcal{C}}\right] \mid \{\mathcal{R}_c, \boldsymbol{X}_{[t_{j-1},t_j]}^{(c)}\}_{c\in\mathcal{C}},\right] \mid \{\mathcal{R}_c\}_{c\in\mathcal{C}}\right]\right] \\ &= \mathbb{E}_{\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{W}}|\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{K}}}\mathbb{E}_{\bar{\mathbb{U}}}\left[a_j\tilde{\rho}_j\right],\end{aligned} \tag{7.21}$$

where expectation subscripts denote the law with which they are taking; $\mathcal{R}$ denotes the law of $\{R_c \sim \mathcal{R}_c : c = 1, \ldots, C\}$, $\bar{\mathbb{W}}$ denotes the law of the C Brownian bridges $\{\mathbb{W}_{\boldsymbol{\Lambda}_c,j} : c = 1, \ldots, C\}$, $\bar{\mathbb{K}}$ denotes the law of $\{\kappa_c : c = 1, \ldots, C\}$ and $\bar{\mathbb{U}}$ denotes the law of $\{\xi_{c,1}, \ldots, \xi_{c,\kappa_c} : c = 1, \ldots, C\} \overset{iid}{\sim} \mathcal{U}[t_{j-1}, t_j]$.

Proof. Following in the style of Beskos et al. [2006a, 2008]; Fearnhead et al. [2008] (see Section 4.4) and Dai et al. [2021, Appendix B], for $j = 1, \ldots, n$, we have

$$\begin{aligned}&\mathbb{E}_{\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{W}}|\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{K}}}\mathbb{E}_{\bar{\mathbb{U}}}\left[a_j\tilde{\rho}_j\right] \\ &\quad= \mathbb{E}_{\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{W}}|\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{K}}}\mathbb{E}_{\bar{\mathbb{U}}}\left[\prod_{c\in\mathcal{C}}\left(\frac{\Delta_j^{\kappa_c}\cdot e^{-(U_j^{(c)}-\boldsymbol{\Phi}_c)\Delta_j}}{\kappa_c!\cdot p\left(\kappa_c|R_c\right)}\prod_{k_c=1}^{\kappa_c}\left(U_j^{(c)} - \phi_c\left(\boldsymbol{X}_{\xi_{c,k_c}}^{(c)}\right)\right)\right)\right] \\ &\quad= \mathbb{E}_{\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{W}}|\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{K}}}\left[\prod_{c\in\mathcal{C}}\left(\frac{\Delta_j^{\kappa_c}\cdot e^{-(U_j^{(c)}-\boldsymbol{\Phi}_c)\Delta_j}}{\kappa_c!\cdot p\left(\kappa_c|R_c\right)}\cdot\left[\int_{t_{j-1}}^{t_j}\frac{U_j^{(c)} - \phi_c\left(\boldsymbol{X}_t^{(c)}\right)}{\Delta_j}\,\mathrm{d}t\right]^{\kappa_c}\right)\right] \\ &\quad= \mathbb{E}_{\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{W}}|\bar{\mathcal{R}}}\left[\prod_{c\in\mathcal{C}}\left(\sum_{k_c=0}^{\infty}\frac{\Delta_j^{k_c}\cdot e^{-(U_j^{(c)}-\boldsymbol{\Phi}_c)\Delta_j}}{k_c!\cdot p\left(k_c|R_c\right)}\cdot\left[\int_{t_{j-1}}^{t_j}\frac{U_j^{(c)} - \phi_c\left(\boldsymbol{X}_t^{(c)}\right)}{\Delta_j}\,\mathrm{d}t\right]^{k_c}\right)\right]\end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}_{\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{W}}|\bar{\mathcal{R}}}\left[\prod_{c\in\mathcal{C}} e^{-(U_j^{(c)}-\boldsymbol{\Phi}_c)\Delta_j}\cdot\left(\sum_{k_c=0}^{\infty}\frac{\Delta_j^{k_c}}{k_c!\cdot p\left(k_c|R_c\right)}\cdot\left[\int_{t_{j-1}}^{t_j}\frac{U_j^{(c)}-\phi_c\left(\boldsymbol{X}_t^{(c)}\right)}{\Delta_j}\,\mathrm{d}t\right]^{k_c}\right)\right]\\
&= \mathbb{E}_{\bar{\mathcal{R}}}\mathbb{E}_{\bar{\mathbb{W}}|\bar{\mathcal{R}}}\left[\prod_{c\in\mathcal{C}} e^{-(U_j^{(c)}-\boldsymbol{\Phi}_c)\Delta_j}\cdot\exp\left\{\int_{t_{j-1}}^{t_j}\left(U_j^{(c)}-\phi_c\left(\boldsymbol{X}_t^{(c)}\right)\right)\,\mathrm{d}t\right\}\right]\\
&= \prod_{c\in\mathcal{C}}\mathbb{E}_{\mathbb{W}_{\boldsymbol{\Lambda}_c,j}}\left[\exp\left\{-\int_{t_{j-1}}^{t_j}\left(\phi_c\left(\boldsymbol{X}_t^{(c)}\right)-\boldsymbol{\Phi}_c\right)\,\mathrm{d}t\right\}\right] =: \rho_j,
\end{aligned}
$$

and hence $a_j\tilde{\rho}_j$ is an unbiased estimator for ρ_j. ■

This unbiased estimator for ρ_j allows for significant flexibility in choosing the law $\mathbb{K}$. Throughout this thesis, the main two choices that we consider in are the GPE-1 and GPE-2 estimators of Fearnhead et al. [2008] (see Section 4.4.3) which lead to the following two estimators for ρ_j:

Definition 7.1.1. (GPE-1 for ρ_j (7.19)): Choosing the law of $\kappa_c \sim \mathrm{Poi}\big((U_j^{(c)}-L_j^{(c)})\Delta_j\big)$ for $c\in\mathcal{C}$ leads to the following estimator:

$$
\tilde{\rho}_j^{(a)}\left(\vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})},\vec{\boldsymbol{x}}_j^{(\mathcal{C})}\right) := \prod_{c\in\mathcal{C}}\left(e^{-L_j^{(c)}\Delta_j}\cdot\prod_{k_c=1}^{\kappa_c}\left[\frac{U_j^{(c)}-\phi_c\left(\boldsymbol{X}_{\xi_{c,k_c}}^{(c)}\right)}{U_j^{(c)}-L_j^{(c)}}\right]\right), \tag{7.22}
$$

where $\exp\{\sum_{c=1}^{C}\boldsymbol{\Phi}_c\Delta_j\}\cdot\tilde{\rho}_j^{(a)}$ is an unbiased estimator for ρ_j.

Definition 7.1.2. (GPE-2 for ρ_j (7.19)): Choosing the law of $\kappa_c\sim\mathrm{NB}(\gamma_c,\beta_c)$ for $c\in\mathcal{C}$ with

$$
\gamma_c := U_j^{(c)}\Delta_j - \int_{t_{j-1}}^{t_j}\phi_c\left(\boldsymbol{x}_{j-1}^{(c)}\cdot\frac{t_j-s}{\Delta_j}+\boldsymbol{x}_j^{(c)}\cdot\frac{s-t_{j-1}}{\Delta_j}\right)\mathrm{d}s, \tag{7.23}
$$

leads to the following estimator:

$$
\tilde{\rho}_j^{(b)}\left(\vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})},\vec{\boldsymbol{x}}_j^{(\mathcal{C})}\right) := \prod_{c\in\mathcal{C}}\left(e^{-U_j^{(c)}\Delta_j}\cdot\frac{\Delta_j^{\kappa_c}\cdot\Gamma(\beta_c)\cdot(\beta_c+\gamma_c)^{\beta_c+\kappa_c}}{\Gamma(\beta_c+\kappa_c)\beta_c^{\beta_c}\gamma_c^{\kappa_c}}\cdot\prod_{k_c=1}^{\kappa_c}\left[U_j^{(c)}-\phi_c\left(\boldsymbol{X}_{\xi_{c,k_c}}^{(c)}\right)\right]\right), \tag{7.24}
$$

where $\exp\{\sum_{c=1}^{C}\boldsymbol{\Phi}_c\Delta_j\}\cdot\tilde{\rho}_j^{(b)}$ is an unbiased estimator for ρ_j.

As we discuss in Section 7.1.2.3, we will be embedding this estimator within a SMC framework, and thus the critical consideration when choosing the law $\mathbb{K}$ is to minimise the variance estimator. In our subsequent simulations, we typically choose the GPE-2 estimator in Condition 7.1.2 since it has been empirically shown to have superior performance in Fearnhead et al. [2008, Section 5] and Dai et al. [2021, Section 3.5]. We summarise the approach to simulating $\tilde{\rho}_j$ in Algorithm 7.1.1.

Algorithm 7.1.1 Simulating $\tilde{\rho}_j$.

1. For $c \in \mathcal{C}$
   (a) $\boldsymbol{z}_{j-1}^{(c)}, \boldsymbol{z}_j^{(c)}$: Transform the path, setting $\boldsymbol{z}_{j-1}^{(c)} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{x}_{j-1}^{(c)}$, and $\boldsymbol{z}_j^{(c)} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{x}_j^{(c)}$.
   (b) R_c: Set $R_c := \boldsymbol{\Lambda}_c^{\frac{1}{2}} R_c^{(z)}$, where $R_c^{(z)} \sim \mathcal{R}_c^{(z)}$ as per Algorithm 4.2.4.
   (c) $L_j^{(c)}, U_j^{(c)}$: Compute lower and upper bounds, $L_X^{(c)}$ and $U_X^{(c)}$, of $\phi_c(\boldsymbol{x})$ for $\boldsymbol{x} \in R_c$ (as per (6.11) and (6.12), or otherwise).
   (d) p_c: Choose $p(\cdot|R_c)$ using either GPE-1 (Condition 7.1.1) or GPE-2 (Condition 7.1.2).
   (e) κ_c, ξ: Simulate $\kappa_c \sim p(\cdot|R_c)$, and simulate $\xi_{c,1}, \ldots, \xi_{c,\kappa_c} \sim \mathcal{U}[t_{j-1}, t_j]$.
   (f) $\boldsymbol{z}^{(c)}$: Simulate $\boldsymbol{z}_{\xi_{c,1}}^{(c)}, \ldots, \boldsymbol{z}_{\xi_{c,\kappa_c}}^{(c)} \sim \mathbb{W}_{\mathbb{I}_d}|R_c^{(z)}$ as per Algorithm 4.2.5.
   (g) $\boldsymbol{X}^{(c)}$: Reverse transform the path, setting $\boldsymbol{X}_{\xi_{c,k_c}}^{(c)} = \boldsymbol{\Lambda}_c^{\frac{1}{2}} \boldsymbol{z}_{\xi_{c,k_c}}^{(c)}$ for $k_c = 1, \ldots, \kappa_c$.
2. Output $\tilde{\rho}_j^{(\cdot)}$ (7.20).

7.1.2.3 Generalised Bayesian Fusion algorithm

As noted earlier, Theorem 7.1.1 suggests that we can simulate from the fusion target density $f^{(\mathcal{C})}$ by simulating $\mathfrak{X} \sim \mathbb{F}$ and retaining the T time marginal, $\boldsymbol{y}^{(\mathcal{C})}$. As suggested by the theory, we do so by means of simulating a number of proposals $\mathfrak{X} \sim \mathbb{P}$ and accepting (or importance weighting) the terminal time marginal $\boldsymbol{y}^{(\mathcal{C})}$ with probability proportional to the Radon-Nikodým derivative in (7.2). We are now able to practically do each of these steps (as discussed in Sections 7.1.2.1 and 7.1.2.2 respectively), but the methodological approach requires careful consideration.

The simplest methodological approach here is a *rejection sampler* (see Section 2.2). To do so, we can simply simulate a proposal from $h_{\mathcal{C}}$ (7.17) (by utilising Theorem 7.1.2) and accept this proposal (and returning the value for $\boldsymbol{y}^{(\mathcal{C})}$ as a sample from $f^{(\mathcal{C})}$) with probability equal to $\prod_{j=0}^{n} \rho_j$. Whilst a rejection sampling approach to sample from $g_{\mathcal{C}}$ in this fashion is valid and would ultimately return i.i.d. draws from our fusion target $f^{(\mathcal{C})}$, this approach would unfortunately suffer from several inefficiencies. In fact, Monte Carlo Fusion Dai et al. [2019] (discussed in Section 5.1) follows a similar approach (although based upon a different formulation of the problem which does not utilise Theorem 7.1.2) and suffers from these computational drawbacks. Note a full discussion of the connections between the methodology outlined in this chapter and the earlier Fusion algorithms are discussed at the end of this section. In particular, we would expect the acceptance probabilities ρ_j (7.19) to decay geometrically with increasing number of sub-posteriors, $|\mathcal{C}|$, as each term in this product is bounded by 1. Furthermore, we expect the acceptance probability $\prod_{j=0}^{n} \rho_j$ to decay exponentially with increasing T. Consequently, a rejection sampling approach will ultimately be impractical and typically would lead to very small acceptance probabilities. Similarly, an importance sampling approach (in which the proposals are all retained with an importance weight of $\prod_{j=0}^{n} \rho_j$) will ultimately suffer from the same issues of robustness in practice.

The BF approach of Dai et al. [2021] introduced the auxiliary temporal partition $\mathcal{P}$ in order to simulate from $g_{\mathcal{C}}$ using SMC, allowing for the *gradual* coalescence of the C stochastic processes. In

particular, we can initialise an SMC algorithm by simulating N particles from the time 0 marginal in $h_{\mathcal{C}}$ (which consists of composing $|\mathcal{C}|$ samples from each of the sub-posterior densities to obtain $\vec{\boldsymbol{x}}_0^{(\mathcal{C})}$), and assigning them an initial un-normalised importance weight given by $w'_{0,i} := \rho_0(\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})})$ for $i = 1, \ldots, n$. This initial particle set constitutes an approximation to the time 0 marginal of $g_{\mathcal{C}}$, and can be sequentially propagated n times through the temporal time mesh $\mathcal{P}$ by simulating $\vec{\boldsymbol{x}}_{j,i}^{(\mathcal{C})} | \vec{\boldsymbol{x}}_{j-1,i}^{(\mathcal{C})} \sim \mathcal{N}_d(\vec{\boldsymbol{M}}_{j,i}^{(\mathcal{C})}, \boldsymbol{V}_j)$ as per (7.11) and (7.12). In our SMC formulation, at each iteration ($j = 1, \ldots, n$) the un-normalised importance weight of every particle is updated by a factor of $\rho_j(\vec{\boldsymbol{x}}_{j-1,i}^{(\mathcal{C})}, \vec{\boldsymbol{x}}_{j,i}^{(\mathcal{C})})$. The weighted particle set obtained after the final nth iteration of the algorithm corresponds to the coalesce time T, meaning that $\boldsymbol{y}_i^{(\mathcal{C})} := \boldsymbol{x}_{n,i}^{(c_1)} = \cdots = \boldsymbol{x}_{n,i}^{(c_{|\mathcal{C}|})}$ for $i = 1, \ldots, N$. By retaining the final time marginal $\boldsymbol{y}^{(\mathcal{C})}$ for each of the N weighted particles (i.e. $\{\boldsymbol{y}_i^{(\mathcal{C})}, w_{n,i}^{(\mathcal{C})}\}_{i=1}^N$), we have an approximation to the desired distribution,

$$f^{(\mathcal{C})}(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \approx \sum_{i=1}^{N} w_{n,i}^{(\mathcal{C})} \cdot \delta_{\boldsymbol{y}_i^{(\mathcal{C})}}(\mathrm{d}\boldsymbol{y}). \tag{7.25}$$

As is common in SMC, to avoid *weight degeneracy* in which the variance of the importance weights degrades rapidly in n, we employ a *resampling* strategy. As such, at the end of each iteration, the weights are *re-normalised* and we follow the conventional approach of Kong et al. [1994] to monitor weight degeneracy by estimating *effective sample size (ESS)* (3.18) of the particle set. If the ESS falls below some user-specified threshold then at the beginning of the next iteration we resample the particle set to get N equally weighted particles. In all of our simulations in the subsequent sections, we used residual resampling [Higuchi, 1997; Liu and Chen, 1998; Whitley, 1994], but note there are a wide variety of resampling methodologies within the SMC literature (see e.g. Gerber et al. [2019] for a recent investigation of the properties of many resampling schemes). As remarked upon in Section 7.1.2.2, due to this normalisation of the particle set weights we can avoid the need to explicitly compute the constants $\boldsymbol{\Phi}_c$ in (7.19), as they are simply constants which cancel.

We term our Fusion approach *Generalised Bayesian Fusion (GBF)* and summarise it in Algorithm 7.1.2. To generalise the algorithm further (and make it amenable to a recursive divide-and-conquer approach), we assume we have access to M importance weighted realisations of each sub-posterior, $\{\boldsymbol{x}_{0,k}^{(c)}, w_k^{(c)}\}_{k=1}^M$ for $c \in \mathcal{C}$. To initialise the algorithm, we start by composing M initial weighted particles by pairing the draws from each sub-posterior $\{\vec{\boldsymbol{x}}_{0,k}^{(\mathcal{C})}\}_{k=1}^M$, and compute the associated (un-normalised) partial weights $\{w_{0,k}^{(\mathcal{C})\prime}\}_{k=1}^M$ where $w_{0,k}^{(\mathcal{C})\prime} := \left(\prod_{c\in\mathcal{C}} w_k^{(c)}\right) \cdot \rho_0(\vec{\boldsymbol{x}}_{0,k}^{(\mathcal{C})})$ for $k = 1, \ldots, M$. If we have $M \neq N$, we resample to obtain N samples from each sub-posterior, otherwise, we choose to only resample if the ESS is below some user-specified threshold. In general, for the Input step of Algorithm 7.1.2, we may have access to different numbers of samples from each sub-posterior: say M_c importance weighted samples for sub-posterior f_c (for $c \in \mathcal{C}$). For our simulations, if $M_c = M$ for $c \in \mathcal{C}$, we simply pair the sub-posterior draws index-wise. This is a basic merging strategy of the sub-posterior realisations and has the advantage that it can be implemented in $O(M)$ cost (and if $M_c \neq M$ for every $c \in \mathcal{C}$ one simply sub-sample to obtain a common number of samples from

each sub-posterior). We found this simple approach to be adequate in our simulations, but note there are more sophisticated options available described in Lindsten et al. [2017, Section 4.1] (and see Section 6.1.2 for a more detailed discussion).

Algorithm 7.1.2 `gbf`$(\mathcal{C}, \{\{\boldsymbol{x}_{0,i}^{(c)}, w_i^{(c)}\}_{i=1}^M, \boldsymbol{\Lambda}_c\}_{c\in\mathcal{C}}, N, \mathcal{P})$: Generalised Bayesian Fusion (GBF).

---

1. **Initialisation** $(j = 0)$:
   (a) **Input:** Importance weighted realisations $\{\boldsymbol{x}_{0,i}^{(c)}, w_i^{(c)}\}_{i=1}^M$ for $c \in \mathcal{C} := (c_1, \ldots, c_{|\mathcal{C}|})$, the user-specified matrices, $\{\boldsymbol{\Lambda}_c : c \in \mathcal{C}\}$, the number of particles required, N, and temporal partition $\mathcal{P} := \{t_0, t_1, \ldots, t_n : 0 =: t_0 < t_1 < \cdots < t_n := T\}$.
   (b) Compose the importance weighted realisations $\{\vec{\boldsymbol{x}}_{0,k}^{(\mathcal{C})}, w_{0,k}^{(\mathcal{C})\prime}\}_{k=1}^M$ where $w_{0,k}^{(\mathcal{C})\prime} := \left(\prod_{c\in\mathcal{C}} w_k^{(c)}\right) \cdot \rho_0(\vec{\boldsymbol{x}}_{0,k}^{(\mathcal{C})})$ for $k = 1, \ldots, M$ as per (6.7).
   (c) $w_{0,k}^{(\mathcal{C})}$**:** For k in 1 to M, compute normalised weight $w_{0,k} = w_{0,k}^{(\mathcal{C})\prime} / \sum_{k'=1}^M w_{0,k'}^{(\mathcal{C})\prime}$.
   (d) g_0^M: Set $g_0^M(\mathrm{d}\vec{\boldsymbol{x}}_0^{(\mathcal{C})}) := \sum_{k=1}^M w_{0,k}^{(\mathcal{C})} \cdot \delta_{\vec{\boldsymbol{x}}_{0,k}^{(\mathcal{C})}}(\mathrm{d}\vec{\boldsymbol{x}}_0^{(\mathcal{C})})$.
   (e) $\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})}$**:** If $M \neq N$, for $i = 1, \ldots, N$, resample $\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})} \sim g_0^M$ and reset $w_{0,i}^{(\mathcal{C})} = \frac{1}{N}$.
2. **Iterative updates**. For $j = 1, \ldots, n$:
   (a) **Resample:** If the ESS $:= \left(\sum_{i=1}^N {w_{j-1,i}^{(\mathcal{C})}}^2\right)^{-1}$ breaches the lower user-specified threshold, then for $i = 1, \ldots, N$, resample $\vec{\boldsymbol{x}}_{j-1,i}^{(\mathcal{C})} \sim g_{j-1}^N$ and reset $w_{j-1,i}^{(\mathcal{C})} = \frac{1}{N}$.
   (b) For i in 1 to N,
      i. $\vec{\boldsymbol{x}}_{j,i}^{(\mathcal{C})}$**:** Simulate $\vec{\boldsymbol{x}}_{j,i}^{(\mathcal{C})} \sim \mathcal{N}_d\left(\vec{\boldsymbol{M}}_{j,i}^{(\mathcal{C})}, \boldsymbol{V}_j\right)$ as per Theorem 7.1.2.
      ii. $w_{j,i}^{(\mathcal{C})\prime}$**:** Compute un-normalised weight $w_{j,i}^{(\mathcal{C})\prime} = w_{j-1,i}^{(\mathcal{C})} \cdot \tilde{\rho}_j(\vec{\boldsymbol{x}}_{j-1,i}^{(\mathcal{C})}, \vec{\boldsymbol{x}}_{j,i}^{(\mathcal{C})})$ as per (7.20) (using Algorithm 7.1.1).
   (c) $w_{j,i}^{(\mathcal{C})}$**:** For i in 1 to N, compute normalised weight $w_{j,i}^{(\mathcal{C})} = w_{j,i}^{(\mathcal{C})\prime} / \sum_{k'=1}^N w_{j,k'}^{(\mathcal{C})\prime}$.
   (d) g_j^N: Set $g_j^N(\mathrm{d}\vec{\boldsymbol{x}}_j^{(\mathcal{C})}) := \sum_{i=1}^N w_{j,i}^{(\mathcal{C})} \cdot \delta_{\vec{\boldsymbol{x}}_{j,i}^{(\mathcal{C})}}(\mathrm{d}\vec{\boldsymbol{x}}_j^{(\mathcal{C})})$.
3. **Output:** $\left\{\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})}, \ldots, \vec{\boldsymbol{x}}_{n-1,i}^{(\mathcal{C})}, \boldsymbol{y}_i^{(\mathcal{C})}, w_{n,i}^{(\mathcal{C})}\right\}_{i=1}^N$, where $\hat{f}^{(\mathcal{C})}(\mathrm{d}\boldsymbol{y}) := g_n^N(\mathrm{d}\boldsymbol{y}) \approx f^{(\mathcal{C})}(\boldsymbol{y})\mathrm{d}\boldsymbol{y}$.

The theory and methodology developed in this section admits the Bayesian Fusion [Dai et al., 2021] and Generalised Monte Carlo Fusion (GMCF) (see Chapter 6) approaches as a special case and is established in the following corollaries:

Corollary 7.1.1. *Setting $\boldsymbol{\Lambda}_c = \mathbb{I}_d$ for $c \in \mathcal{C} := \{1, \ldots, C\}$, where $\mathbb{I}_d$ is the identity matrix of dimension d, and applying the approach outlined in Algorithm 7.1.2 recovers the Bayesian Fusion approach of Dai et al. [2021, Algorithm 1] (also given in Algorithm 5.2.2).*

Corollary 7.1.2. *Setting $\mathcal{P} := \{0, T\}$ and applying the approach outlined in Algorithm 7.1.2 recovers the Generalised Monte Carlo Fusion approach of Algorithm 6.1.2.*

As alluded to in Section 5.2, one of the main reasons why GBF is typically more robust than the GMCF is that there is more flexibility in the algorithm. We have seen that in the GMCF (and

MCF (see Section 5.1)) approaches, there is a trade-off when choosing T. In particular, recall with the MCF, if T is chosen to be small, then the first acceptance probability ρ^{bm} (5.8) is smaller, while the second acceptance probability Q^{bm} (5.9) is going to be larger. In the GMCF approach where an importance sampling approach is used, this corresponds to having a larger variance in the normalised importance weights from the initialisation of the algorithm (meaning a low ESS after the first step of the algorithm), but having smaller variance in the normalised weights in the second step of the algorithm (hence larger ESS). For larger values of T, then the opposite is true.

A key drawback with MCF and GMCF is that in many practical settings, it is unrealistic to be able to find the optimal tuning parameter T easily, and even if the optimal T is found, it is not likely that you will have good performance across the two steps of the algorithms. A particular case where this is a problem is when combining conflicting (heterogeneous) sub-posteriors. In such settings, it will be incredibly difficult to find a T which leads to good effective sample size at the end of the two importance sampling steps in Algorithm 6.1.2 (in a rejection sampling paradigm, we will have poor acceptance rates at the two accept/reject steps in Algorithm 5.1.3). In contrast, the BF and GBF approaches introduce a temporal partition of time T, denoted $\mathcal{P}$. This provides the methods with enough flexibility to ensure there is a good initialisation of the algorithm whilst still having stable performance during the iterative steps in the algorithm. In particular, it is possible to make T sufficiently large such that the initial importance weights $\{\rho_{0,i}\}_{i=1}^{N}$ (see Step 1b) have low variance. Furthermore, we can ensure we have stable importance weights during the iterative steps of the algorithm by imposing a finer temporal mesh for $\mathcal{P}$. Unfortunately, this comes at the cost of increasing the number of iterations in the algorithm, n, but the advantage is that we are able to obtain a more robust algorithm for the fusion problem.

7.2 Divide-and-Conquer Generalised Bayesian Fusion

In Section 6.2, we embedded the Generalised Monte Carlo Fusion (GMCF) approach within a Divide-and-Conquer Sequential Monte Carlo (D&C-SMC) framework [Lindsten et al., 2017] (see Section 3.4) and we found this to vastly improve the scalability of the GMCF approach with regards to increasing number of sub-posteriors to combine (see for instance Section 6.3.2). The key idea was that since the fundamental goal of the methodology is to approximate f in (1.1), it is possible to construct a recursive *divide-and-conquer* approach in which the sub-posteriors are combined in stages to recover f (as opposed to combining the sub-posterior samples in a single step, i.e. $\mathcal{C} := \{1, \ldots, C\}$). The notable advantage of appealing to the D&C-SMC framework is that we can consequently take advantage of the theory and methodologies which have been developed for this framework. As we noted in Section 3.4, the theoretical properties of D&C-SMC are increasingly-well characterised (see e.g. Kuntz et al. [2021b]). In this section, we follow the same approach as in Section 6.2 to incorporate the Generalised Bayesian Fusion (GBF) approach of Section 7.1 within a D&C-SMC algorithm. Since we have seen how we can do this with GMCF in Section 6.1, in this section, we will mostly re-define notation which is required to do this for GBF.

As noted in Section 6.2, we can represent the order by which sub-posteriors are combined by using *tree diagrams*, which are termed *hierarchies* (see e.g. Figure 6.2). In these hierarchies, the intermediate vertices represent intermediate (*auxiliary*) densities up to proportionality. The approximation of the distribution associated with any non-leaf vertex is obtained by an application of Fusion methodology to the densities of the children of that vertex. In Section 6.2, we focused on two alternative approaches to the fork-and-join method; namely the *balanced-binary tree* approach where we combine two sub-posteriors at a time (illustrated in Figure 6.2a), and the *progressive tree* approach whereby sub-posteriors are fused one at a time (illustrated in Figure 6.2b). While these are two simple tree hierarchies that can be applied, the methodology that we develop in this section is a general framework which can be used with any such tree.

We again adopt the notation from Section 6.2 to introduce our divide-and-conquer approach. Let $\mathbb{T} = (\mathcal{V}, \mathcal{E})$ denote a tree with vertices $\mathcal{V}$ and (directed) edge set $\mathcal{E}$. Let Leaf($\mathbb{T}$) denote the leaves of the tree (which represent the sub-posteriors that we wish to ultimately unify: $f_1, \ldots, f_c$), Root($\mathbb{T}$) denote the root of the tree (which represents the fusion target density f (1.1)) and Ch(v) denote the children of vertex $v \in \mathcal{V}$ where $\text{Ch}(t) = \emptyset$ if t is a leaf. Let $\mathcal{V} = \{v_0, v_1, \ldots, v_C, \ldots\}$ be the set of vertices, with $v_0 = \text{Root}(\mathbb{T})$, $\{v_1, \ldots, v_C\} = \text{Leaf}(\mathbb{T})$ and as many intermediate vertices as are required to specify the tree. Furthermore, to directly apply the methodology developed in Section 7.1.2 we define the following notation for non-leaf vertices $v \notin \text{Leaf}(\mathbb{T})$: let $\mathcal{C}_v := \cup_{u \in \text{Ch}(v)} \mathcal{C}_u$ denote the index set representing the sub-posteriors that we want to unify for vertex $v \notin \text{Leaf}(\mathbb{T})$. In addition, to simplify the notation and avoid an unnecessary level of subscripts, we index densities and other quantities by v rather than $\mathcal{C}_v$ when it is clear what is intended. In particular, let $\boldsymbol{\Lambda}_v := \boldsymbol{\Lambda}_{\mathcal{C}_v}$, $\vec{\boldsymbol{x}}_t^{(v)} := \vec{\boldsymbol{x}}_t^{(\mathcal{C}_v)}$, $\tilde{\boldsymbol{x}}_t^{(v)} := \tilde{\boldsymbol{x}}_t^{(\mathcal{C}_v)}$, $\boldsymbol{y}^{(v)} := \boldsymbol{y}^{(\mathcal{C}_v)}$ where $\boldsymbol{y}^{(v)} \sim f_v := f^{(\mathcal{C}_v)}$ for $v \notin \text{Leaf}(\mathbb{T})$. Let $\mathbb{W}_{\boldsymbol{\Lambda}_v, j}$ denote the law of a Brownian bridge $\{\boldsymbol{X}_t^{(v)}, t \in [t_{j-1}, t_j]\}$ with $\boldsymbol{X}_{t_{j-1}}^{(v)} := \boldsymbol{x}_{j-1}^{(v)}$ and $\boldsymbol{X}_{t_j}^{(v)} := \boldsymbol{x}_j^{(v)}$ with covariance $\boldsymbol{\Lambda}_v$ for $j = 1, \ldots, n$. The extended target and proposal densities for vertex $v \notin \text{Leaf}(\mathbb{T})$ are denoted $g_v := g_{\mathcal{C}_v}$ and $h_v := h_{\mathcal{C}_v}$, respectively. Lastly, the importance sampling weights for $v \notin \text{Leaf}(\mathbb{T})$ are given by $\rho_0^{(v)}(\vec{\boldsymbol{x}}^{(v)}) := \rho_0(\vec{\boldsymbol{x}}^{(\mathcal{C}_v)})$ and $\rho_j^{(v)}(\vec{\boldsymbol{x}}^{(v)}, \boldsymbol{y}^{(v)}) := \rho_j(\vec{\boldsymbol{x}}^{(\mathcal{C}_v)}, \boldsymbol{y}^{(\mathcal{C}_v)})$ for all j.

From this recursive perspective, sample approximations of auxiliary densities obtained at one level of any tree are themselves treated as sub-posteriors at the next level up. As such, one can iteratively apply GBF and work through the levels of the tree from the leaves to the root, using at each stage the output of one step as the input for the subsequent step. An advantage of our divide-and-conquer approach is that as fewer sub-posteriors are combined at each stage, we avoid the rapidly diminishing and variable importance weights as noted in Section 6.2.

As in Section 6.2, we describe the Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF) approach by specify an algorithm that is carried out at each vertex $v \in \mathcal{V}$ which leads to a recursive procedure; an initial call to `d&c.gbf`(Root($\mathcal{V}$), ...) carries out the overall approach. For $v \in \mathcal{V}$, we define a procedure (as given in Algorithm 7.2.1), which returns a weighted particle set $\{\vec{\boldsymbol{x}}_{0,i}^{(v)}, \ldots, \vec{\boldsymbol{x}}_{n-1,i}^{(v)}, \boldsymbol{y}_i^{(v)}, w_{n,i}^{(v)}\}_{i=1}^N$ where $w_{n,i}^{(v)}$ denotes the normalised importance weight of particle i

for vertex $v \in \mathcal{V}$. From this particle set, we can take the marginal weighted samples for $\boldsymbol{y}^{(v)}$ to approximate the fusion density $f_v \propto \prod_{u \in \mathrm{Ch}(v)} f_u$ for vertex $v \in \mathcal{V}$. Recall that the leaf vertices, v_c for $c = 1, \ldots, C$, represent each of the sub-posteriors. It is straightforwardly possible to additionally incorporate importance sampling for the leaf vertices but for simplicity we assume that we have access to unweighted samples for the sub-posteriors. Therefore, at these leaf vertices, we simply sample from the sub-posteriors. If v is a non-leaf vertex, we simply call Algorithm 7.1.2 by inputting the importance weighted samples $\{\boldsymbol{y}_i^{(u)}, w_i^{(u)}\}_{i=1}^N$ for $u \in \mathrm{Ch}(v)$. As we did in Section 6.2, although the auxiliary distributions are defined on larger spaces, we do not need to retain sampled values which are not subsequently used; to better manage the memory required to implement our method, we can choose to retain only returning the weighted particle set $\{\boldsymbol{y}_i^{(v)}, w_i^{(v)}\}_{i=1}^N$ since we only require this to compute the importance weights in Algorithm 7.1.2 at each vertex $v \notin \mathrm{Leaf}(\mathbb{T})$.

Algorithm 7.2.1 `d&c.gbf`$(v, N, \mathcal{P})$: Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF).

Given: Sub-posteriors, $\{f_u\}_{u \in \mathrm{Leaf}(\mathbb{T})}$, and preconditioning matrices $\{\boldsymbol{\Lambda}_u\}_{u \in \mathbb{T}}$.
Input: Node in tree, v, the number of particles N, and (optionally) the temporal mesh partitions $\{\mathcal{P}_u\}_{u \in \mathrm{Ch}(v)}$, $\mathcal{P}_v$.

1. For $u \in \mathrm{Ch}(v)$,
   (a) $\left\{\vec{\boldsymbol{x}}_{0,i}^{(u)}, \ldots, \vec{\boldsymbol{x}}_{n-1,i}^{(u)}, \boldsymbol{y}_i^{(u)}, w_{n,i}^{(u)}\right\}_{i=1}^N \leftarrow$ `d&c.gbf`$(u, N, \mathcal{P}_u)$.
2. If $v \in \mathrm{Leaf}(\mathbb{T})$,
   (a) For $i = 1, \ldots, N$, sample $\boldsymbol{y}_i^{(v)} \sim f_v(\boldsymbol{y})$.
   (b) **Output:** $\{\emptyset, \boldsymbol{y}_i^{(v)}, \frac{1}{N}\}_{i=1}^N$.
3. If $v \notin \mathrm{Leaf}(\mathbb{T})$,
   (a) If $\mathcal{P}_v$ is not inputted, apply guidance from Section 7.3.1 and Section 7.3.2.
   (b) **Output:** Call `gbf`$(\mathrm{Ch}(v), \{\{\boldsymbol{y}_i^{(u)}, w_i^{(u)}\}_{i=1}^N, \boldsymbol{\Lambda}_u\}_{u \in \mathrm{Ch}(v)}, N, \mathcal{P}_v)$.

Note that in Algorithm 7.2.1, we allow the user to specify different temporal partitions at each node and level (i.e. $\{\mathcal{P}_u\}_{u \in \mathrm{Ch}(v)}$, $\mathcal{P}_v$). As we explore fully in Section 7.3, when we develop guidance for user chosen tuning parameters, having this flexibility on the temporal partition can lead to a far more robust and efficient implementation of Algorithm 7.2.1.

7.3 Implementational guidance for Generalised Bayesian Fusion

In this section, we develop guidance for choosing the parameter T and the temporal partition $\mathcal{P}$ (and so n implicitly) for our Generalised Bayesian Fusion (GBF) approach (Algorithm 7.1.2). Note that this guidance can be used directly *at each node* within our Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF) approach (Algorithm 7.2.1). Since GBF is fundamentally a sequential Monte Carlo (SMC) algorithm, we want to choose these hyperparameters to ensure that the discrepancy between subsequent proposals and targets are not degenerate. For this reason, and in common with Dai et al. [2021, Section 3], we look at the incremental weight changes and study the

conditional effective sample size (CESS) [Zhou et al., 2016]:

$$\mathrm{CESS}_j := \frac{\left(\sum_{i=1}^N \tilde{\rho}_{j,i}\right)^2}{\sum_{i=1}^N \tilde{\rho}_{j,i}^2} \text{ for } j = 1, \ldots, n; \qquad \mathrm{CESS}_0 := \frac{\left(\sum_{i=1}^N \rho_{0,i}\right)^2}{\sum_{i=1}^N \rho_{0,i}^2}, \tag{7.26}$$

where $\rho_{0,i}$ and $\tilde{\rho}_{j,i}$ are given in (6.7) and (7.20) respectively.

In order to develop heuristics to choose hyperparameters, we consider the idealised setting of combining multivariate Gaussian sub-posteriors with mean vector $\boldsymbol{a}_c$ and covariance matrix $\frac{b|\mathcal{C}|}{m}\boldsymbol{\Lambda}_c$, for some $b > 0$, for $c \in \mathcal{C}$. The target is $f \sim \mathcal{N}_d(\tilde{\boldsymbol{a}}, \frac{b|\mathcal{C}|}{m}\boldsymbol{\Lambda}_{\mathcal{C}})$, where $\tilde{\boldsymbol{a}} := \left(\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}\right)^{-1}\left(\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}\boldsymbol{a}_c\right)$ and $\boldsymbol{\Lambda}_{\mathcal{C}} := \left(\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}\right)^{-1}$. For BF, Dai et al. [2021] considered this setting along with $\boldsymbol{\Lambda}_c = \mathbb{I}_d$ for all $c \in \mathcal{C}$. By imposing an additional assumption that the partition was a *regular* mesh, we obtain a temporal mesh $\mathcal{P}$. In this section, we instead develop guidance for T (see Section 7.3.1) in the more sophisticated GBF setting, and then in Section 7.3.2 investigate the more challenging selection of $\mathcal{P}$ without assumption on its regularity (i.e. permitting an *irregular*, or *adaptive*, choice of mesh). As a consequence we instead implicitly find n. It should be noted that the findings in our work can be directly applied in the BF setting to improve upon that methodology.

In our idealised setting the key consideration is the degree to which the sub-posteriors disagree with one another. To measure how significantly the *sub-posterior conflict*, we define

$$\sigma_{\boldsymbol{a}}^2 := \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} (\boldsymbol{a}_c - \tilde{\boldsymbol{a}})^\intercal \boldsymbol{\Lambda}_c^{-1}(\boldsymbol{a}_c - \tilde{\boldsymbol{a}}). \tag{7.27}$$

We further consider the two following conditions in order to explore how the algorithm hyperparameters should change according to sub-posterior heterogeneity:

Condition 7.3.1. *SH(λ). The sub-posteriors obey the SH(λ) condition (for some $\lambda > 0$) if*

$$\sigma_{\boldsymbol{a}}^2 = \frac{b(|\mathcal{C}| - 1)\lambda}{m}. \tag{7.28}$$

This represents a natural condition which arises in many settings, for instance if $\frac{m}{|\mathcal{C}|}$ of the data is randomly allocated to each sub-posteriors then $\sigma_{\boldsymbol{a}}^2 \sim \frac{b}{m}\chi^2_{|\mathcal{C}|-1}$ and have mean $\frac{b(|\mathcal{C}|-1)}{m}$. For $\frac{m}{|\mathcal{C}|}$ large, then we expect for $\lambda > 1$, the sub-posteriors would obey the SH(λ) condition with high probability.

Condition 7.3.2. *SSH(γ). The sub-posteriors obey the SSH(γ) condition (for some $\gamma > 0$) if*

$$\sigma_{\boldsymbol{a}}^2 = b\gamma. \tag{7.29}$$

This represents a setting where the sub-posterior heterogeneity does not decay with data size m.

Remark 7.3.1. Choice of (b): *In the case that* $\{\boldsymbol{\Lambda}_c\}_{c\in\mathcal{C}}$ *are chosen to be the estimated covariance matrices for each sub-posterior, then we set* $b = \frac{m}{|\mathcal{C}|}$*, since the sub-posteriors in our idealised setting,* $f_c \sim \mathcal{N}_d(\boldsymbol{a}_c, \frac{b|\mathcal{C}|}{m}\boldsymbol{\Lambda}_c)$*, will have variance which closely matches the sub-posterior variance. In general, we want to choose* b *such that* $\frac{b|\mathcal{C}|}{m}\boldsymbol{\Lambda}_c$ *is close to the variance of sub-posterior* f_c *for* $c \in \mathcal{C}$*.*

We study empirically our choices of tuning parameter in the idealised settings described by the SH(λ) condition and SSH(γ) condition in Sections 7.4.1–7.4.2 respectively.

7.3.1 Guidance for choosing T

The time horizon T only directly affects the initial weighting given to each of the N particles through ρ_0 in (6.7). Thus, to develop guidance for selecting T, we study CESS_0 in (7.26).

Theorem 7.3.1. *Let* $f_c \sim \mathcal{N}_d(\boldsymbol{a}_c, \frac{b|\mathcal{C}|}{m}\boldsymbol{\Lambda}_c)$ *for* $c \in \mathcal{C}$*, then considering the initial conditional effective sample size* $CESS_0$ *we have that as* $N \to \infty$*, the following convergence in probability holds*

$$N^{-1}CESS_0 \xrightarrow{p} \exp\left\{-\frac{\sigma^2_{\boldsymbol{a}}\left(\frac{b}{m}\right)}{\left(\frac{T}{|\mathcal{C}|}+\frac{b}{m}\right)\left(\frac{T}{|\mathcal{C}|}+\frac{2b}{m}\right)}\right\}\cdot\left[1+\frac{\left(\frac{|\mathcal{C}|b}{Tm}\right)^2}{1+\frac{2|\mathcal{C}|b}{Tm}}\right]^{-\frac{(|\mathcal{C}|-1)d}{2}}. \tag{7.30}$$

Proof. Considering the initial conditional effective sample size, CESS_0, we have

$$\begin{aligned}
N^{-1}\text{CESS}_0 := N^{-1}\left[\frac{\left(\sum_{i=1}^N \rho_{0,i}\right)^2}{\sum_{i=1}^N \rho_{0,i}^2}\right] &\to \frac{(\mathbb{E}[\rho_{0,i}])^2}{\mathbb{E}[\rho_{0,i}^2]} \\
&= \frac{\mathbb{E}\left[\exp\left\{-\sum_{c\in\mathcal{C}}\frac{(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}-\boldsymbol{x}_0^{(c)})^\intercal\boldsymbol{\Lambda}_c^{-1}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}-\boldsymbol{x}_0^{(c)})}{2T}\right\}\right]^2}{\mathbb{E}\left[\exp\left\{-\sum_{c\in\mathcal{C}}\frac{(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}-\boldsymbol{x}_0^{(c)})^\intercal\boldsymbol{\Lambda}_c^{-1}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}-\boldsymbol{x}_0^{(c)})}{T}\right\}\right]} \\
&= \frac{\mathbb{E}\left[e^{-\frac{|\mathcal{C}|\sigma^2}{2T}}\right]^2}{\mathbb{E}\left[e^{-\frac{|\mathcal{C}|\sigma^2}{T}}\right]},
\end{aligned} \tag{7.31}$$

where $\sigma^2 := \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}-\boldsymbol{x}_0^{(c)})^\intercal\boldsymbol{\Lambda}_c^{-1}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}-\boldsymbol{x}_0^{(c)})$ where $\boldsymbol{x}_0^{(c)} \sim \mathcal{N}_d(\boldsymbol{a}_c, \frac{b|\mathcal{C}|}{m}\boldsymbol{\Lambda}_c)$. To get an expression for $N^{-1}\text{CESS}_0$, we begin by obtaining the moment generating function (mgf) for σ^2. First note

$$\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}(\boldsymbol{x}_0^{(c)}-\tilde{\boldsymbol{a}})^\intercal\boldsymbol{\Lambda}_c^{-1}(\boldsymbol{x}_0^{(c)}-\tilde{\boldsymbol{a}}) = \sigma^2 + \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}-\tilde{\boldsymbol{a}})^\intercal\boldsymbol{\Lambda}_c^{-1}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}-\tilde{\boldsymbol{a}}). \tag{7.32}$$

Considering the term $\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}(\boldsymbol{x}_0^{(c)}-\tilde{\boldsymbol{a}})^\intercal\boldsymbol{\Lambda}_c^{-1}(\boldsymbol{x}_0^{(c)}-\tilde{\boldsymbol{a}})$ and letting $\boldsymbol{Y}_c := \boldsymbol{\Lambda}_c^{-\frac{1}{2}}(\boldsymbol{x}_0^{(c)}-\tilde{\boldsymbol{a}})$, then $\boldsymbol{Y}_c$ has

mean $\boldsymbol{\Lambda}_c^{-\frac{1}{2}}(\boldsymbol{a}_c - \tilde{\boldsymbol{a}})$ and variance $\frac{b|\mathcal{C}|}{m}\mathbb{I}_d$. Hence $\sqrt{\frac{m}{b|\mathcal{C}|}}\boldsymbol{Y}_c$ has mean $\sqrt{\frac{m}{b|\mathcal{C}|}}\boldsymbol{\Lambda}_c^{-\frac{1}{2}}(\boldsymbol{a}_c - \tilde{\boldsymbol{a}})$ and variance $\mathbb{I}_d$, and so let

$$\lambda = \sum_{c\in\mathcal{C}} \left\| \sqrt{\frac{m}{b|\mathcal{C}|}}\boldsymbol{\Lambda}_c^{-\frac{1}{2}}(\boldsymbol{a}_c - \tilde{\boldsymbol{a}}) \right\|^2 = \frac{m}{b|\mathcal{C}|}\sum_{c\in\mathcal{C}} (\boldsymbol{a}_c - \tilde{\boldsymbol{a}})^\intercal \boldsymbol{\Lambda}_c^{-1}(\boldsymbol{a}_c - \tilde{\boldsymbol{a}}) = \frac{m}{b}\sigma_{\boldsymbol{a}}^2,$$

then $\frac{m}{b|\mathcal{C}|}\sum_{c\in\mathcal{C}} \|\boldsymbol{Y}_c\|^2 \sim \chi^2(|\mathcal{C}|d, \lambda)$ distribution (i.e. $\frac{m}{b|\mathcal{C}|}\sum_{c\in\mathcal{C}} (\boldsymbol{x}_0^{(c)} - \tilde{\boldsymbol{a}})^\intercal \boldsymbol{\Lambda}_c^{-1}(\boldsymbol{x}_0^{(c)} - \tilde{\boldsymbol{a}})$ has a non-central $\chi^2(|\mathcal{C}|d, \lambda)$ distribution) with mgf

$$M_1(s) := \frac{\exp\left\{\frac{\lambda s}{1-2s}\right\}}{(1-2s)^{\frac{|\mathcal{C}|d}{2}}}. \tag{7.33}$$

Secondly, consider $\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} (\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \tilde{\boldsymbol{a}})^\intercal \boldsymbol{\Lambda}_c^{-1}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \tilde{\boldsymbol{a}}) = \frac{1}{|\mathcal{C}|}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \tilde{\boldsymbol{a}})^\intercal \boldsymbol{\Lambda}_{\mathcal{C}}^{-1}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \tilde{\boldsymbol{a}})$, where $\boldsymbol{\Lambda}_{\mathcal{C}}^{-1} := \sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1}$. Then since $\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} \sim \mathcal{N}_d(\tilde{\boldsymbol{a}}, \frac{b|\mathcal{C}|}{m}\boldsymbol{\Lambda}_{\mathcal{C}})$, then $\boldsymbol{Z} := \sqrt{\frac{m}{b|\mathcal{C}|}}\boldsymbol{\Lambda}^{-\frac{1}{2}}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \tilde{\boldsymbol{a}}) \sim \mathcal{N}_d(\boldsymbol{0}, \mathbb{I}_d)$ and so $\|\boldsymbol{Z}\|^2 \sim \chi_d^2$ (i.e. $\frac{m}{b|\mathcal{C}|}\sum_{c\in\mathcal{C}} (\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \tilde{\boldsymbol{a}})^\intercal \boldsymbol{\Lambda}_c^{-1}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \tilde{\boldsymbol{a}})$ has χ_d^2 distribution) with mgf

$$M_2(s) := (1-2s)^{-\frac{d}{2}}. \tag{7.34}$$

From (7.32), we have

$$\sigma^2 = \frac{b}{m}\underbrace{\left[\frac{m}{b|\mathcal{C}|}\sum_{c\in\mathcal{C}} (\boldsymbol{x}_0^{(c)} - \tilde{\boldsymbol{a}})^\intercal \boldsymbol{\Lambda}_c^{-1}(\boldsymbol{x}_0^{(c)} - \tilde{\boldsymbol{a}})\right]}_{\sim\chi^2(|\mathcal{C}|d,\lambda)} - \frac{b}{m}\underbrace{\left[\frac{m}{b|\mathcal{C}|}\sum_{c\in\mathcal{C}} (\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \tilde{\boldsymbol{a}})^\intercal \boldsymbol{\Lambda}_c^{-1}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})} - \tilde{\boldsymbol{a}})\right]}_{\sim\chi_d^2},$$

where $\lambda = \frac{m\sigma_{\boldsymbol{a}}^2}{b}$. Therefore, using (7.33) and (7.34), the mgf for σ^2 is given by

$$M_{\sigma^2}(s) = \frac{M_1(\frac{sb}{m})}{M_2(\frac{sb}{m})} = \exp\left\{\frac{m\sigma_{\boldsymbol{a}}^2 s}{m-2sb}\right\} \cdot \left(1 - 2\frac{sb}{m}\right)^{-\frac{(|\mathcal{C}|-1)d}{2}}, \text{ where } \frac{sb}{m} < \frac{1}{2}. \tag{7.35}$$

Given the mgf of σ^2, then

$$\begin{aligned} N^{-1}\text{CESS}_0 &\to \frac{\mathbb{E}\left[e^{-\frac{|\mathcal{C}|\sigma^2}{2T}}\right]^2}{\mathbb{E}\left[e^{-\frac{|\mathcal{C}|\sigma^2}{T}}\right]} = \frac{M_{\sigma^2}\left(-\frac{|\mathcal{C}|}{2T}\right)^2}{M_{\sigma^2}\left(-\frac{|\mathcal{C}|}{T}\right)} \\ &= \frac{\left[\exp\left\{\frac{m\sigma_{\boldsymbol{a}}^2\left(-\frac{|\mathcal{C}|}{2T}\right)}{m-2\left(-\frac{|\mathcal{C}|b}{2T}\right)}\right\} \cdot \left(1 - 2\left(-\frac{|\mathcal{C}|}{2T}\right)\frac{b}{m}\right)^{-\frac{(|\mathcal{C}|-1)d}{2}}\right]^2}{\exp\left\{\frac{m\sigma_{\boldsymbol{a}}^2\left(-\frac{|\mathcal{C}|}{T}\right)}{m-2\left(-\frac{|\mathcal{C}|b}{T}\right)}\right\} \cdot \left(1 - 2\left(-\frac{|\mathcal{C}|}{T}\right)\frac{b}{m}\right)^{-\frac{(|\mathcal{C}|-1)d}{2}}} \end{aligned}$$

$$
\begin{aligned}
&= \frac{\exp\left\{-\frac{m\sigma_{\boldsymbol{a}}^2\left(\frac{|\mathcal{C}|}{T}\right)}{m+\frac{|\mathcal{C}|b}{T}}\right\}\cdot\left(1+\frac{|\mathcal{C}|b}{Tm}\right)^{-(|\mathcal{C}|-1)d}}{\exp\left\{-\frac{m\sigma_{\boldsymbol{a}}^2\left(\frac{|\mathcal{C}|}{T}\right)}{m+2\left(\frac{|\mathcal{C}|b}{T}\right)}\right\}\cdot\left(1+2\left(\frac{|\mathcal{C}|b}{Tm}\right)\right)^{-\frac{(|\mathcal{C}|-1)d}{2}}} \\
&= \exp\left\{-\frac{\sigma_{\boldsymbol{a}}^2}{\frac{T}{|\mathcal{C}|}+\frac{b}{m}}\right\}\cdot\exp\left\{\frac{\sigma_{\boldsymbol{a}}^2}{\frac{T}{|\mathcal{C}|}+\frac{2b}{m}}\right\}\cdot\left[\frac{\left(1+\frac{|\mathcal{C}|b}{Tm}\right)^2}{1+2\left(\frac{|\mathcal{C}|b}{Tm}\right)}\right]^{-\frac{(|\mathcal{C}|-1)d}{2}} \\
&= \exp\left\{-\frac{\sigma_{\boldsymbol{a}}^2\left(\frac{b}{m}\right)}{\left(\frac{T}{|\mathcal{C}|}+\frac{b}{m}\right)\left(\frac{T}{|\mathcal{C}|}+\frac{2b}{m}\right)}\right\}\cdot\left[1+\frac{\left(\frac{|\mathcal{C}|b}{Tm}\right)^2}{1+\frac{2|\mathcal{C}|b}{Tm}}\right]^{-\frac{(|\mathcal{C}|-1)d}{2}},
\end{aligned}
$$

and so Theorem 7.3.1 immediately follows. ■

The following corollary considers the effect of T on CESS_0 in the SH(λ) and SSH(γ) settings:

Corollary 7.3.1. *If for some constant $k_1 > 0$, T is chosen such that $T \geq \frac{b|\mathcal{C}|^{3/2}k_1}{m}$, then the following lower bounds on $CESS_0$ hold:*

(a) If SH(λ) holds for some $\lambda > 0$, then

$$\lim_{N\to\infty} N^{-1}CESS_0 \geq \exp\left\{-\frac{\lambda}{k_1^2}-\frac{d}{2k_1^2}\right\}. \tag{7.36}$$

(b) If SSH(γ) holds for some $\gamma > 0$, and $T \geq k_2|\mathcal{C}|^{\frac{1}{2}}$ for some constant $k_2 > 0$, then

$$\lim_{N\to\infty} N^{-1}CESS_0 \geq \exp\left\{-\frac{b\gamma}{k_1k_2}-\frac{d}{2k_1^2}\right\}. \tag{7.37}$$

Proof. Under Condition 7.3.1, $\sigma_{\boldsymbol{a}}^2 = \frac{(|\mathcal{C}|-1)\lambda}{m} < \frac{|\mathcal{C}|\lambda}{m}$, so for the first term in (7.30),

$$
\begin{aligned}
\exp\left\{-\frac{\sigma_{\boldsymbol{a}}^2\left(\frac{b}{m}\right)}{\left(\frac{T}{|\mathcal{C}|}+\frac{b}{m}\right)\left(\frac{T}{|\mathcal{C}|}+\frac{2b}{m}\right)}\right\} &\geq \exp\left\{-\frac{\sigma_{\boldsymbol{a}}^2 b|\mathcal{C}|^2}{T^2m}\right\} \\
&\geq \exp\left\{-\frac{b^2|\mathcal{C}|^3\lambda}{T^2m^2}\right\} \\
&\geq \exp\left\{-\frac{\lambda}{k_1^2}\right\},
\end{aligned} \tag{7.38}
$$

where $T \geq \frac{b|\mathcal{C}|^{3/2}k_1}{m}$ for some constant $k_1 > 0$, and for the second term in (7.30), then

$$\begin{aligned}
\left[1 + \frac{\left(\frac{|\mathcal{C}|b}{Tm}\right)^2}{1 + \frac{2|\mathcal{C}|b}{Tm}}\right]^{-\frac{(|\mathcal{C}|-1)d}{2}} &\geq \left[\exp\left\{\frac{\left(\frac{|\mathcal{C}|b}{Tm}\right)^2}{1 + \frac{2|\mathcal{C}|b}{Tm}}\right\}\right]^{-\frac{(|\mathcal{C}|-1)d}{2}} \\
&= \exp\left\{-\frac{\left(\frac{|\mathcal{C}|b}{Tm}\right)^2 (|\mathcal{C}|-1)d}{2(1 + \frac{2C}{Tm})}\right\} \\
&\geq \exp\left\{-\frac{\left(\frac{|\mathcal{C}|^3 b^2}{T^2 m^2}\right) d}{2}\right\} \\
&\geq \exp\left\{-\frac{d}{2k_1^2}\right\}, \qquad (7.39)
\end{aligned}$$

with $T \geq \frac{b|\mathcal{C}|^{3/2}k_1}{m}$. Hence, under Condition 7.3.1 and choosing $T \geq \frac{b|\mathcal{C}|^{3/2}k_1}{m}$, combining the bounds from (7.38) and (7.39) gives (7.36). Under Condition 7.3.2, $\sigma_{\boldsymbol{a}}^2 = b\gamma$, if we assume $T \geq \frac{b|\mathcal{C}|^{3/2}k_1}{m}$ for some constant $k_1 > 0$, and $T \geq |\mathcal{C}|^{\frac{1}{2}}k_2$ for some constant $k_2 > 0$, then

$$\left(\frac{T}{|\mathcal{C}|} + \frac{b}{m}\right)\left(\frac{T}{|\mathcal{C}|} + \frac{2b}{m}\right) \geq \frac{T^2}{|\mathcal{C}|^2} \geq \frac{bk_1k_2}{m},$$

and so we have

$$\exp\left\{-\frac{\sigma_{\boldsymbol{a}}^2\left(\frac{b}{m}\right)}{\left(\frac{T}{|\mathcal{C}|} + \frac{b}{m}\right)\left(\frac{T}{|\mathcal{C}|} + \frac{2b}{m}\right)}\right\} \geq \exp\left\{-\frac{\frac{b^2\gamma}{m}}{\frac{bk_1k_2}{m}}\right\} = \exp\left\{-\frac{b\gamma}{k_1k_2}\right\}. \qquad (7.40)$$

Hence, under Condition 7.3.2 and choosing T such that $T \geq \frac{b|\mathcal{C}|^{3/2}k_1}{m}$ and $T \geq |\mathcal{C}|^{\frac{1}{2}}k_2$, we can combine the bounds from (7.40) and (7.39) to obtain the bound in (7.37). ■

To provide a systematic way to choose k_1 and k_2 (and so determine T), we must estimate several quantities: b, a positive constant which ensures $\frac{b|\mathcal{C}|}{m}\boldsymbol{\Lambda}_c$ closely matches the variance of the sub-posterior; λ, related to the variance of the sub-posterior means; and $\sigma_{\boldsymbol{a}}^2$, which is defined in (7.27) and is the variance of the sub-posterior means. After obtaining these quantities, we can use Remark 7.3.2 in order to choose k_1 and k_2.

Remark 7.3.2. Choice of (k_1, k_2)*: To choose* k_1 *and* k_2*, we first specify* $\zeta \in (0, 1)$ *to be a lower bound on the initial effective sample size that we would desire. We then can consider which situation that we are likely to be in, and then:*

1. *Under SH*(λ)*, suppose we want to ensure* $N^{-1}CESS_0$ *is above* $\zeta \in (0, 1)$*, from* (7.36)*, we have* $\exp\left\{-\frac{\lambda}{k_1^2} - \frac{d}{2k_1^2}\right\} = \zeta$*, which implies we choose* $k_1 = \sqrt{-\frac{(\lambda + \frac{d}{2})}{\log(\zeta)}}$.

2. *Under SSH(γ), suppose we want to ensure $N^{-1}CESS_0$ is above $\zeta \in (0,1)$, then from* (7.37)*, we have*

$$\exp\left\{-\frac{b\gamma}{k_1 k_2} - \frac{d}{2k_1^2}\right\} = \zeta. \tag{7.41}$$

Recall that for SSH(γ), we must have $T \geq \max\left\{\frac{b|\mathcal{C}|^{3/2}k_1}{m}, |\mathcal{C}|^{\frac{1}{2}}k_2\right\}$. Since we wish T to be small, we would like k_1 and k_2 to be small, and thus we set these two terms equal to each other and find $k_2 = \frac{b|\mathcal{C}|k_1}{m}$. Substituting into (7.41)*, we then choose $k_1 = \sqrt{-\frac{\left(\frac{\gamma m}{C} + \frac{d}{2}\right)}{\log(\zeta)}}$.*

Given k_1 and k_2, T can be chosen such that $T \geq \frac{b|\mathcal{C}|^{3/2}k_1}{m}$ if SH(λ) holds, and $T \geq \max\left\{\frac{b|\mathcal{C}|^{3/2}k_1}{m}, |\mathcal{C}|^{\frac{1}{2}}k_2\right\}$ if SSH(γ) holds. Typically we want to minimise the number of iterations in Algorithm 7.1.2, so we choose the smallest T which satisfies the user-specified $\zeta \in (0,1)$.

7.3.2 Guidance for choosing $\mathcal{P}$

In this section, we develop guidance for choosing the temporal mesh $\mathcal{P}$ by considering the CESS for the jth iteration of the algorithm (i.e. in the time interval $(t_{j-1}, t_j]$), then iterating for $j = 1, \ldots, n$.

To simplify the analysis of Algorithm 7.1.2, for which there is considerable flexibility in the choice of proposal distribution for our unbiased estimator of the importance weights (see Theorem 7.1.3 of Section 7.1.2.2), we assume that we have access to the *optimal* unbiased estimator. Recall from Section 4.4.3 that Fearnhead et al. [2008, Theorem 1] (and Dai et al. [2021, Appendix B]) showed that the variance of the unbiased estimator $a_j\tilde{\rho}_j$ is minimised when $p(\kappa_c|R_c) \sim \text{Poi}(\lambda_c)$, where

$$\lambda_c := \left[\Delta_j \int_{t_{j-1}}^{t_j} \left(U_j^{(c)} - \phi_c\left(\boldsymbol{X}_t^{(c)}\right)\right)^2 \mathrm{d}t\right]^{\frac{1}{2}}, \tag{7.42}$$

for $c \in \mathcal{C}$. Under the optimal choice, the second moment is finite and $\mathbb{E}\left[\left(a_j\tilde{\rho}_j\right)^2\right] \leq 1 < \infty$. In practice choosing this optimal distribution for $\mathbb{K}$ is not possible since the integral in (7.42) cannot be evaluated directly. This is why in Section 7.1.2.2 we choose alternative simulatable distributions (as described in Conditions 7.1.1–7.1.2), which try to match this optimal distribution closely. With this optimal choice, we establish the following theorem:

Theorem 7.3.2. *Let $p(\kappa_c|R_c)$ in* (7.20) *be a Poisson distribution with intensity given in* (7.42)*, for $c \in \mathcal{C}$, and k_3, k_4 be positive constants. If $\lim_{\Delta_j \to 0}$ is taken over sequences of $\Delta_j := t_j - t_{j-1} \to 0$ with*

$$t_j - t_{j-1} \leq \tilde{\Delta}_j := \min\left\{\frac{b^2|\mathcal{C}|k_3}{\mathbb{E}\left[\nu_j\right]m^2}, \left(\frac{b^2|\mathcal{C}|k_4}{2m^2 d}\right)^{\frac{1}{2}}\right\}, \tag{7.43}$$

where

$$\nu_j := \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}\left(\boldsymbol{x}_{j-1}^{(c)} - \boldsymbol{a}_c\right)^{\intercal}\boldsymbol{\Lambda}_c^{-1}\left(\boldsymbol{x}_{j-1}^{(c)} - \boldsymbol{a}_c\right), \tag{7.44}$$

and the expectation $\mathbb{E}[\nu_j]$ *is taken over* $\vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})}$*, we have*

$$\lim_{\Delta_j \to 0} \lim_{N \to \infty} N^{-1} CESS_j \geq e^{-k_3 - k_4}, \tag{7.45}$$

where $\lim_{\Delta_j \to 0} \lim_{N \to \infty} N^{-1} CESS_j$ *means convergence in probability.*

Proof. As $N \to \infty$, we have

$$N^{-1}\mathrm{CESS}_j := N^{-1}\left[\frac{\left(\sum_{i=1}^N \tilde{\rho}_{j,i}\right)^2}{\sum_{i=1}^N \tilde{\rho}_{j,i}^2}\right] = \left[\frac{\left(N^{-1}\sum_{i=1}^N a_j\tilde{\rho}_{j,i}\right)^2}{N^{-1}\sum_{i=1}^N \left(a_j\tilde{\rho}_{j,i}\right)^2}\right] \to \frac{\mathbb{E}\left[a_j\tilde{\rho}_j\right]^2}{\mathbb{E}\left[\left(a_j\tilde{\rho}_j\right)^2\right]},$$

where $a_j := \exp\{\sum_{c\in\mathcal{C}} \boldsymbol{\Phi}_c \Delta_j\}$. Since $a_j\tilde{\rho}_j$ is an unbiased estimate of ρ_j (see Theorem 7.1.3), then

$$\begin{aligned}\mathbb{E}\left[a_j\tilde{\rho}_j\right] &= \prod_{c\in\mathcal{C}} \mathbb{E}_{\mathbb{W}_{\boldsymbol{\Lambda}_c,j}}\left(\exp\left\{-\int_{t_{j-1}}^{t_j}\left(\phi_c\left(\boldsymbol{X}_t^{(c)}\right) - \boldsymbol{\Phi}_c\right)\right\}\right) \\ &= \mathbb{E}_{\bar{\mathbb{W}}_{\boldsymbol{\Lambda}}}\left(\exp\left\{-\sum_{c\in\mathcal{C}}\int_{t_{j-1}}^{t_j}\phi_c\left(\boldsymbol{X}_t^{(c)}\right)\right\}\right)\cdot a_j\end{aligned}$$

where $\bar{\mathbb{W}}_{\boldsymbol{\Lambda}}$ denotes the law of the collection of Brownian bridges $\{\mathbb{W}_{\boldsymbol{\Lambda}_c,j} : c \in \mathcal{C}\}$ for each j. Note that under the optimal distribution for $p(\kappa_c|R_c)$ (a Poisson distribution with intensity given in (7.42)), then $\mathbb{E}\left[\left(a_j\tilde{\rho}_j\right)^2\right] \leq 1$ [Fearnhead et al., 2008; Dai et al., 2021], so

$$\lim_{N\to\infty} N^{-1}\mathrm{CESS}_j \geq \mathbb{E}\left[a_j\tilde{\rho}_j\right]^2 = \left[\mathbb{E}_{\bar{\mathbb{W}}_{\boldsymbol{\Lambda}}}\left(\exp\left\{-\sum_{c\in\mathcal{C}}\int_{t_{j-1}}^{t_j}\phi_c\left(\boldsymbol{X}_t^{(c)}\right)\right\}\right)\right]^2 \cdot a_j^2.$$

If $f_c \sim \mathcal{N}_d(\boldsymbol{a}_c, \frac{b|\mathcal{C}|}{m}\boldsymbol{\Lambda}_c)$, then $\phi_c(\boldsymbol{x}) = \frac{1}{2}\left(\left(\frac{m}{b|\mathcal{C}|}\right)^2(\boldsymbol{x}-\boldsymbol{a}_c)^\intercal\boldsymbol{\Lambda}_c^{-1}(\boldsymbol{x}-\boldsymbol{a}_c) - \frac{md}{b|\mathcal{C}|}\right)$ which has global lower bound $\boldsymbol{\Phi}_c = -\frac{1}{2}\left(\frac{md}{b|\mathcal{C}|}\right)$ (since the minimum of ϕ_c occurs at the mean, $\boldsymbol{a}_c$). Then by considering small intervals (t_{j-1}, t_j) and taking the limit of $\Delta_j := t_j - t_{j-1} \to 0$, then

$$\begin{aligned}&\lim_{\Delta_j\to 0}\lim_{N\to\infty} N^{-1}\mathrm{CESS}_j \\ &\geq \lim_{\Delta_j\to 0}\left[\mathbb{E}\left(\mathbb{E}\left\{\mathbb{E}\left(\exp\left\{-\sum_{c\in\mathcal{C}}\int_{t_{j-1}}^{t_j}\phi_c\left(\boldsymbol{X}_t^{(c)}\right)\mathrm{d}t\right\}\middle|\boldsymbol{\xi}_j, \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})}\right)\middle|\vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})}\right\}\right)\right]^2 \cdot a_j^2 \\ &\geq \lim_{\Delta_j\to 0}\left[\mathbb{E}\left(\mathbb{E}\left\{\mathbb{E}\left(\exp\left\{-\frac{\Delta_j}{2}\sum_{c\in\mathcal{C}}\left(\frac{m}{b|\mathcal{C}|}\right)^2(\boldsymbol{x}_j^{(c)}-\boldsymbol{a}_c)^\intercal\boldsymbol{\Lambda}_c^{-1}(\boldsymbol{x}_j^{(c)}-\boldsymbol{a}_c)\right\}\middle|\boldsymbol{\xi}_j, \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})}\right)\middle|\vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})}\right\}\right)\right]^2 \\ &\geq \left[\mathbb{E}\left(\mathbb{E}\left\{\lim_{\Delta_j\to 0}\mathbb{E}\left(\exp\left\{-\frac{\Delta_j}{2}\sum_{c\in\mathcal{C}}\left(\frac{m}{b|\mathcal{C}|}\right)^2(\boldsymbol{x}_j^{(c)}-\boldsymbol{a}_c)^\intercal\boldsymbol{\Lambda}_c^{-1}(\boldsymbol{x}_j^{(c)}-\boldsymbol{a}_c)\right\}\middle|\boldsymbol{\xi}_j, \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})}\right)\middle|\vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})}\right\}\right)\right]^2,\end{aligned}$$

(by using a trapezoidal rule approximation of the integral and exploiting the use of small intervals) where $\lim_{\Delta_j \to 0}$ and expectations are exchanged using the dominated convergence theorem (as the exponential term is bounded above by 1 and its expectation exists [Dai et al., 2021, Appendix C]).

From (7.58) in Corollary 7.3.2, we note that $\boldsymbol{x}_j^{(c)}$ only depends $\boldsymbol{x}_{j-1}^{(c)}$ through $\boldsymbol{\xi}_j$ and $\boldsymbol{\zeta}_j^{(c)}$ for all $c \in \mathcal{C}$, and we have

$$\boldsymbol{x}_j^{(c)} \Big| \boldsymbol{\xi}_j, \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})} \sim \mathcal{N}_d \left(\mathbb{E}\left[\boldsymbol{x}_j^{(c)} \Big| \boldsymbol{\xi}_j, \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})} \right], \frac{T - t_j}{T - t_{j-1}} \Delta_j \boldsymbol{\Lambda}_c \right),$$

and consequently,

$$\left(\frac{T - t_j}{T - t_{j-1}} \Delta_j \right)^{-1} \sum_{c \in \mathcal{C}} (\boldsymbol{x}_j^{(c)} - \boldsymbol{a}_c)^\intercal \boldsymbol{\Lambda}_c^{-1} (\boldsymbol{x}_j^{(c)} - \boldsymbol{a}_c) \sim \chi^2(|\mathcal{C}|d, \lambda_j'),$$

with moment generating function $M_j(s) := \exp\left\{ \frac{\lambda_j' s}{1-2s} \right\} \cdot (1-2s)^{-\frac{|\mathcal{C}|d}{2}}$, where

$$\begin{aligned} \lambda_j' &= \left(\frac{T - t_j}{T - t_{j-1}} \Delta_j \right)^{-1} \sum_{c \in \mathcal{C}} \left(\mathbb{E}\left[\boldsymbol{x}_j^{(c)} \Big| \boldsymbol{\xi}_j, \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})} \right] - \boldsymbol{a}_c \right)^\intercal \boldsymbol{\Lambda}_c^{-1} \left(\mathbb{E}\left[\boldsymbol{x}_j^{(c)} \Big| \boldsymbol{\xi}_j, \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})} \right] - \boldsymbol{a}_c \right) \\ &= \left(\frac{T - t_j}{T - t_{j-1}} \Delta_j \right)^{-1} |\mathcal{C}| \sigma_{t_j}^2, \end{aligned}$$

with

$$\sigma_{t_j}^2 := \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \left(\mathbb{E}\left[\boldsymbol{x}_j^{(c)} \Big| \boldsymbol{\xi}_j, \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})} \right] - \boldsymbol{a}_c \right)^\intercal \boldsymbol{\Lambda}_c^{-1} \left(\mathbb{E}\left[\boldsymbol{x}_j^{(c)} \Big| \boldsymbol{\xi}_j, \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})} \right] - \boldsymbol{a}_c \right).$$

Letting $s = -\frac{1}{2} \left(\frac{m}{b|\mathcal{C}|} \right)^2 \left(\frac{T - t_j}{T - t_{j-1}} \right) \Delta_j^2$, then

$$\begin{aligned} \lim_{\Delta_j \to 0} \lim_{N \to \infty} N^{-1} \mathrm{CESS}_j &\geq \left[\mathbb{E}\left(\mathbb{E}\left\{ \lim_{\Delta_j \to 0} \exp\left\{ \frac{\lambda_j' s}{1-2s} \right\} \middle| \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})} \right\} \right) \right]^2 \cdot (1-2s)^{-|\mathcal{C}|d} \\ &\geq \left[\mathbb{E}\left(\mathbb{E}\left\{ \lim_{\Delta_j \to 0} \exp\left\{ \frac{-\frac{1}{2}\left(\frac{m^2}{b^2 C} \right) \sigma_{t_j}^2 \Delta_j}{1-2s} \right\} \middle| \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})} \right\} \right) \right]^2 \cdot (1-2s)^{-|\mathcal{C}|d}. \end{aligned}$$

From (7.58), we have

$$\mathbb{E}\left[\boldsymbol{x}_j^{(c)} \Big| \boldsymbol{\xi}_j, \vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})} \right] = \left[\frac{\Delta_j^2}{T - t_{j-1}} \right]^{\frac{1}{2}} \boldsymbol{\xi}_j + \frac{T - t_j}{T - t_{j-1}} \boldsymbol{x}_{j-1}^{(c)} + \frac{t_j - t_{j-1}}{T - t_{j-1}} \tilde{\boldsymbol{x}}_{j-1},$$

and so we have $\lim_{\Delta_j \to 0} \sigma_{t_j}^2 =: \nu_j$ where ν_j is given in (7.44). Using Jensen's inequality, we can get

$$\lim_{\Delta_j \to 0} \lim_{N \to \infty} N^{-1} \mathrm{CESS}_j \geq \lim_{\Delta_j \to 0} \left[\exp\left\{ \frac{-\frac{1}{2} \mathbb{E}\left[\nu_j \right] \left(\frac{m^2}{b^2 |\mathcal{C}|} \right) \Delta_j}{1-2s} \right\} \right]^2 \cdot (1-2s)^{-|\mathcal{C}|d}$$

$$\geq \lim_{\Delta_j \to 0} \exp\left\{ \frac{-\mathbb{E}\left[\nu_j\right]\left(\frac{m^2}{b^2|\mathcal{C}|}\right)\Delta_j}{1-2s} \right\} \cdot (1-2s)^{-|\mathcal{C}|d}. \tag{7.46}$$

Consider the first term in (7.46), then taking the limit $\Delta_j \to 0$ implies that $s \to 0$, and if $\Delta_j \leq \frac{b^2|\mathcal{C}|k_3}{\mathbb{E}[\nu_j]m^2}$ for some $k_3 > 0$, then

$$\exp\left\{ \frac{-\mathbb{E}\left[\nu_j\right]\left(\frac{m^2}{b^2|\mathcal{C}|}\right)\Delta_j}{1-2s} \right\} \geq \exp\left\{-k_3\right\}. \tag{7.47}$$

Similarly for the second term in (7.46), if $\Delta_j \leq \left(\frac{b^2|\mathcal{C}|k_4}{2m^2 d}\right)^{\frac{1}{2}}$, we have

$$\begin{aligned}
(1-2s)^{-|\mathcal{C}|d} &\geq \exp\left\{4s|\mathcal{C}|d\right\} \\
&= \exp\left\{ 4|\mathcal{C}|d\left(-\frac{1}{2}\left(\frac{m}{b|\mathcal{C}|}\right)^2 \left(\frac{T-t_j}{T-t_{j-1}}\right)\Delta_j^2 \right) \right\} \\
&= \exp\left\{ -2\left(\frac{m^2}{b^2|\mathcal{C}|}\right) d\Delta_j^2 \right\} \geq \exp\left\{-k_4\right\}.
\end{aligned} \tag{7.48}$$

Combining the bounds in (7.47) and (7.48), and taking the limit $\Delta_j \to 0$ over sequences of $t_j - t_{j-1} \to 0$, with (7.43), we arrive at the result given in the theorem. ■

Remark 7.3.3. *In Theorem 7.3.2, ν_j (as defined in* (7.44)*) describes the* scaled/weighted *average variation of the $|\mathcal{C}|$ trajectories of the distribution of their proposed update locations with respect to their individual sub-posterior means (i.e. describing how far $\boldsymbol{x}_{j-1}^{(c)}$ is from $\boldsymbol{a}_c$). Since the GBF approach has $|\mathcal{C}|$ trajectories which are initialised from their respective sub-posterior distributions and coalesce to a common end point, this variation is mainly determined by a combination of: (i) how large the time horizon T is; (ii) how large the interval we are simulating over for this iteration $(t_{j-1}, t_j]$; and (iii) how much the sub-posteriors conflict which we determine by looking at the variation in their means as per* (7.27)*. Given a weighted particle set from the $(j-1)$th iteration of the algorithm, $\{\vec{\boldsymbol{x}}_{j-1,i}^{(\mathcal{C})}, w_{j-1,i}^{(\mathcal{C})}\}_{i=1}^N$, a natural estimator for $\mathbb{E}\left[\nu_j\right]$ is*

$$\widehat{\mathbb{E}\left[\nu_j\right]} = \sum_{i=1}^N w_{j-1,i}^{(\mathcal{C})} \left(\frac{1}{|\mathcal{C}|} \sum_{c\in\mathcal{C}} \left(\boldsymbol{x}_{j-1,i}^{(c)} - \boldsymbol{a}_c\right)^{\intercal} \boldsymbol{\Lambda}_c^{-1} \left(\boldsymbol{x}_{j-1,i}^{(c)} - \boldsymbol{a}_c\right) \right). \tag{7.49}$$

Following Theorem 7.3.2 and Remark 7.3.3 we now have the additional problem of specifying k_3 and k_4, and using the result to develop practical guidance. We do so by first choosing a lower bound on the conditional effective sample size that we would tolerate, $\zeta' \in (0,1)$, and select k_3 and k_4 such that $e^{-k_3-k_4} = \zeta'$ and compute $t_j = \min\left\{T, t_{j-1} + \tilde{\Delta}_j\right\}$ recursively at each iteration until $j = n$ such that $t_n = T$.

Note that we expect to have very different performance with different choices of k_3 and k_4. For instance, we can obtain a very high CESS_j by simply choosing k_3 very small and setting $k_4 = -\log(\zeta') - k_3$, which ultimately leads to having very small intervals sizes $\tilde{\Delta}_j$. Choosing small interval sizes may help computationally simulating $\tilde{\rho}_j$, but this comes at the cost of having more iterations of the algorithm, leading to an increased communication between the cores. Natural choices for jointly specifying k_3 and k_4 are ones which lead to the largest interval size which still satisfies $N^{-1}\text{CESS}_j \geq \zeta' \in (0,1)$, as this minimises the number of iterations in Algorithm 7.1.2.

We now consider the previously introduced *regular* and *irregular* (*adaptive*) mesh selection of $\mathcal{P}$ in Section 7.3.2.1 and Section 7.3.2.2 respectively. As noted in Section 5.2.3, although the implementational guidance for the (standard) BF approach can be found in Dai et al. [2021, Section 3.2], the guidance developed in this section for choosing T and $\mathcal{P}$ also applies to the BF algorithm (Algorithm 5.2.2) by setting $\mathbf{\Lambda}_c = \mathbb{I}_d$ for all $c \in \mathcal{C} = \{1, \ldots, C\}$.

7.3.2.1 A regular mesh construction

For algorithmic simplicity and to avoid computing (7.43) at each iteration of Algorithm 7.1.2 to determine the interval size for iteration j of the algorithm, it is possible to construct a *regular mesh* $\mathcal{P}$ whereby each interval size is the same for each iteration, i.e. $\tilde{\Delta}_j = \Delta$ for each $j = 1, \ldots, n$ where $n = \lceil T/\Delta \rceil$ (where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x). This simplification of regularity was suggested in Dai et al. [2021, Remark 6]. They noted that for large datasets in which observations were randomly allocated to sub-posteriors, that one would expect sub-posterior heterogeneity to be small. Hence one would expect $\mathbb{E}[\nu_j]$ to be small (of $\mathcal{O}(m^{-1})$). In their simulations in Dai et al. [2021, Section 3 and 4], they set $k_3 = k_4 = 1$ and let $\Delta := t_{j-1} - t_j = \sqrt{(b^2|\mathcal{C}|k_4)/(2m^2 d)}$ for all j. The rationale presented in Dai et al. [2021, Remark 6] does not hold in generality so in this section, we instead develop a more systematic way to construct a regular mesh. In particular, setting $k_3 = k_4$ as they suggest is sub-optimal.

Given a user specified lower bound on CESS_j, $\zeta' \in (0,1)$, we want to minimise the number of iterations of Algorithm 7.1.2 Step 2. This is achieved with reference to Theorem 7.3.2 and by choosing a combination of k_3 and k_4 such that: (i) $\exp\{-k_3 - k_4\} \geq \zeta'$ (i.e. CESS_j for any j does not violate the chosen ζ'); and (ii), $\frac{b^2|\mathcal{C}|k_3}{\mathbb{E}[\nu_j]m^2} \geq \sqrt{\frac{b^2|\mathcal{C}|k_4}{2m^2 d}}$ for each j. The difficulty here is that at each iteration, we must compute the average variation of the trajectories, $\mathbb{E}[\nu_j]$. Of course, this is not possible directly and so an estimate $\widehat{\mathbb{E}[\nu_j]}$ is computed as per (7.49). To ensure the chosen ζ' is not violated at *any* iteration we follow the guidance of (7.43) by taking a supremum over all intervals of this estimator (i.e. $\sup_j \widehat{\mathbb{E}[\nu_j]}$). This choice allows us to specify k_3 and k_4 to obtain n and $\mathcal{P}$.

For ease of practical implementation of Algorithm 7.1.2, it is desirable to avoid any recursive definitions of n and $\mathcal{P}$ (i.e. they are specified prior to calling Algorithm 7.1.2 Step 2 where they are required). In this setting we would need to estimate $\sup_j \widehat{\mathbb{E}[\nu_j]}$ based upon *only* the *initial* (weighted) sub-posterior realisations $\{\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})}, w_{0,i}^{(\mathcal{C})}\}_{i=1}^{M}$ obtained in Algorithm 7.1.2 Step 1b.

Following Remark 7.3.3, we would expect $\mathbb{E}[\nu_j]$ to be maximised at $t = T$ (corresponding to (7.51)), but in some instance may also occur at $t = 0$ (corresponding to (7.52)). In most practical applications of GBF it will be at $t = T$ as the proposal for the coalescence of the $|\mathcal{C}|$ stochastic processes has a Gaussian distribution with mean $\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}$ with variance $T\boldsymbol{\Lambda}_{\mathcal{C}}$ (as a consequence of Theorem 7.1.2 and considering $s = 0$ and $t = T$). On the other hand, if the sub-posterior means are very close together, the largest variation in the trajectories from their respective means could occur at the start of the bridge. As such, we propose taking the larger of these two scenarios to arrive at the following approximation:

$$\sup_j \widehat{\mathbb{E}\left[\nu_j\right]} \approx \max\{\Psi_1, \Psi_2\}, \tag{7.50}$$

where

$$\Psi_1 := \sum_{i=1}^{M} w_{0,i}^{(\mathcal{C})} \frac{1}{|\mathcal{C}|} \sum_{c\in\mathcal{C}} \left(\tilde{\boldsymbol{x}}_{0,i}^{(\mathcal{C})} - \boldsymbol{a}_c\right)^{\intercal} \boldsymbol{\Lambda}_c^{-1} \left(\tilde{\boldsymbol{x}}_{0,i}^{(\mathcal{C})} - \boldsymbol{a}_c\right), \tag{7.51}$$

$$\Psi_2 := \sum_{i=1}^{M} w_{0,i}^{(\mathcal{C})} \frac{1}{|\mathcal{C}|} \sum_{c\in\mathcal{C}} \left(\boldsymbol{x}_{0,i}^{(c)} - \boldsymbol{a}_c\right)^{\intercal} \boldsymbol{\Lambda}_c^{-1} \left(\boldsymbol{x}_{0,i}^{(c)} - \boldsymbol{a}_c\right), \tag{7.52}$$

and where $w_{0,i}^{(\mathcal{C})}$ are the initial particle weights given in Algorithm 7.1.2 Step 1b.

Our approximation of $\sup_j \widehat{\mathbb{E}\left[\nu_j\right]}$ has obvious limitations: it may not be conservative enough to ensure the user chosen ζ' is not breached; and it may be too conservative and lead to choosing n too high. In practice we have found it to be a robust approximation.

Once we have a suitable estimate of $\sup_j \widehat{\mathbb{E}\left[\nu_j\right]}$, we need to find a suitable choice for k_3 and k_4 to ensure that we always choose the RHS side of (7.43) (as that leads to a regular mesh) and satisfies ζ'. As there are many combinations of k_3 and k_4 which can return a regular mesh, we aim to find the combination which returns the largest interval size. We can do this by means of the following proposition which considers the jth interval of the partition:

Proposition 7.3.3. *Considering the jth interval of $\mathcal{P}$ (i.e. $[t_{j-1}, t_j]$), given a user-specified threshold $\zeta' \in (0,1)$ and estimate $\widehat{\mathbb{E}[\nu_j]}$ of $\mathbb{E}[\nu_j]$, then the largest interval size which satisfies $N^{-1}CESS_j \geq \zeta'$ is given by*

$$\tilde{\Delta}_j = \sqrt{\frac{b^2|\mathcal{C}|k_{4,j}}{2m^2 d}},$$

where,

$$k_{4,j} := \frac{\left(\frac{\widehat{\mathbb{E}[\nu_j]}^2 m^2}{2b^2|\mathcal{C}|d} - 2\log(\zeta')\right) - \sqrt{\left(2\log(\zeta') - \frac{\widehat{\mathbb{E}[\nu_j]}^2 m^2}{2b^2|\mathcal{C}|d}\right)^2 - 4\log(\zeta')^2}}{2}. \tag{7.53}$$

Proof. Using Theorem 7.3.2, then for iteration j, we want to choose $\exp\{-k_{3,j} - k_{4,j}\} = \zeta' \in (0,1)$,

and so $k_{3,j} = -\log(\zeta') - k_{4,j}$. By substituting this into (7.43), we can choose the mesh size as

$$\tilde{\Delta}_j = \min\left\{\frac{b^2|\mathcal{C}|[-\log(\zeta') - k_{4,j}]}{\mathbb{E}[\nu_j]m^2}, \left(\frac{b^2|\mathcal{C}|k_{4,j}}{2m^2 d}\right)^{\frac{1}{2}}\right\}, \tag{7.54}$$

where $k_{4,j} < -\log(\zeta')$ (in order to ensure that $k_{3,j} > 0$). Here, we want the largest interval which satisfies $N^{-1}\text{CESS}_j \geq \zeta'$. This corresponds to choosing $k_{4,j}$ with

$$\begin{aligned}
\frac{b^2|\mathcal{C}|[-\log(\zeta') - k_{4,j}]}{\mathbb{E}[\nu_j]m^2} &= \left(\frac{b^2|\mathcal{C}|k_{4,j}}{2m^2 d}\right)^{\frac{1}{2}} \\
\implies \frac{b^4|\mathcal{C}|^2[-\log(\zeta') - k_{4,j}]^2}{\mathbb{E}[\nu_j]^2 m^4} &= \frac{b^2|\mathcal{C}|k_{4,j}}{2m^2 d} \\
\implies [-\log(\zeta') - k_{4,j}]^2 &= \frac{\mathbb{E}[\nu_j]^2 m^2}{2b^2|\mathcal{C}|d} k_{4,j} \\
\implies \log(\zeta')^2 + 2k_{4,j}\log(\zeta') + k_{4,j}^2 &= \frac{\mathbb{E}[\nu_j]^2 m^2}{2b^2|\mathcal{C}|d} k_{4,j} \\
\implies k_{4,j}^2 + \left(2\log(\zeta') - \frac{\mathbb{E}[\nu_j]^2 m^2}{2b^2|\mathcal{C}|d}\right) k_{4,j} + \log(\zeta')^2 &= 0.
\end{aligned} \tag{7.55}$$

Applying the quadratic formula to solve (7.55) gives

$$k_{4,j} = \frac{\left(\frac{\mathbb{E}[\nu_j]^2 m^2}{2b^2|\mathcal{C}|d} - 2\log(\zeta')\right) \pm \sqrt{\left(2\log(\zeta') - \frac{\mathbb{E}[\nu_j]^2 m^2}{2b^2|\mathcal{C}|d}\right)^2 - 4\log(\zeta')^2}}{2}.$$

Note that we have the constraints that $0 < k_{4,j} < -\log(\zeta')$, and since from (7.55), we have

$$k_{4,j}^2 + \left(2\log(\zeta') - \frac{\mathbb{E}[\nu_j]^2 m^2}{2b^2|\mathcal{C}|d}\right) k_{4,j} = -\log(\zeta')^2,$$

then we will always choose the smaller root and arrive at the statement of the theorem. ■

Using Proposition 7.3.3, we can substitute our estimate of $\sup_j \widehat{\mathbb{E}[\nu_j]}$ into (7.53), and compute the regular interval size $\Delta := \sqrt{\frac{b^2|\mathcal{C}|k_4}{2m^2 d}}$ and $n = \lceil T/\Delta \rceil$. In effect, we are setting $k_4 = \sup_j k_{4,j}$. Whilst choosing $k_4 < \sup_j k_{4,j}$ would lead to another regular mesh, we are attempting to find the choice of k_3 and k_4 which leads to the largest interval size. This process is summarised in Algorithm 7.3.1.

Algorithm 7.3.1 Computing regular mesh $\mathcal{P}$.

1. **Input:** Time $T > 0$ and importance weighted particles $\{\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})}, w_{0,i}^{(\mathcal{C})}\}_{i=1}^M$.
2. Compute estimate of $\sup_j \widehat{\mathbb{E}[\nu_j]}$ as per (7.50).
3. Compute k_4 using the estimate from Step 2 as per (7.53).
4. Compute $\Delta := \sqrt{\frac{b^2|\mathcal{C}|k_4}{2m^2 d}}$ and let $n = \lceil T/\Delta \rceil$.
5. For $j = 1, \ldots, n$, let $t_j = \min\{T, t_{j-1} + \Delta\}$.
6. **Output:** $\mathcal{P} := \{t_0, \ldots, t_n\}$.

7.3.2.2 An adaptive mesh construction

While the regular mesh introduced in Section 7.3.2.1 offers the simplicity of choosing the temporal partition prior to the iterative steps in Algorithm 7.1.2 Step 2, the construction of the regular mesh essentially computes the *worst* case scenario of the trajectory variation, and consequently this can lead to an excessive number of points in $\mathcal{P}$. Specifically, it can be an overly conservative approach to constructing $\mathcal{P}$. An alternative mesh construction is to have an *irregular* (or *adaptive*) mesh which we outline in this section.

Suppose we are at the beginning of the jth iteration of Algorithm 7.1.2 Step 2, we have in effect simulated our $|\mathcal{C}|$ stochastic processes up to time $t_{j-1} < T$. We can now consider the placement of the next point in the partition, t_j, with reference to the user chosen $\zeta' \in (0,1)$. In particular, we want the interval to be as large as possible while ensuring that the CESS_j does not degrade by more than ζ'. To do this we can compute an estimate of $\mathbb{E}[\nu_j]$ as per (7.49) and appeal to Proposition 7.3.3 in order to choose $k_{4,j}$, and consequently the interval size Δ_j for that particular iteration j to set $t_j = \min\{t_{j-1} + \Delta_j, T\}$. Once we reach T we simply halt iterating Algorithm 7.1.2 Step 2.

In contrast to the regular mesh construction in Section 7.3.2.1, we cannot compute the temporal mesh prior to Algorithm 7.1.2 Step 2. Therefore, the computation of the interval size for iteration j must be done immediately after Step 2a and prior to Step 2b of Algorithm 7.1.2. In this setting, the number of steps in Algorithm 7.1.2 (i.e. n) is not known in advance. Given the construction of the regular mesh assumes the *worst case* interval in selecting the mesh size, we would expect that the overall n would be lower in our adaptive approach. Indeed, we show this empirically in our simulation studies in Section 7.4. We summarise this approach in Algorithm 7.3.2.

Algorithm 7.3.2 Computing adaptive mesh $\mathcal{P}$ (computing Δ_j at iteration j immediately after Algorithm 7.1.2 Step 2a).

---

1. **Input:** Time $T > 0$ and importance weighted particles $\{\vec{\boldsymbol{x}}_{j-1,i}^{(\mathcal{C})}, w_{j-1,i}^{(\mathcal{C})}\}_{i=1}^{N}$.
2. Compute $\widehat{\mathbb{E}[\nu_j]}$ as per (7.49).
3. Compute k_4 with the estimate from Step 2 as per (7.53).
4. Compute $t_j = \min\left\{T, t_{j-1} + \sqrt{\frac{b^2|\mathcal{C}|k_4}{2m^2 d}}\right\}$.
5. **Output:** $\Delta_j := t_j - t_{j-1}$.

7.3.3 Practical implementational considerations

One of the main aims of this thesis is to further develop the Fusion methodology so that it can be applied to a wider range of practical settings. In this chapter, we have outlined theory and methodology to tackle the general fusion problem of sampling from $f^{(\mathcal{C})}$ by combining samples from the sub-posteriors f_c for $c \in \mathcal{C}$, and the implementational guidance provided thus far is for the general application of GBF methodology. However, in many practical settings there will be additional constraints which require us to modify Algorithm 7.1.2 appropriately. Examples include

settings where latency between cores is problematic, or in scenarios where functional evaluations of the sub-posterior densities f_c are not available. In this section, we revisit the practical implementational considerations outlined by Dai et al. [2021, Section 3.7] for the (standard) BF approach (also discussed in Section 5.2.3) and generalise them to our setting. To clarify, the implementation of our methodology in examples presented in Sections 7.4–7.5 do not exploit these modifications.

7.3.3.1 Reducing communication between the cores

For BF, Dai et al. [2021, Section 3.7.1–3.7.2] identified two steps where communication between cores could be reduced (see Section 5.2.3.2). In particular, it is possible to limit the amount of communication between cores when initialising the particle set, and also when propagating the particles in the iterative steps of the algorithm. In a distributed/parallel setting, it is desirable to reduce the number of communication between cores since there is a latency penalty for each communication leading to a more computationally expensive algorithm. In this section, we outline how we can take these ideas from Dai et al. [2021, Section 3.7.1–3.7.2] and apply it in our setting.

In Algorithm 7.1.2 Step 1b, the particles are composed by pairing the sub-posterior draws index-wise to obtain $\{\vec{\boldsymbol{x}}_{0,i}^{(\mathcal{C})}\}_{i=1}^{M}$ which requires a communication between the cores. To fully initialise the algorithm, we must assign importance weights to the particles which requires an additional two communications between the cores; namely a communication back to the individual cores to provide the weighted mean of the particles $\tilde{\boldsymbol{x}}_{0,i}$, and a communication between the cores to compute $\rho_{0,i}(\vec{\boldsymbol{x}}_{0,\cdot}^{(\mathcal{C})})$ (since (6.7) can be decomposed into a product of $|\mathcal{C}|$ terms corresponding to the individual contributions from each sub-posterior). Following the approach of Dai et al. [2021, Section 3.7.1], let $\tilde{\theta} \in \mathbb{R}^d$ be a weighted average of approximate modes (or means) of each sub-posterior. Noting that this can be computed in a single pre-processing step prior to initialisation, then we can modify the proposal mechanism for the initial draw to be from the density

$$\tilde{f}_c\left(\boldsymbol{x}_0^{(c)}\right) \propto \exp\left\{-\frac{(\boldsymbol{x}_0^{(c)}-\tilde{\boldsymbol{\theta}})^\intercal \boldsymbol{\Lambda}_c^{-1}(\boldsymbol{x}_0^{(c)}-\tilde{\boldsymbol{\theta}})}{2T}\right\} \cdot f_c\left(\boldsymbol{x}_0^{(c)}\right), \tag{7.56}$$

then by modifying the algorithm by replacing ρ_0 with

$$\tilde{\varrho}_0 := \exp\left\{\frac{(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}-\tilde{\boldsymbol{\theta}})^\intercal \boldsymbol{\Lambda}_{\mathcal{C}}^{-1}(\tilde{\boldsymbol{x}}_0^{(\mathcal{C})}-\tilde{\boldsymbol{\theta}})}{2T}\right\}, \tag{7.57}$$

where $\boldsymbol{\Lambda}_{\mathcal{C}}^{-1} := (\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1})$, we can see that

$$\tilde{\varrho}_0\left(\vec{\boldsymbol{x}}_0^{(\mathcal{C})}\right) \cdot \prod_{c\in\mathcal{C}} \tilde{f}_c\left(\boldsymbol{x}_0^{(c)}\right) \propto \rho_0\left(\vec{\boldsymbol{x}}_0^{(\mathcal{C})}\right) \cdot \prod_{c\in\mathcal{C}} f_c\left(\boldsymbol{x}_0^{(c)}\right).$$

Since we subsequently re-normalise the importance weights, we do not need to compute any constant of proportionality for $\tilde{\varrho}_0$. Adopting this approach means that we can sample from $\tilde{f}_c$ on each core

independently and evaluate the modified importance weight without any further communication between the cores. This therefore reduces the number of communications required to initialise the particle set from three (in the original formulation) to two (since this approach does require one communication to compute $\tilde{\theta}$). The modified initialisation is summarised in Algorithm 7.3.3.

Algorithm 7.3.3 Particle set initialisation modification (to replace Algorithm 7.1.2 Step 1b).

1(b) For k in 1 to M,

(i) $\vec{\boldsymbol{x}}_{0,k}^{(\mathcal{C})}$: For $c \in \mathcal{C}$, simulate $\boldsymbol{x}_{0,k}^{(c)} \sim \tilde{f}_c$ (7.56). Set $\vec{\boldsymbol{x}}_{0,k}^{(\mathcal{C})} := (\boldsymbol{x}_{0,k}^{(c_1)}, \ldots, \boldsymbol{x}_{0,k}^{(c_{|\mathcal{C}|})})$.

(ii) Compute un-normalised weight $w_{0,k}^{(\mathcal{C})\prime} := \left(\prod_{c\in\mathcal{C}} w_k^{(c)}\right) \cdot \tilde{\varrho}_0(\vec{\boldsymbol{x}}_{0,k}^{(\mathcal{C})})$ as per (7.57).

There is also scope to reduce the number of communications required to propagate the particle set in Algorithm 7.1.2 Step 2(b)i. To propagate the particles, there is a communication between the cores in order to compute $\vec{\boldsymbol{M}}_j^{(\mathcal{C})} := \vec{\boldsymbol{M}}_{t_{j-1},t_j}^{(\mathcal{C})}$ as per (7.11) since this requires the current position of each of the $|\mathcal{C}|$ trajectories. Once we have computed this and propagated the samples, a further communication back to the cores would be necessary so that each core can compute their contribution to the $\tilde{\rho}_j$ importance weight. Alternatively, we can utilise Corollary 7.3.2 so that each of the $|\mathcal{C}|$ processes can propagate their own individual particles to compose $\vec{\boldsymbol{x}}_j^{(\mathcal{C})}$.

Corollary 7.3.2. *Simulating $\vec{\boldsymbol{x}}_j^{(\mathcal{C})} \sim \mathcal{N}_d\left(\vec{\boldsymbol{M}}_j^{(\mathcal{C})}, \boldsymbol{V}_j\right)$, the required transition from $\vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})}$ to $\vec{\boldsymbol{x}}_j^{(\mathcal{C})}$ in Algorithm 7.1.2 Step 2(b)i, can be expressed as*

$$\boldsymbol{x}_j^{(c)} = \left[\frac{\Delta_j^2}{T - t_{j-1}}\right]^{\frac{1}{2}} \boldsymbol{\xi}_j + \left[\frac{T - t_j}{T - t_{j-1}}\Delta_j\right]^{\frac{1}{2}} \boldsymbol{\eta}_j^{(c)} + \boldsymbol{M}_j^{(c)}, \tag{7.58}$$

where $\boldsymbol{\xi}_j \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{\Lambda}_{\mathcal{C}})$, $\boldsymbol{\eta}_j^{(c)} \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{\Lambda}_c)$ and $\boldsymbol{M}_j^{(c)}$ is the cth sub-vector of $\vec{\boldsymbol{M}}_j^{(\mathcal{C})}$ given by (7.11).

Proof. From part (a) of Theorem 7.1.2, we have $\vec{\boldsymbol{x}}_j^{(\mathcal{C})} \sim \mathcal{N}_d\left(\vec{\boldsymbol{M}}_j^{(\mathcal{C})}, \boldsymbol{V}_j\right)$ where $\vec{\boldsymbol{M}}_j^{(\mathcal{C})} := \vec{\boldsymbol{M}}_{t_{j-1},t_j}^{(\mathcal{C})}$ is given by (7.11) and $\boldsymbol{V}_j := \boldsymbol{V}_{t_{j-1},t_j}$ is given by (7.12). From (7.58), the mean and covariance matrix of $\vec{\boldsymbol{x}}_j^{(\mathcal{C})}$ given $\vec{\boldsymbol{x}}_j^{(\mathcal{C})}$ are also given by $\vec{\boldsymbol{M}}_j^{(\mathcal{C})}$ and $\boldsymbol{V}_j$ as required. ■

Notice that the interaction between the $|\mathcal{C}|$ trajectories only occurs through $\tilde{\boldsymbol{x}}_{j-1}$ which can be computed at the previous iteration, and we can communicate this along with $\boldsymbol{\xi}_j$ at the same time. This removes an unnecessary communication between the cores at every iteration, resulting in a more efficient algorithm if latency is a concern. This approach is presented in Algorithm 7.3.4.

Algorithm 7.3.4 Particle set propagation modification (to replace Algorithm 7.1.2 Step 2(b)i).

2(b)i.

(A) For $c \in \mathcal{C}$, simulate $\boldsymbol{x}_{j,i}^{(c)} | (\tilde{\boldsymbol{x}}_{j-1,i}, \boldsymbol{x}_{j-1,i}^{(c)})$ in (7.58).

(B) Set $\vec{\boldsymbol{x}}_{j,i}^{(\mathcal{C})} := (\boldsymbol{x}_{j,i}^{(c_1)}, \ldots, \boldsymbol{x}_{j,i}^{(c_{|\mathcal{C}|})})$ and compute $\tilde{\boldsymbol{x}}_{j,i}^{(\mathcal{C})} := (\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1})^{-1} (\sum_{c\in\mathcal{C}} \boldsymbol{\Lambda}_c^{-1} \boldsymbol{x}_{j,i}^{(c)})$.

7.3.3.2 Alternative methods for updating the particle set weights

In this chapter (and in Chapter 6), we have assumed that we have been able to compute functionals of each sub-posterior f_c for $c \in \mathcal{C}$, but as we have noted in Section 5.2.3.3, there are many settings where it may be impractical or infeasible to do so. This may be case if there is some form of intractability of the sub-posteriors [Andrieu and Roberts, 2009], or if the evaluation of such quantities is simply too computationally expensive (for instance in large data settings [Pollock et al., 2020; Bouchard-Côté et al., 2018; Bierkens et al., 2019; Baker et al., 2019; Dai et al., 2021]). In these settings, we no longer are able to evaluate ϕ_c in (6.9) which is necessary to update the particle weights in the iterative steps of Algorithm 7.1.2. However, we have seen in Section 5.2.3.3, it is possible to consider alternative unbiased estimators for $\tilde{\rho}_j$ in Step 2c.

Corollary 7.3.3. *[Dai et al., 2021, Corollary 3] The estimator*

$$\tilde{\varrho}_j\left(\vec{\boldsymbol{x}}_{j-1}^{(\mathcal{C})}, \vec{\boldsymbol{x}}_j^{(\mathcal{C})}\right) := \prod_{c\in\mathcal{C}}\left(\frac{\Delta_j^{\kappa_c}\cdot e^{-\bar{U}_X^{(c)}\Delta_j}}{\kappa_c!\cdot p\left(\kappa_c|R_c\right)}\prod_{k_c=1}^{\kappa_c}\left(\bar{U}_X^{(c)} - \tilde{\phi}_c\left(\boldsymbol{X}_{\xi_{c,k_c}}^{(c)}\right)\right)\right), \tag{7.59}$$

where $\tilde{\phi}_c$ is an unbiased estimator of ϕ_c and $\bar{U}_j^{(c)}$ is a constant such that $\tilde{\phi}_c(\boldsymbol{x}) \leq \bar{U}_j^{(c)}$ for $\boldsymbol{x} \in R_c$.

Proof. This follows directly from Theorem 7.1.3. ■

The estimator $\tilde{\varrho}_j$ in Corollary 7.3.3 can therefore be used as a substitute for $\tilde{\rho}_j$ in Algorithm 7.1.2 Step 2c. However, we must be careful in constructing $\tilde{\varrho}_j$ since its introduction typically increases the variance of the estimator which increases the variance in the weights. In particular, by using Corollary 7.3.3, the number of expected functional evaluations will change from K to K' and so we must consider the growth in the ratio K'/K as $m_c \to \infty$ [Pollock et al., 2020; Dai et al., 2021].

Consider the example setting provided in Dai et al. [2021, Appendix E] and Section 5.2.3.3), where we have a large number of data points associated to each sub-posterior (i.e we have $m_c \gg 1$ data points for core $c \in \mathcal{C}$) then computing ϕ_c in (6.9) is an expensive $\mathcal{O}(m_c)$ operation. Assuming that the sub-posteriors admit a structure with conditional independence and can be factorised as

$$f_c(\boldsymbol{x}) \propto \prod_{i=1}^{m_c} l_{i,c}(\boldsymbol{x}), \tag{7.60}$$

then we could use the following naive unbiased estimator for ϕ_c^{dl}:

$$\tilde{\phi}_c(\boldsymbol{x}) = \frac{m_c}{2}\left(\nabla \log l_{I,c}(\boldsymbol{x}^*)^\intercal \boldsymbol{\Lambda}_c \nabla \log l_{J,c}(\boldsymbol{x}^*) + \mathrm{Tr}\big(\boldsymbol{\Lambda}_c \nabla^2 \log l_{I,c}(\boldsymbol{x}^*)\big)\right), \tag{7.61}$$

where $I, J \overset{\text{iid}}{\sim} \mathcal{U}\{1,\ldots,m_c\}$. Although using such an estimator has the advantage of having $\mathcal{O}(1)$ cost when evaluating, this comes at the cost of an $\mathcal{O}(m_c)$ inflation in the expected number of

evaluations when evaluating $\tilde{\varrho}_j$ over $\tilde{\rho}_j$. However, following the approach of Pollock et al. [2020, Section 4] and Dai et al. [2021, Appendix E], we can choose some suitable *control variates* and compute $\nabla \log f_c$ and $\nabla^2 \log f_c$ at points *close* to either the mode of the sub-posterior, $\hat{\boldsymbol{x}}_c$, or the mode of the target posterior $\hat{\boldsymbol{x}}$ (where close means within $\mathcal{O}(m_c^{-\frac{1}{2}})$ of the true respective modes). Computing these control variates will typically be one-time $\mathcal{O}(m_c)$ computations. Now let

$$\tilde{\alpha}_{I,c}(\boldsymbol{x}) := n \cdot [\nabla \log l_{I,c}(\boldsymbol{x}) - \nabla \log l_{I,c}(\boldsymbol{x}^*)], \tag{7.62}$$

$$\tilde{H}_{I,c}(\boldsymbol{x}) := n \cdot [\nabla^2 \log l_{I,c}(\boldsymbol{x}) - \nabla^2 \log l_{I,c}(\boldsymbol{x}^*)], \tag{7.63}$$

then since $\log f_c(\boldsymbol{x}) = \sum_{i=1}^{m_c} \log l_{i,c}(\boldsymbol{x})$, we have

$$\mathbb{E}_{\mathcal{A}}\left[\tilde{\alpha}_{I,c}(\boldsymbol{x})\right] = \alpha_c(\boldsymbol{x}), \quad \mathbb{E}_{\mathcal{A}}\left[\tilde{H}_{I,c}(\boldsymbol{x})\right] = H_c(\boldsymbol{x}). \tag{7.64}$$

where $\alpha_c(\boldsymbol{x}) := \nabla \log f_c(\boldsymbol{x}) - \nabla \log f_c(\boldsymbol{x}^*)$ and $H_c(\boldsymbol{x}) := \nabla^2 \log f_c(\boldsymbol{x}) - \nabla^2 \log f_c(\boldsymbol{x}^*)$ and $\mathcal{A}$ is the law of $I \sim \mathcal{U}\{1, \ldots, n\}$. Noting that $\phi_c(\boldsymbol{x})$ in (6.9) can be re-expressed as

$$\phi_c(\boldsymbol{x}) = \frac{1}{2}\left[\alpha_c(\boldsymbol{x})^\intercal \boldsymbol{\Lambda}_c (2\nabla \log f_c(\boldsymbol{x}^*) + \alpha_c(\boldsymbol{x})) + \mathrm{Tr}(\boldsymbol{\Lambda}_c H_c(\boldsymbol{x}))\right] + C^*, \tag{7.65}$$

where $C^* := \frac{1}{2}\left(\nabla \log f_c(\boldsymbol{x}^*)^\intercal \boldsymbol{\Lambda}_c \nabla \log f_c(\boldsymbol{x}^*) + \mathrm{Tr}\big(\boldsymbol{\Lambda}_c \nabla^2 \log f_c(\boldsymbol{x}^*)\big)\right)$, then this leads to the following unbiased estimator for ϕ_c:

$$\tilde{\phi}_c(\boldsymbol{x}) := \frac{1}{2}\left[\alpha_{I,c}(\boldsymbol{x})^\intercal (2\nabla \log f_c(\boldsymbol{x}^*) + \alpha_{J,c}(\boldsymbol{x})) + \mathrm{Tr}\Big(\boldsymbol{\Lambda}_c \tilde{H}_{I,c}(\boldsymbol{x})\Big)\right] + C^*, \tag{7.66}$$

where $I, J \overset{\text{iid}}{\sim} \mathcal{U}\{1, \ldots, m_c\}$, i.e. if now we let $\mathcal{A}$ be the law of $I, J \overset{\text{iid}}{\sim} \mathcal{U}\{1, \ldots, m_c\}$, we have $\mathbb{E}_{\mathcal{A}}\left[\tilde{\phi}_c(\boldsymbol{x})\right] = \phi_c(\boldsymbol{x})$.

The constants $\|\nabla \log f_c(\boldsymbol{x}^*)\|^2$, $\mathrm{Tr}\big(\nabla^2 \log f_c(\boldsymbol{x}^*)\big)$ can be evaluated at $\mathcal{O}(m_c)$ cost, but they only need to be computed once prior to calling Algorithm 7.1.2. The unbiased estimator $\tilde{\phi}_c(\boldsymbol{x})$ uses only double draws from $\{1, \ldots, m_c\}$, although Pollock et al. [2020] notes that it would be possible to average over multiple draws (sampling from $\{1, \ldots, m_c\}$ with replacement) which could reduce the variance of the estimator at the cost of increasing the number of data points to evaluate at.

7.4 Simulation studies

In this section, we study empirically the performance of our Fusion algorithms (Sections 7.1 and 7.2), and selection of tuning parameters (T, n and $\mathcal{P}$ as discussed in Section 7.3) in our two idealised key settings—the SH(λ) setting (Condition 7.3.1) and SSH(γ) setting (Condition 7.3.2) described in Section 7.3. We do this in Sections 7.4.1 and 7.4.2 respectively. For simplicity, here we focus on BF and GBF, noting that GBF is simply D&C-GBF with a *fork-and-join tree* hierarchy (as in Figure 6.1). Note that the earlier BF approach is simply a special case of GBF with $\boldsymbol{\Lambda}_c = \mathbb{I}_d$

for $c \in \mathcal{C}$, and so comparison with this work is straight-forward. Finally, in Section 7.4.3 we compare the performance of Fusion methodologies (including D&C-GMCF (of Section 6.2) and D&C-GBF) with increasing dimensionality. We return in Section 7.5 to consider more substantive examples using real data. To compare the performance of different approaches we consider their computational cost (by computing both the computational run-times and the number of iterations in Algorithm 7.1.2, n, as a proxy for amount of communication between the cores) and *Integrated Absolute Distance (IAD)* defined in (6.25).

Throughout this section we use the GPE-2 estimator of ρ_j as given in Definition 7.1.2 and use the Trapezoidal rule to estimate the mean γ_c in (7.23) and set $\beta_c = 10$ for $c \in \mathcal{C}$. Details on how to find the corresponding scripts to run these experiments is given in Appendix A, and necessary calculations to implement these examples are given in Appendix B.4.

7.4.1 Sub-posterior Homogeneity

We first study the guidance developed for T and $\mathcal{P}$ in Section 7.3 for GBF (Algorithm 7.1.2) in the SH(λ) setting of Condition 7.3.1. Recall, this is the setting in which we are combining homogeneous sub-posteriors, and would naturally arise if a dataset was split randomly across several cores. To study this setting, we consider the idealised scenario of combining $C = 10$ bivariate Gaussian sub-posteriors, with a range of data sizes from $m = 1000$ to $m = 40000$, which have been randomly split across the $C = 10$ cores. In particular, each sub-posterior has mean $\mathbf{0} = (0, 0)$ and variance $\frac{C}{m}\boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$ with $\rho = 0.9$. For this example, we apply both BF and GBF with a fixed particle set size of $N = 10000$.

To examine the guidance for T and $\mathcal{P}$, we consider the effect of different choices for T and $\mathcal{P}$ to CESS_0 and CESS_j for $j = 1, \ldots, n$ with increasing data size. We consider the four following choices of T and $\mathcal{P}$:

1. a fixed choice of T and n to obtain $\mathcal{P}$ (for GBF, $T = 1$ and $n = 5$, and for BF, $T = 0.005$ and $n = 5$),
2. using the recommended T from the guidance in Section 7.3.1 and fixed $n = 5$ to obtain $\mathcal{P}$,
3. using the recommended T and $\mathcal{P}$ using a regular mesh (as outlined in Algorithm 7.3.1),
4. using the recommended T and $\mathcal{P}$ using a adaptive mesh (as outlined in Algorithm 7.3.2).

In implementing the BF and GBF, we set our lower tolerable bounds for the initial (CESS_0) and the iterative (CESS_j) conditional effective sample sizes to be $0.5N$ (i.e. we set $\zeta = \zeta' = 0.5$) and resample if ESS falls below $0.5N$. The procedure for obtaining the recommended T and $\mathcal{P}$ for each approach is outlined in Remark 7.4.1.

Remark 7.4.1. *We set the tuning parameters for BF and GBF (for the SH(λ) setting of Section 7.4.1) as follows:*

1. *Following the guidance outlined in Remark 7.3.2, and with $\zeta = 0.5, \lambda = 1$ and $d = 2$, we have $k_1 = \sqrt{-\frac{(\lambda+\frac{d}{2})}{\log(\zeta)}} \approx 1.7$. For GBF, $\mathbf{\Lambda}_c$ is the estimated covariance matrices for sub-posterior $c = 1, \ldots, C$, so $b = \frac{m}{C}$ (see Remark 7.3.1), and we choose $T = C^{\frac{1}{2}} k_1$. For BF, $\mathbf{\Lambda}_c = \mathbb{I}_d$ for $c = 1, \ldots, C$, so we have $b = 1$ and so we choose $T = C^{3/2} k_1 / m$.*

2. *When using the regular mesh, we use Algorithm 7.3.1 to obtain $\mathcal{P}$. First let $\zeta' = 0.5$ then for GBF we have $b = \frac{m}{C}$, and so $\Delta_j = \Delta = \sqrt{\frac{k_4}{2Cd}}$ for each j, where k_4 is computed as per (7.53) and computing an estimate of the supremum of $\widehat{\mathbb{E}[\nu_j]}$ as per (7.50). For BF, $b = 1$, so $\Delta_j = \Delta = \sqrt{\frac{Ck_4}{2m^2 d}}$ for each j.*

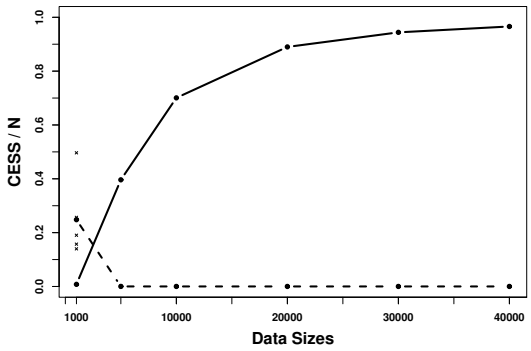
3. *When using the adaptive mesh, we use Algorithm 7.3.2 to obtain Δ_j recursively at each iteration to construct $\mathcal{P}$. We let $\zeta' = 0.5$ and for GBF (where $b = \frac{m}{C}$) we compute $t_j = \min\{T, t_{j-1} + \Delta_j\}$ where $\Delta_j = \sqrt{\frac{k_4}{2Cd}}$ at each iteration of Algorithm 7.1.2, until we have $t_j = T$. For the standard BF approach, note that $b = 1$ so we must compute $\Delta_j = \sqrt{\frac{Ck_4}{2m^2 d}}$ instead at each iteration.*

The conditional effective sample size of the GBF and (standard) BF approaches with increasing data size in this SH(λ) setting are shown in Figure 7.1. First considering the results from fixing T and n in Figure 7.1a, we can see that BF lacks robustness with increasing data size. Here CESS_0 improves with increasing data size (m), which is due to the sub-posteriors becoming increasingly similar with m in this idealised scenario. However, as we increase m the fixed choice for T (and hence the size of the intervals) becomes increasingly inappropriate for the sub-posteriors, which leads to a degradation in average CESS_j. In contrast, GBF incorporates global information about the sub-posteriors (i.e. the variance of the sub-posteriors), so there is no change in performance with m. Note there is a trade-off with the choice of T: a small T leads to poor behaviour on initialisation (i.e. low CESS_0), but good behaviour at each iteration (i.e. high average CESS_j).

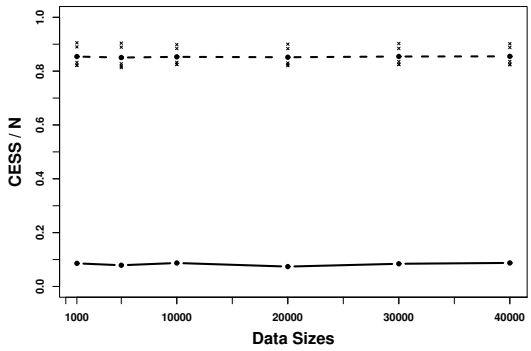
Considering Figure 7.1b, we see that scaling T following the guidance in Section 7.3.1 immediately stabilises CESS_0, although CESS_j performance is still poor (n is too small). In Figure 7.1c and Figure 7.1d, we see that utilising both the guidance for T and the mesh $\mathcal{P}$ drastically improves the performance of both BF and GBF. In both cases GBF outperforms BF: it achieves higher average CESS_j, and the variance of CESS_j is lower. Given BF is a special case of GBF, this improvement can be ascribed to the use of estimated covariance matrices for $\mathbf{\Lambda}_c$. In particular, this choice leads to a lower variance unbiased estimator for ρ_j, and an improved proposal h^{bf} (7.17) for g^{bf} (7.18).

From Figure 7.1e we see that with BF that without our guidance on T and $\mathcal{P}$, average IAD is poor, and the variance of the IAD is very large. In contrast, GBF with our guidance is robust

across the different scenarios. Comparing the regular and adaptive meshes simply using CESS_0 and CESS_j would imply that the regular mesh is performing better (since it has slightly better CESS_j), however the adaptive mesh is slightly more computationally efficient as shown by having a smaller mesh size, n, (illustrated in Figure 7.1f) and having a faster algorithm run-time (illustrated in Figure 7.1g). By looking at the IAD obtained for these approaches, we can see that we are able to obtain similar performance at a lower cost with the adaptive mesh construction.

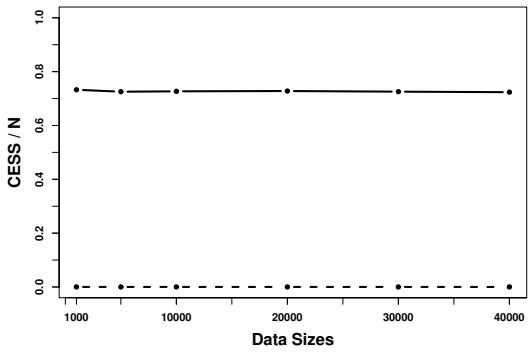


(i) BF

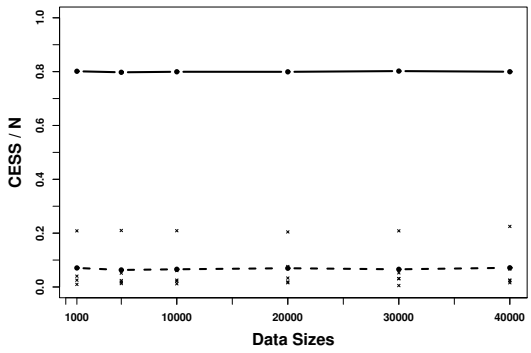


(ii) GBF

(a) Fixed user-specified T and n.



(i) BF

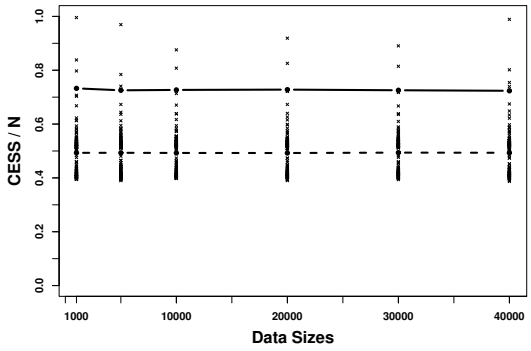


(ii) GBF

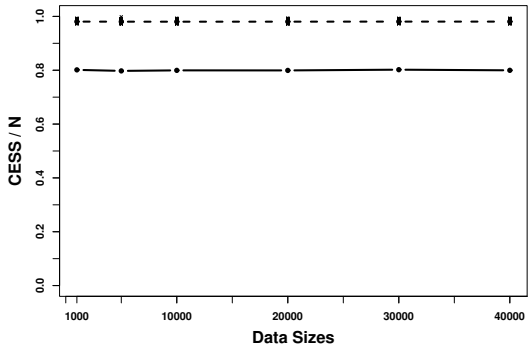
(b) Recommended T and fixed n.

7.4.2 Sub-posterior Heterogeneity

Now we study the guidance for T and $\mathcal{P}$ for GBF (Algorithm 7.1.2) developed in Section 7.3 in the SSH(γ) setting of Condition 7.3.2. This represents the setting where sub-posterior heterogeneity does not decay with data size. Here, we consider the scenario of combining $C = 2$ bivariate Gaussian sub-posteriors, $f_c \sim \mathcal{N}\left(\boldsymbol{\mu}_c, \frac{2}{m}\boldsymbol{\Sigma}\right)$, where $\boldsymbol{\mu}_1 = -(0.25, 0.25)$ and $\boldsymbol{\mu}_2 = (0.25, 0.25)$ and $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$, with $\rho = 0.9$. We again consider a range of data sizes, which ranges from $m = 250$ to $m = 2500$ and are randomly split between $C = 2$ cores. We apply BF and GBF with a fixed particle set size of

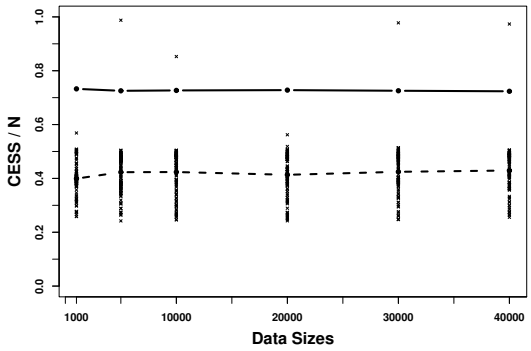


(i) BF

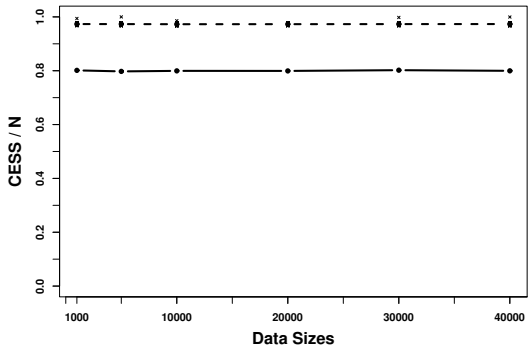


(ii) GBF

(c) Recommended T and recommended regular mesh $\mathcal{P}$.

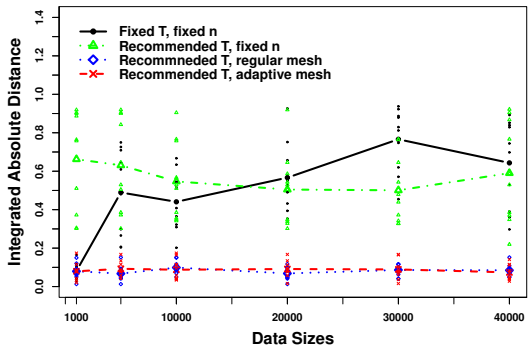


(i) BF

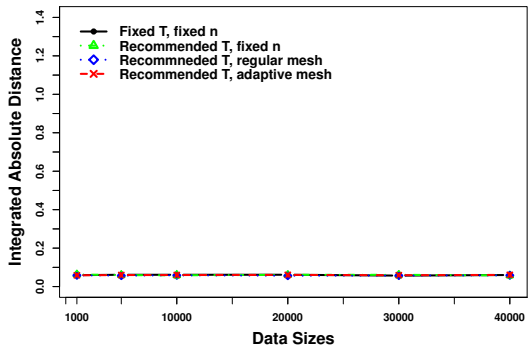


(ii) GBF

(d) Recommended T and recommended adaptive mesh $\mathcal{P}$.

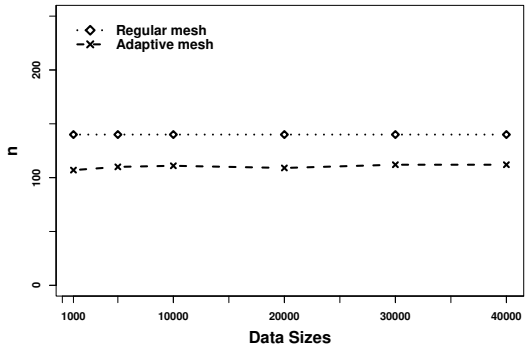


(i) BF

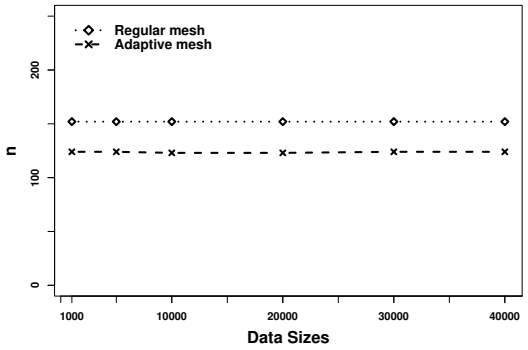


(ii) GBF

(e) Integrated absolute distance: lines connect the mean IAD (averaged over ten runs) while the points denote the individual IAD achieved on each run.

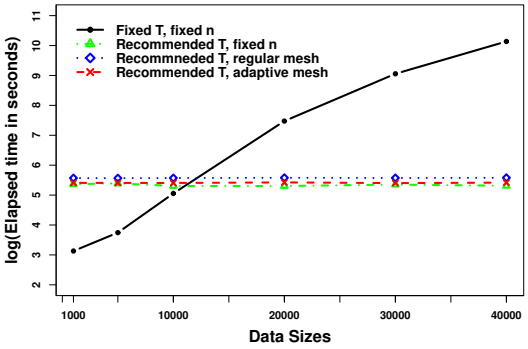


(i) BF

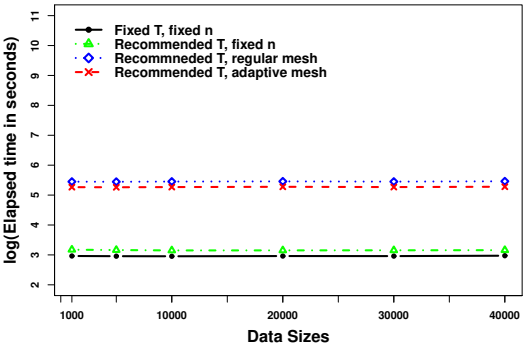


(ii) GBF

(f) Comparison of mesh sizes between regular and adaptive schemes.



(i) BF



(ii) GBF

(g) Mean average computational run-times (based on ten runs).

Figure 7.1: Bivariate Gaussian example in SH(λ) setting with increasing data size. In Figures 7.1a, 7.1b, 7.1c, 7.1d solid lines denote initial CESS (CESS_0), and dotted lines denote averaged CESS in subsequent iterations ($\frac{1}{n}\sum_{j=1}^{n}\text{CESS}_j$), and crosses denote CESS_j for each $j = 1, \ldots, n$.

$N = 10000$. In this setting as m increases the sub-posterior heterogeneity increases since the sub-posteriors have diminishing overlapping support. In BF (where $\mathbf{\Lambda}_1 = \mathbf{\Lambda}_2 = \mathbb{I}_d$) this heterogeneity is not captured, and $\sigma^2_{\boldsymbol{a}} = 0.125$ irrespective of m. By contrast, our generalised approach is able to capture the heterogeneity with m with the inclusion of the estimated covariance matrices $\{\mathbf{\Lambda}_c\}_{c=1,2}$.

As with the previous example in Section 7.4.1, we will investigate the effect of varying T and $\mathcal{P}$ with m, and its impact upon CESS_0 and CESS_j. We consider the following choices for T and $\mathcal{P}$:

1. a fixed choice of T and n to obtain $\mathcal{P}$ (for GBF, $T = 2$ and $n = 5$, and for BF, $T = 0.01$ and $n = 5$),

2. using the recommended T from the guidance in Section 7.3.1 and fixed $n = 5$ to obtain $\mathcal{P}$,

3. using the recommended T and $\mathcal{P}$ using a regular mesh (as outlined in Algorithm 7.3.1),

4. using the recommended T and $\mathcal{P}$ using a adaptive mesh (as outlined in Algorithm 7.3.2).

When applying the guidance, we set the lower tolerable bounds on the initial (CESS_0) and iterative (CESS_j) conditional effective sample sizes to be $0.5N$ (i.e. we set $\zeta = \zeta' = 0.5$) and resample if the ESS drops below $0.5N$. The procedure for obtaining the recommended T and $\mathcal{P}$ in each approach is outlined in Remark 7.4.2.

Remark 7.4.2. *We set the tuning parameters for BF and GBF (for the SSH(γ) setting of Section 7.4.2) as follows:*

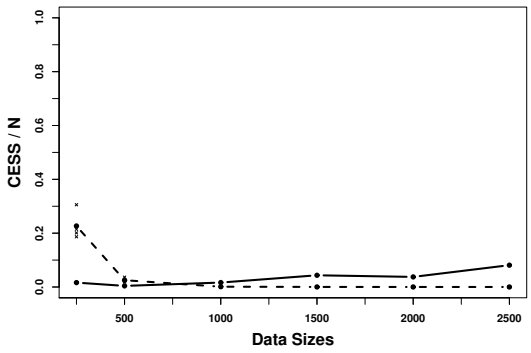
1. *We follow the guidance outlined in Remark 7.3.2, noting that $\zeta = 0.5$ and $d = 2$. For GBF, $\mathbf{\Lambda}_{c=1,2}$ are the estimated covariance matrices for each of the sub-posteriors, so $b = \frac{m}{C}$ (see Remark 7.3.1), and $\gamma = m\sigma^2_{\boldsymbol{a}}/C$ (where $\sigma^2_{\boldsymbol{a}}$ is estimated from the sub-posterior samples). Consequently, we can compute $k_1 = k_2 = \sqrt{-\left(\frac{\gamma m}{C} + \frac{d}{2}\right)/\log(\zeta)}$, and choose $T = C^{\frac{1}{2}}k_1$. For BF, $\mathbf{\Lambda}_{c=1,2} = \mathbb{I}_d$, so $b = 1$ and $\gamma = \sigma^2_{\boldsymbol{a}}$, and so we can compute $k_1 = \sqrt{-\left(\frac{\gamma m}{C} + \frac{d}{2}\right)/\log(\zeta)}$ and $k_2 = \frac{Ck_1}{m}$, and choose $T = \frac{C^{3/2}k_1}{m} = C^{\frac{1}{2}}k_2$.*

2. *When using the regular mesh, we use Algorithm 7.3.1 to obtain $\mathcal{P}$. As $\zeta' = 0.5$, we have for GBF $b = \frac{m}{C}$, and so $\Delta_j = \Delta = \sqrt{\frac{k_4}{2Cd}}$ for each j where k_4 is computed as per (7.53) (with $\sup_j \widehat{\mathbb{E}[\nu_j]}$ computed as per (7.50)). For BF we have $b = 1$, so $\Delta_j = \Delta = \sqrt{\frac{Ck_4}{2m^2d}}$ for each j.*

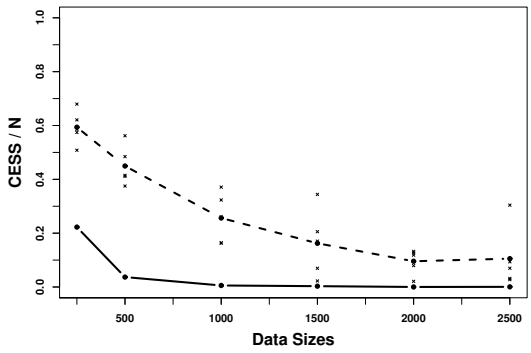
3. *When using the adaptive mesh, we use Algorithm 7.3.2 to obtain Δ_j recursively at each iteration to construct $\mathcal{P}$. With $\zeta' = 0.5$ for the GBF (where $b = \frac{m}{C}$) we compute $t_j = \min\{T, t_{j-1} + \Delta_j\}$ where $\Delta_j = \sqrt{\frac{k_4}{2Cd}}$ at each iteration of Algorithm 7.1.2 until we have $t_j = T$. For BF with $b = 1$ we have instead $\Delta_j = \sqrt{\frac{Ck_4}{2m^2d}}$ at each iteration.*

CESS for BF and GBF with increasing m in this SSH(γ) setting are shown in Figure 7.2. We can immediately see that the SSH(γ) setting is much more challenging than the idealised SH(λ) setting of Section 7.4.1, which is unsurprising as in this case the sub-posteriors are becoming increasingly mismatched as m increases. In Figure 7.2a, we see that fixing T and n is not ideal for either method. As shown in Figure 7.2b, there is a positive effect for both BF and GBF in using our recommended scaling of T in the quality of the initialisation. In Figure 7.2c and Figure 7.2d, where both the guidance for T and $\mathcal{P}$ are implemented, we see a substantial improvement in the performance of both approaches with respect to CESS, with our new GBF approach outperforming BF.

In Figure 7.2c we see that the use of a regular mesh in choosing $\mathcal{P}$, following our guidance, provides robust CESS$_j$ with low variance. Indeed, it appears to outperform the adaptive mesh approach for $\mathcal{P}$ (see Figure 7.2d). However, as discussed in Section 7.3.2.2, the regular mesh is overly conservative, and when we factor in the reduced number of iterations required in the adaptive case (Figure 7.2f), along with the overall reduction in computational cost (Figure 7.2g) for comparable IAD (Figure 7.2e), we see that the use of an adaptive mesh is preferable.

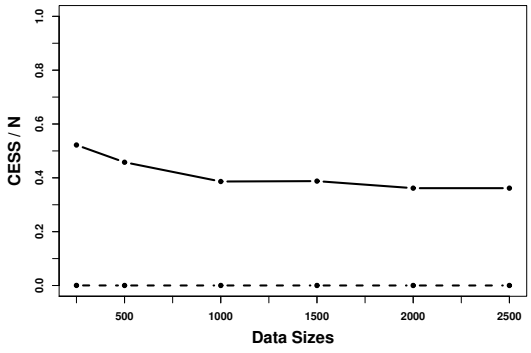


(i) BF

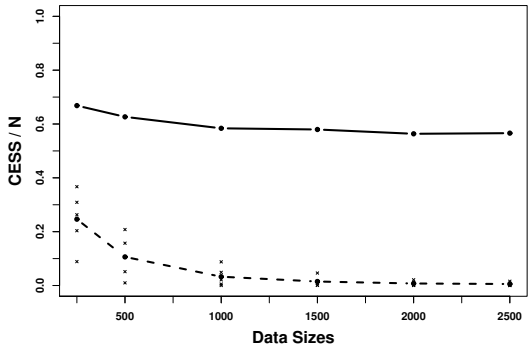


(ii) GBF

(a) Fixed user-specified T and n.

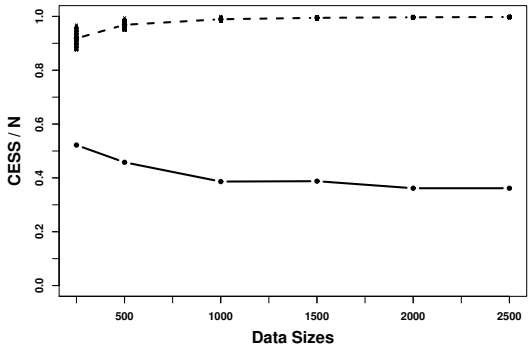


(i) BF

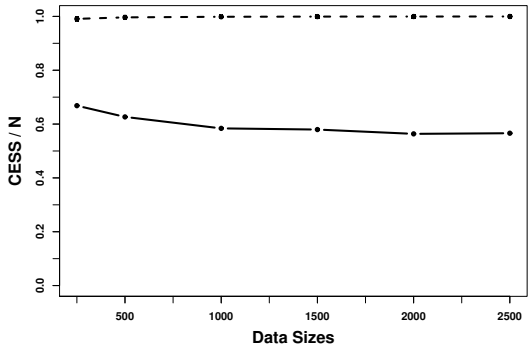


(ii) GBF

(b) Recommended T and fixed n.

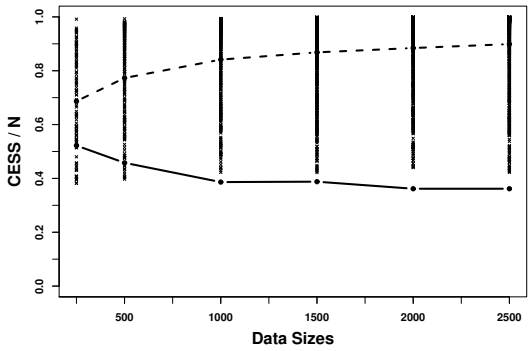


(i) BF

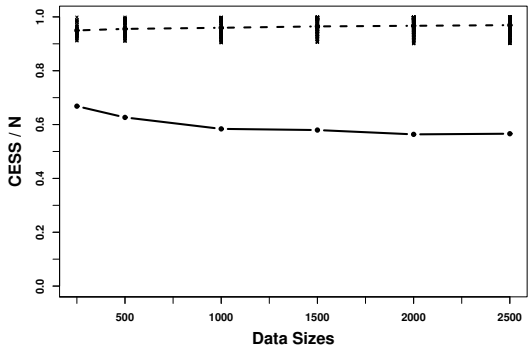


(ii) GBF

(c) Recommended T and recommended regular mesh $\mathcal{P}$.

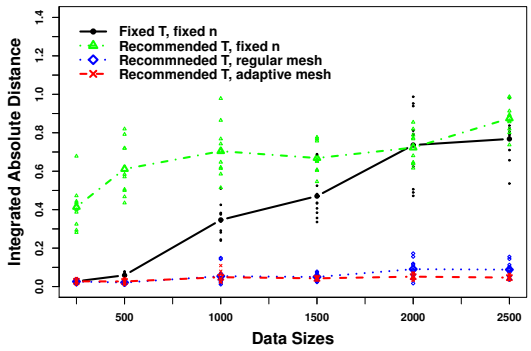


(i) BF

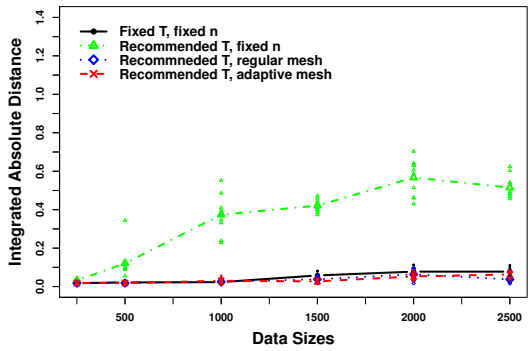


(ii) GBF

(d) Recommended T and recommended adaptive mesh $\mathcal{P}$.

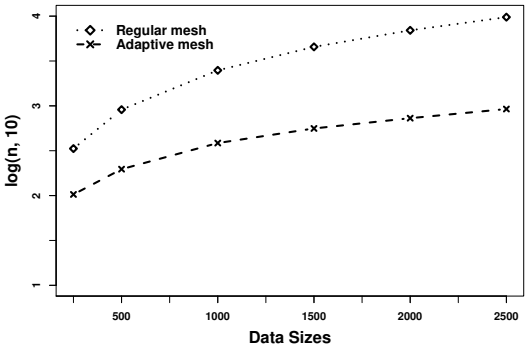


(i) BF

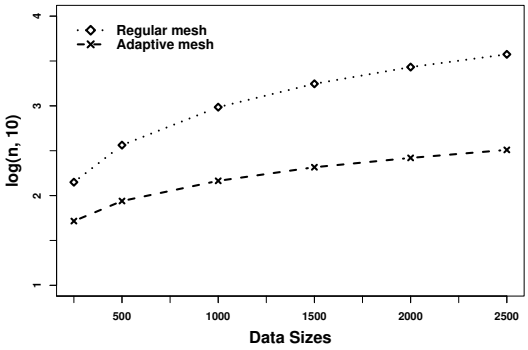


(ii) GBF

(e) Integrated absolute distance: lines connect the mean IAD (averaged over ten runs) while the points denote the individual IAD achieved on each run.

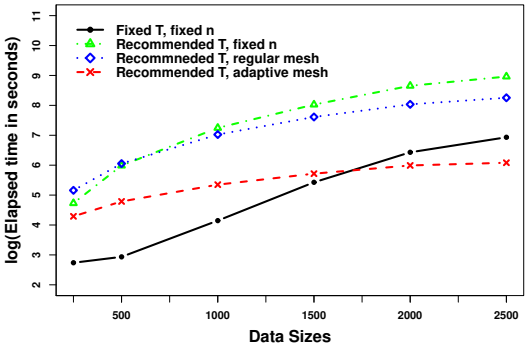


(i) BF

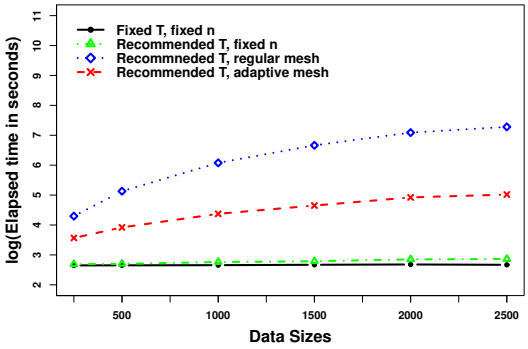


(ii) GBF

(f) Comparison of mesh sizes between regular and adaptive schemes.



(i) BF



(ii) GBF

(g) Mean average computational run-times (based on ten runs).

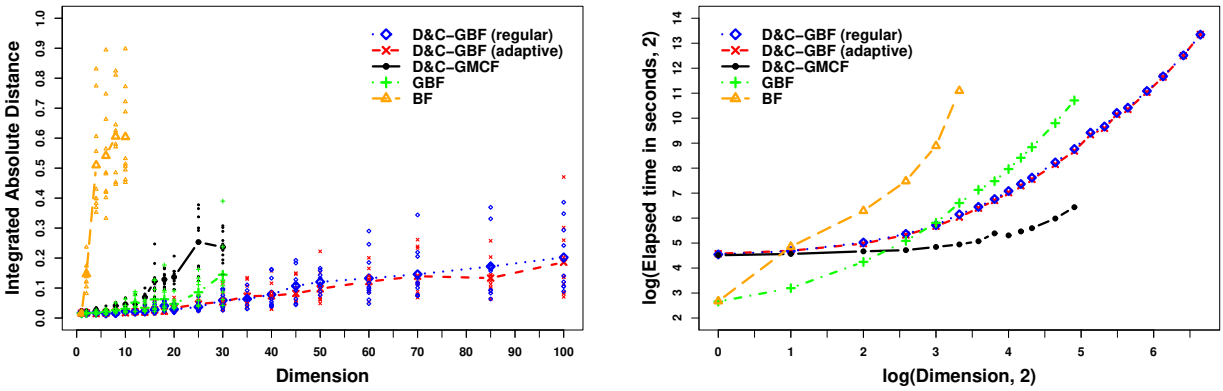
Figure 7.2: Bivariate Gaussian example in SSH(γ) setting with increasing data size. In Figures 7.2a, 7.2b, 7.2c, 7.2d solid lines denote initial CESS (CESS_0), and dotted lines denote averaged CESS in subsequent iterations ($\frac{1}{n}\sum_{j=1}^{n}\text{CESS}_j$), and crosses denote CESS_j for each $j = 1, \ldots, n$.

7.4.3 Dimension study

In this section, we empirically study the performance of Fusion approaches (BF, GBF, D&C-GMCF and D&C-GBF) with increasing dimensionality. To do so we consider a d-dimensional multivariate Gaussian $f \propto \prod_{c=1}^{C} f_c$, where we let $C = 8$ and $f_c \sim \mathcal{N}_d(\mathbf{0}, C\boldsymbol{\Sigma})$, and where

$$\begin{aligned} \Sigma_{ii} &= 1, \qquad \text{for all } i = 1, \ldots, d, \\ \Sigma_{ij} &= 0.9, \qquad \text{for all } i \neq j,\ (i,j) \in \{1, \ldots, d\}, \end{aligned}$$

and simply vary d. For BF and GBF we use an adaptive mesh for $\mathcal{P}$, and for D&C-GBF we consider both a regular and adaptive mesh for $\mathcal{P}$ with a balanced-binary tree hierarchy. In all cases we use the guidance developed in Section 7.3. As we are in the SH(λ) setting (the true sub-posterior means are the same), we set $\lambda = 1$. The lower bounds of the tolerable initial and iterative CESS are set to $0.05N$ (i.e. $\zeta = \zeta' = 0.05$) and we resample if the ESS drops below $0.5N$, where $N = 10000$.



(a) Integrated Absolute Distance.

(b) Computational cost.

Figure 7.3: Comparison of Fusion methodologies with increasing dimensionality (in the setting of Section 7.4.3). In Figure 7.3a, lines connect the mean IAD (averaged over ten runs) while the points denote the individual IAD achieved on each run.

From Figure 7.3a, the performance of all Fusion methods degrades as we increase the dimensionality both in terms of the average IAD and also the variance. Since our target exhibits large correlation between the components, BF (recall this is simply the GBF approach with $\boldsymbol{\Lambda}_c = \mathbb{I}_d$ for all $c = 1, \ldots, C$) struggles even in low dimensions for this particular problem, whereas our generalised approaches which we have developed in this thesis offer a much better scaling with dimensionality. D&C-GBF comfortably outperforms existing Fusion approaches for even moderate dimensionality in terms of IAD and computational cost.

7.5 Examples

In this section, we consider a number of models applied to a variety datasets, and suppose the dataset is randomly split into C (disjoint) subsets. We compare (in terms of computational runtime and IAD) the performance of our Fusion methodologies (GBF, D&C-GMCF and D&C-GBF) with other established (approximate) methodologies. As a benchmark for the target f we use `Stan` [Carpenter et al., 2017] to implement an MCMC sampler for the target posterior distribution using the full dataset. In implementing Fusion methodologies, we use the GPE-2 variants of Algorithm 2c and Algorithm 7.1.1 as before. Our implementations for GBF and D&C-GBF are as presented in Sections 7.1 and 7.2, following the guidance presented in Section 7.3 (but without the inclusion of any adaptions such as those presented in Section 7.3.3). In implementing D&C-GBF we use the balanced-binary tree hierarchy.

The approximate methodologies we contrast our implementation against are *Consensus Monte Carlo (CMC)* [Scott et al., 2016], the kernel density averaging approach of Neiswanger et al. [2014] (which we term *KDEMC*), and the *Weierstrass Sampler (WRS)* [Wang and Dunson, 2013]. For full details on where to find the corresponding code/scripts to implement these examples, see Appendix A. Furthermore, in Appendix B, we supply details of calculations required to implement all examples found in this section.

7.5.1 Robust regression

In this section, we consider the *'Combined Cycle Power Plant'* dataset available from the *UCI Machine Learning Repository* [Kaya et al., 2012; Tüfekci, 2014]. The dataset comprises $m = 9568$ records of the **net hourly electrical output** of a combined cycle power plant over 6 years between 2006 and 2011, together with four (hourly averaged) *ambient* variables: **temperature**; **ambient-pressure**; **relative-humidity**; and, **exhaust-vacuum**.

To model electrical output using the ambient variables, we use a robust regression model:

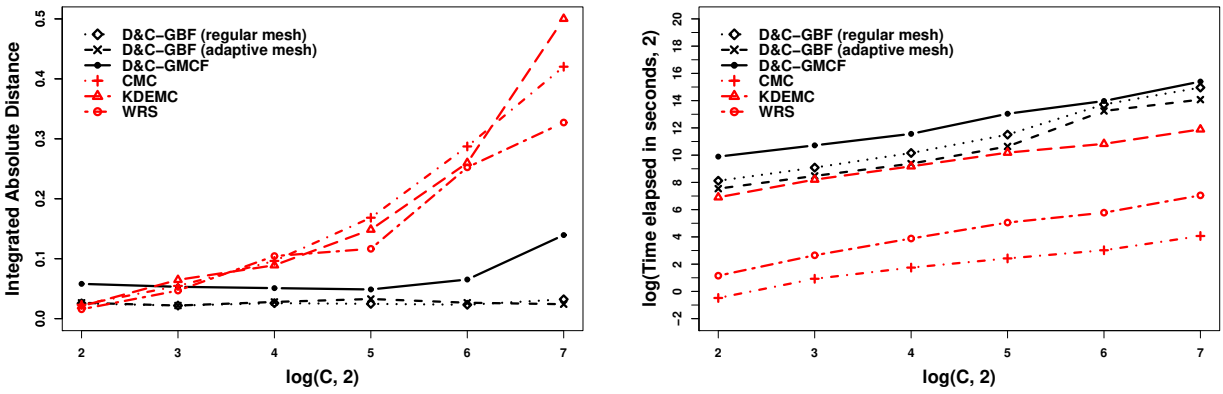
$$\begin{aligned} y_i &\sim \mathrm{t}(\nu, X_i\boldsymbol{\beta}, \sigma), \quad i = 1, \ldots, n, \\ \beta_{j'} &\sim \mathcal{N}_1\left(\mu_{\beta_{j'}}, \sigma^2_{\beta_{j'}}\right), \quad j' = 0, \ldots, p, \end{aligned}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ is the dependent variable (electrical output), $X \in \mathbb{R}^{n\times(p+1)}$ is the design matrix, $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ is the vector of predictor (ambient) variables which we want to perform inference on. For simplicity, we assume that ν, σ, $\mu_{j'}$ and $\sigma^2_{\beta_{j'}}$ for $j' = 0, \ldots, p$ are known.

For our dataset $p = 4$, and so $d = 5$. We consider $C \in \{4, 8, 16, 32, 64, 128\}$ cores, each of which is assigned a random split of the data. We use `Stan` to sample the sub-posteriors (with $\mu_{j'} = 0$ and $\sigma^2_{\beta_{j'}} = 10C$ for $j' = 0, \ldots, p$), which we will attempt to unify as in (1.1). We use the approximate CMC, KDEMC and WRS approaches to do this, together with our Fusion approaches. For D&C-

GMCF, we set $T = 1$ and $N = 100000$, whereas for D&C-GBF, we will set $N = 10000$. We have a different number of particles N for these two approaches so that the computational cost of each Fusion method is comparable. In implementing D&C-GBF we set $\zeta = 0.5$, $\zeta' = 0.05$, and consider both the regular and adaptive mesh variants of the temporal partition, $\mathcal{P}$. For each Fusion method, we resample if ESS falls below $0.5N$. The results are presented in Figure 7.4.

Figure 7.4a clearly shows that of all the approaches considered, D&C-GBF provides the highest quality and most reliable sample approximation for f, and is the most robust to increasing C. For a comparable computational cost, D&C-GMCF does not perform as well as D&C-GBF and we can observe a drop in performance for larger C. However, D&C-GMCF is still more scalable with respect to C over the approximate methods here. Although more computationally intensive, D&C-GBF has a cost which grows at the same rate as the approximate methodologies considered. Of the variants of D&C-GBF considered, the adaptive mesh outperforms the regular mesh.



(a) Integrated Absolute Distance.

(b) Computational cost.

Figure 7.4: Comparison of competing methodologies to Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF) applied to a robust regression problem with power plant dataset (in the setting of Section 7.5.1).

7.5.2 Negative Binomial regression

Here we consider the *'Bike Sharing'* dataset available on the *UCI Machine Learning Repository* [Fanaee-T and Gama, 2014]. The dataset contains $m = 17379$ records of the **total count of bikes on rental each hour**, together with seven variables: **seasonality** (a categorical variable with four levels: spring, summer, autumn, winter); **weekend** (binary, taking value 1 if a weekend, and 0 if not); **holiday**; (binary, taking value 1 if a holiday, and 0 if not); **rush-hour** (binary, taking value 1 if recorded on a weekday between 7AM-9AM or 4PM-7PM, and 0 if not); **weather** (binary, taking value 1 if 'clear', and 0 if not); **temperature** (continuous); and, **wind-speed** (continuous). For the purposes of our work, we replaced the variable **seasonality** with three binary variables codifying the four levels.

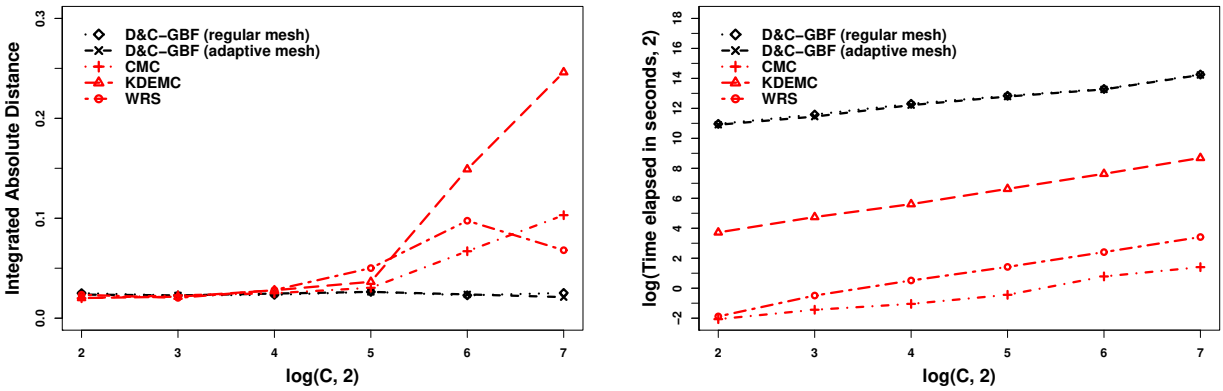
To model the total count of bikes on rental, we use the following Negative binomial regression model with Gaussian priors on the regression parameters:

$$\begin{aligned} y_i &\sim \text{NB}(\mu_i, r), \qquad \text{where } \log(\mu_i) = X_i\boldsymbol{\beta}, \qquad i = 1, \ldots, n, \\ \beta_{j'} &\sim \mathcal{N}_1\left(\mu_{\beta_{j'}}, \sigma^2_{\beta_{j'}}\right), \qquad j' = 0, \ldots, p, \end{aligned}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ is our total count of bikes on rental, $X \in \mathbb{R}^{n\times(p+1)}$ is the design matrix, $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ is the vector of predictor variables. For simplicity, r, $\mu_{\beta_{j'}}$, $\sigma^2_{\beta_{j'}}$ for $j' = 0, \ldots, p$ are assumed known.

For this dataset $p = 9$, and so $d = 10$. As in Section 7.5.1, we split the dataset amongst $C \in \{4, 8, 16, 32, 64, 128\}$ cores, and use `Stan` with $\mu_{j'} = 0$ and $\sigma^2_{\beta_{j'}} = 10C$ for $j' = 0, \ldots, p$, to recover the respective sub-posteriors. To implement D&C-GBF we set $N = 10000$, $\zeta = 0.2$, $\zeta' = 0.05$, and consider both regular and adaptive mesh variants of $\mathcal{P}$, and resample if the ESS drops below $0.5N$.

The results in Figure 7.5 again show that, when contrasted with existing (approximate) approaches, D&C-GBF provides the most accurate sample approximation, and is robust and consistent with increasing C.



(a) Integrated Absolute Distance.

(b) Computational cost.

Figure 7.5: Comparison of competing methodologies to Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF) applied to a Negative Binomial regression problem with bike sharing dataset (in the setting of Section 7.5.2).

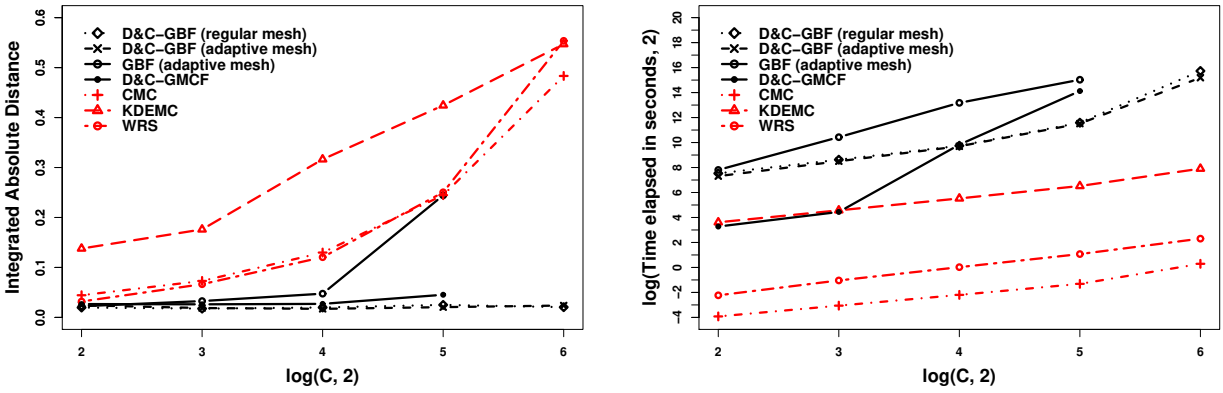
7.5.3 Logistic regression

In this section, we consider recovering posterior distribution of a logistic regression model (6.27) with standard Gaussian priors applied to a number of different datasets. In Section 7.5.3.1, we revisit the simulated dataset from Section 6.4.1 and in Sections 7.5.3.2 and 7.5.3.3, we consider a smart grid dataset and NYC airport flight dataset, respectively.

7.5.3.1 Simulated data

In this example, we revisit the simulated dataset we introduced in Section 6.4.1. Recall, we have a dataset simulated from a logistic regression model with $m = 1000$ records and the dimensionality of the problem is $d = 5$. To conduct Fusion we first equally split the data between $C \in \{4, 8, 16, 32, 64\}$ cores and use `Stan` to sample from the logistic regression model with Gaussian prior distributions with mean 0 and variance C on each parameter to find a sample approximation of each sub-posterior. Together with the approximate methodologies, we implemented our D&C-GBF approach with $N = 10000$, $\zeta = 0.2$, $\zeta' = 0.05$, and both regular and adaptive temporal partition meshes. Here, we also consider applying Generalised Bayesian Fusion (GBF) (i.e. directly applying Algorithm 7.1.2 with $\mathcal{C} := \{1, \ldots, C\}$ which is equivalent to D&C-GBF within a fork-and-join tree hierarchy, as per Figure 6.1). We again plot the results from Section 6.4.1 and further include the results from applying our GBF and D&C-GBF methodologies in Figure 7.6.

Considering Figure 7.6a, we see again that D&C-GBF achieves the best sample approximation, and the quality of the sample approximation is robust to increasing C. Note that our divide-and-conquer framework offers significant gains, with D&C-GBF outperforming GBF in terms of robustness with C (even with the same tuning parameter guidance being followed). Compared with D&C-GMCF, D&C-GBF offers a better IAD performance whilst having a computational cost which looks to scale better (although it is slightly more expensive for lower number of sub-posteriors). Note that CMC outperforms all other approximate methodologies, which leaves the practitioner with a clear decision: if a cheap but approximate methodology is needed use CMC, but if accuracy is the goal then D&C-GBF should be used.



(a) Integrated Absolute Distance.

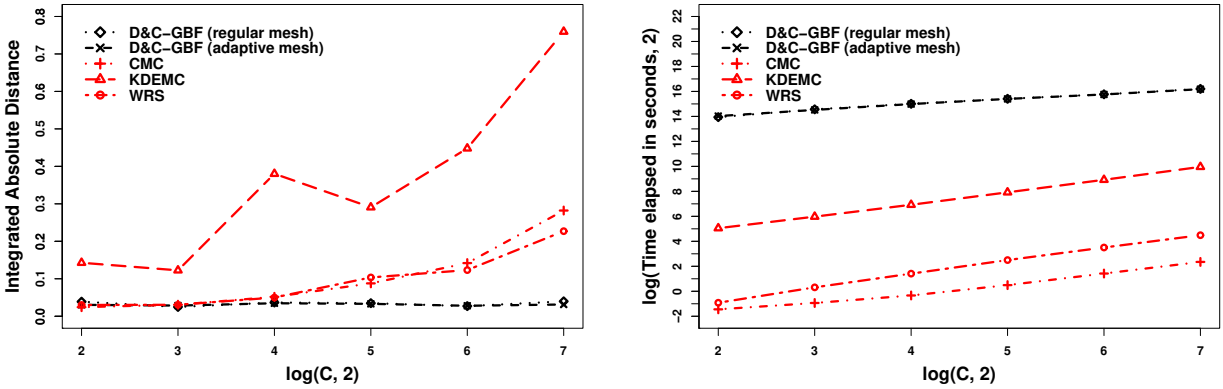
(b) Computational cost.

Figure 7.6: Comparison of competing methodologies to Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF) applied to a logistic regression problem with simulated data (in the setting of Section 7.5.3.1).

7.5.3.2 Smart grid stability data

We consider the *'Smart Grid Stability'* dataset available on the *Kaggle* [Schäfer et al., 2016] which is a simulated dataset consisting of $m = 60000$ records of **smart grid stability** (taking value 1 if stable, and 0 otherwise). We model this binary outcome with $p = 12$ predictors (so $d = 13$). We again split the dataset equally between $C \in \{4, 8, 16, 32, 64, 128\}$ cores, and use `Stan` together with a Gaussian prior distributions with mean 0 and variance C on each of the parameters, to arrive at our C sub-posteriors. D&C-GBF is implemented using a balanced-binary tree with $N = 10000$, $\zeta = 0.2$ and $\zeta' = 0.05$. As with previous examples, we also implement CMC, KDEMC and WRS and compare the performances of each approach. The results are shown in Figure 7.7.

We can see that D&C-GBF achieves the best IAD of all methodologies, and is robust to increasing number of sub-posteriors C. Comparing to the approximate methods, our Fusion approach comes at a higher computational cost, but this seems to scale slightly better in this example when compared to the other methodologies considered.



(a) Integrated Absolute Distance.

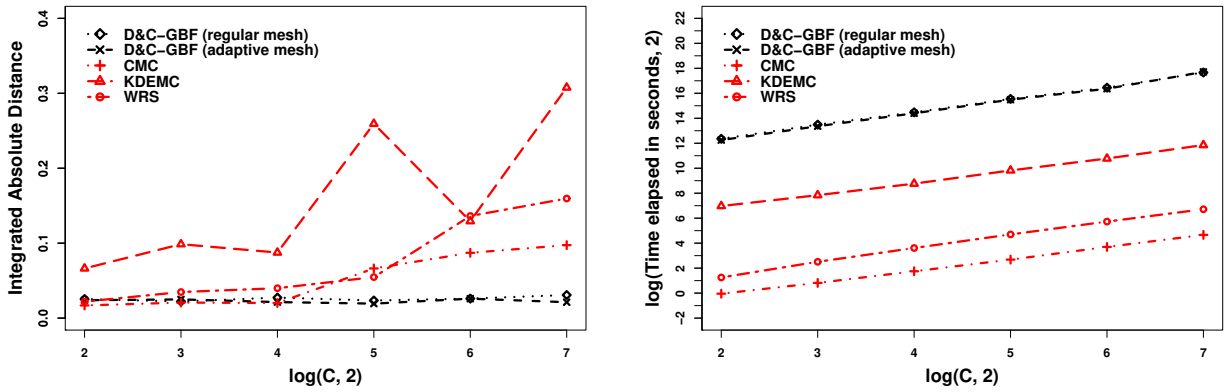
(b) Computational cost.

Figure 7.7: Comparison of competing methodologies to Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF) applied to a logistic regression problem with smart grid dataset (in the setting of Section 7.5.3.2).

7.5.3.3 NYC flights data

Finally, we consider the `nycflights13` dataset (available via the `nycflights13` **R** package on CRAN [Wickham, 2021]). In this study we predict on-time arrival of airplanes, by creating binary observations for **arrival-delay** (taking the value 1 if the flight arrived 1 minute or more late, and 0 otherwise). We model this using $p = 20$ predictor variables (so $d = 21$). After removing any entries with `NA` values, in total the dataset was of size $m = 327346$. This dataset was split randomly across $C \in \{4, 8, 16, 32, 64, 128\}$ cores, and we used `Stan` to find sample approximations of each sub-posterior (using Gaussian priors with mean 0 and variance C for each parameter). D&C-GBF was implemented with $N = 30000$, $\zeta = 0.2$ and $\zeta' = 0.05$. The results are shown in Figure 7.8.

Results for this example are in-line with those earlier: D&C-GBF provides the best sample approximation, is robust to increasing C, but comes at the expense of increased computational cost. In such large data settings, additional modifications (such as the ones explored in Section 7.3.3) could be explored for greater scalability. Although approximate methodologies have been specifically developed to tackle Bayesian big-data problems, here we see that they struggle to recover f even in this idealised scenario. They additionally (and critically) lack robustness when scaled with C.



(a) Integrated Absolute Distance.

(b) Computational cost.

Figure 7.8: Comparison of competing methodologies to Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF) applied to a logistic regression problem with `nycflights13` dataset (in the setting of Section 7.5.3.3).

To further compare the methodologies, we consider fixing $C = 64$ and varying the computational budget for each method by varying the sample size N in order to study the effect of increased computation on IAD performance. We observe the performance of each method by computing the IAD against the same benchmark for the target f that was used above (based upon $N = 30000$ samples using `Stan`). Since the IAD of the approximate methodologies typically had large variance, we run these methods 10 times and take an average as they are relatively inexpensive to run. We additionally plot the minimum and maximum IAD achieved in the 10 runs as to show the variance in performance at different sample sizes. To save on computation, we decide to stop performing replicates of an algorithm approximately at the point when performing the method once would take over an hour, or if it appears that the average IAD performance of the method is not changing much with more computation. As such, for CMC and KDEMC, we considered a wide range of sample sizes $N = 500$ to $N = 200000$ but for KDEMC, we only took a range from $N = 500$ to $N = 50000$. For D&C-GBF, we considered $N = 500$ to $N = 30000$ which we did not do replicate runs of due to the computational budget that such experiment would require. However, we expect the error to decrease with computation as the Monte Carlo error of our methodology should decrease with N.

In Figure 7.9, we plot the IAD performance of each method against the computational run-time of the algorithms. For CMC and WRS, the average and variance of the IAD decreases with

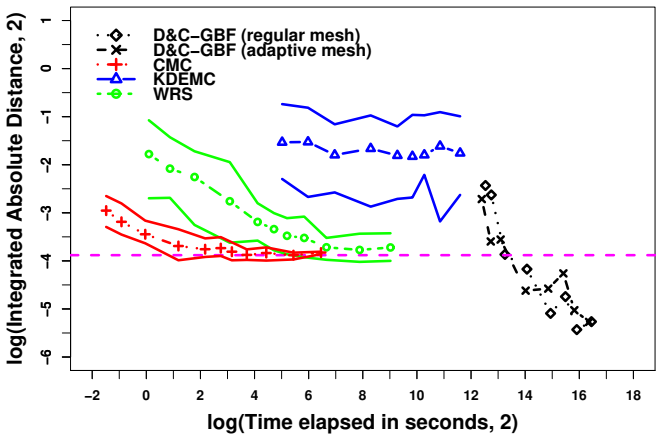


Figure 7.9: Integrated absolute distance against computational budget for competing methodologies to Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF) applied to a logistic regression problem with `nycflight` with fixing $C = 64$ (in the setting of Section 7.5.3.3).

more computation, but both methods quickly reach a point where IAD no longer decreases. We additionally plot a pink dashed line which is the minimum mean value IAD achieved for CMC, as this seems to the point which the IAD of CMC converges to. Neither the average nor the variance IAD for KDEMC decrease here. Therefore, in this example, CMC offered the best performance out of the approximate methods at a lower computational cost.

If accuracy is important, we can see that Fusion should be used as each approximate methodologies reach a point where performance does not increase with greater computation. It is clear in this instance that if a cheap methodology is needed (possibly due to a lack of computational budget or lack of time), CMC performs the best, but if accuracy is important, then our D&C-GBF methodology should be employed. However, it is important to note that this is a setting which is relatively favourable to CMC as the problem is not too far from Gaussian in part due to the reasonably large number of data points on each core, $m/64 \approx 5000$, whereas we have seen several instances earlier in this section where all approximate methodologies performed poorly (e.g. in the small data setting of Section 7.5.3.1). The disadvantage of using approximate methodologies is needing to understand the biases being imposed, which can be hard to do in practical settings.

Chapter 8

Concluding Remarks

The Fusion approach to unifying sub-posteriors into a coherent sample approximation of the posterior (as in (1.1)), offers fundamental advantages over approximation based approaches. In particular, Fusion avoids having to impose any distributional approximation on the sub-posteriors, meaning it is more robust to a wider range of models, and circumvents needing to understand the impact of imposed approximations on the unified posterior. To date, Fusion approaches (such as the Monte Carlo Fusion (MCF) [Dai et al., 2019] and Bayesian Fusion (BF) [Dai et al., 2021]) have had impractical computational cost in a number of realistic settings, lacking robustness when considering: the number of sub-posteriors being unified; when unifying highly correlated sub-posteriors; the dimensionality of the sub-posteriors; and when considering conflicting sub-posteriors. In this thesis, we have substantially addressed the practical issues of existing Fusion approaches by means of a number of theoretical and methodological extensions of the original framework. In particular, we generalised the MCF and BF approaches in Section 6.1 and Section 7.1 respectively which incorporates available global information for each sub-posterior in order to construct more informative proposals, and through illustrative examples have shown the benefits of doing so (see for instance Sections 6.3.1, 7.4.1 and 7.4.2).

To address the problem of scalability with number of sub-posteriors, C, we embedded our Fusion approaches within a Divide-and-Conquer Sequential Monte Carlo (D&C-SMC) framework [Lindsten et al., 2017; Kuntz et al., 2021b]. We first introduced the *Divide-and-Conquer Generalised Monte Carlo Fusion (D&C-GMCF)* approach in Section 6.2, together with a number of *tree hierarchies*, which allow the sub-posteriors to be combined in stages to recover the fusion target density f. As demonstrated in Section 6.3.2, D&C-GMCF is a robust approach to unifying large numbers of sub-posteriors. Furthermore, as shown in Section 6.3.3, even in the setting of conflicting sub-posteriors, D&C-GMCF together with tempering and an appropriate hierarchy can result in a practical Fusion algorithm. In Section 6.4 we applied our D&C-GMCF approach to realistic datasets and compared its performance with competing approximate methodologies and in all of these

settings, our implementation of D&C-GMCF offered the best performance in terms of Integrated Absolute Distance (IAD) to an appropriate benchmark, at a modest computational cost.

Although our D&C-GMCF approach was able to build upon the existing MCF methodology and can be applied to a wider range of settings, we noted that there are settings where it was difficult to find an appropriate choice for T in Algorithm 6.2.1. The problem here is when we recursively call the Generalised Monte Carlo Fusion (GMCF) algorithm in Algorithm 6.2.1, it can be difficult to find a suitable time T which leads to a sufficiently large effective sample size after both importance sampling steps. This often occurs when combining sub-posteriors which have little overlapping support. In Section 6.3.3, we suggested a potential approach to combine conflicting sub-posteriors by employing a tempering idea but it is not entirely clear how to use this in practice. We subsequently considered embedding the more practical Generalised Bayesian Fusion (GBF) approach within a D&C-SMC algorithm to obtain the *Divide-and-Conquer Generalised Bayesian Fusion (D&C-GBF)* approach. By using the provided guidance for selecting the hyperparameters T and $\mathcal{P}$ required for the GBF approach in Section 7.3, we saw in Section 7.4.3 that our D&C-GBF approach was the most scalable Fusion approach to date with regards to dimension. Finally, in Section 7.5, we compared our D&C-GBF approach with a number of Fusion methodologies and found that it offered best performance in terms of IAD even at a similar computational cost. However, when compared to existing approximate methodologies, we still saw that our Fusion approaches typically came at an increased computational cost.

There are a number of interesting avenues for extending the work of this thesis. An immediate avenue of research is to pursue continued methodological advances of the Fusion methodology to further reduce the computational cost of the algorithms in order to make it more competitive to existing approximate approaches. Towards this goal, it would be interesting to adapt the Fusion approach to constraints in practical settings. In particular, one application instance is when considering a truly distributed *'big data'* setting where communication between different cores is expensive [Scott et al., 2016]. We discussed several practical implementation considerations in Section 7.3.3, but proper implementation of these techniques in larger data settings have yet to be explored. It would be particularly interesting to investigate embedding a sub-sampling approach within the Fusion algorithms (akin to the approaches of Pollock et al. [2020]; Bouchard-Côté et al. [2018]; Baker et al. [2019]; Bierkens et al. [2019]). We also note that there is a growing literature on implementing SMC approaches in parallel and distributed settings (see for instance Doucet and Lee [2018, Section 7.5.3]; Bolic et al. [2005]; Lee et al. [2010]; Murray et al. [2016]) which may also be interesting to integrate with our Fusion approaches. Furthermore, as discussed in the introduction, another particularly promising direction is considering (1.1) under *privacy constraints* of the individual sources [Yıldırım and Ermiş, 2019]. In this setting, we may have a number of parties that wish to combine their distributional analysis on a common parameter space and model but cannot reveal their distribution or data due to confidentiality. This application could also motivate variant tree hierarchies for D&C-GBF.

From a theoretical perspective, current Fusion methodologies only consider sub-posteriors on a common parameter space. One direction of interest is extending Fusion methodology to combine sub-posteriors with varying dimension. The *Markov Melding* framework of Goudie et al. [2019]; Manderson and Goudie [2022] where separate sub-models (potentially of differing dimension) are fitted to different data sources and then joined, is promising. In this setting, the tree hierarchies could be defined by the model itself. To mitigate computational robustness of Fusion with increasing dimension in this setting, it may be possible to further utilise the D&C-SMC methodology in Lindsten et al. [2017]. Furthermore, when considering the problem of performing Bayesian inference for large datasets, we have only considered the case where we assume the data is i.i.d., however there are many models where this is not the case (e.g. in hidden Markov models (HMMs) and with time series data). Ou et al. [2021] and Wang and Srivastava [2021] are two recent works which look at divide-and-conquer methods for Bayesian inference with non i.i.d. data. It would be interesting to explore an extension of the current Fusion methodology to non i.i.d. settings.

From a methodological perspective, it can be argued that the Fusion methodologies can difficult to implement since it utilises the path-space rejection sampling methodology which we discussed in Section 4.4. There is much work to do in this area to make Fusion methodologies more accessible by means of producing a software package which implements much of this machinery. Indeed, the availability of programs such as `BUGS` [Gilks et al., 1994], `JAGS` [Plummer et al., 2003], `Stan` [Carpenter et al., 2017] (which we have used a number of times in this thesis) have made standard Markov chain Monte Carlo samplers available to researchers and are used in many applications. There has also been some recent work by Corbella et al. [2022] which aims to make algorithms based on *Piecewise Deterministic Markov Processes (PDMPs)* more accessible by providing software implementations which only also only require the functional form of the target density of interest. The latter two approaches have an embedded *Automatic Differentiation (AD)* tool (see Baydin et al. [2018] for a review of such methods) which allows the user to only provide the functional form of the probability density function of interest. This is of particular importance to Fusion since the algorithms require the computation of the first and second order derivatives of the log-sub-posteriors, so being able to circumvent the need to analytically find the derivatives would be helpful for practitioners who are looking to implement the Fusion algorithms discussed within this thesis. Ideally, to make Fusion more easily accessible to a wider range of researchers or practitioners, one could envisage an *Automatic Fusion* (inspired by the *Automatic Zig-Zag* algorithm of Corbella et al. [2022]) approach whereby the user simply needs to provide the sub-posterior densities (potentially along with sub-posteriors samples) to combine the sub-posterior samples. Note that numerical optimisation methods to compute bounds of ϕ_c (6.9) may also be necessary for such an implementation too.

Part III

Appendices

Appendix A

Implementational details for examples

In this chapter, we provide details on where to find the code to implement the experiments in this thesis. All statistical computations found in this thesis were written in **R** [R Core Team, 2022], C++ and **Rcpp** [Eddelbuettel, 2013]. Code for the Fusion algorithms discussed in this thesis (along with documentation) can be found on GitHub at `https://github.com/rchan26/DCFusion`. In this package, we provide functions for the Fusion algorithms discussed in Chapters 5, 6 and 7 for the different examples we considered (see Appendix B for the calculations required for these examples). Furthermore, we have seen that the Fusion methodologies discussed in the thesis rely on algorithms for simulating sample paths of Brownian bridges, layered Brownian bridges, diffusions, and other related processes (i.e. methods discussed in Chapter 4). Code to implement algorithms found in Chapter 4 which are necessary for the implementation of Fusion can be found at `https://github.com/rchan26/layeredBB`.

To implement the approximate methodologies considered in this thesis (i.e. the methods discussed in Section 1.2.1), we used existing implementations for them. In particular, *Consensus Monte Carlo* [Scott et al., 2016] and the approach of Neiswanger et al. [2014] (which we termed *Kernel Density Estimate Monte Carlo (KDEMC)*) are implemented using implementations can be found the `parallelMCMCcombine` package in **R** available from [Miroshnikov and Conlon, 2014]. The *Weierstrass Sampler (WRS)* and is implemented using an existing **R** implementation (which can be found on GitHub at `https://github.com/wwrechard/weierstrass`).

To perform posterior (and sub-posterior) sampling for the examples in Sections 6.4 and 7.5, we used `Stan` [Carpenter et al., 2017]. Our `Stan` code for posterior sampling can be found at the following **R** packages:

- For the logistic regression example: `https://github.com/rchan26/HMCBLR`.
- For the robust regression example: `https://github.com/rchan26/HMCBRR`.

- For the negative binomial regression example: `https://github.com/rchan26/HMCGLMR`.

All experiments were ran on a 2021 MacBook Pro (M1 Pro, 14-inch, 16GB RAM) besides the examples provided in Sections 7.5.3.2 and 7.5.3.3 where these experiments were ran on Microsoft Azure computing platform using a 64 core Data Science Virtual Machine with 128GB RAM in order to deal with the larger datasets.

For the remainder of the chapter, we will provide details where the datasets that were used in this thesis can be found.

A.1 Credit card data example

In Section 6.4.2, we fitted a logistic regression model to the *'Default of credit card clients'* dataset available from the *UCI Machine Learning Repository* [Yeh and Lien, 2009]. This can be found at `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`.

A.2 Power plant data example

In Section 7.5.1, we considered a robust regression example with the *'Combined Cycle Power Plant'* dataset available from the *UCI Machine Learning Repository* [Kaya et al., 2012; Tüfekci, 2014] at `https://archive.ics.uci.edu/ml/datasets/combined+cycle+power+plant`.

A.3 Bike sharing data example

In Section 7.5.2, we considered a negative binomial regression example with the *'Bike Sharing'* dataset available from the *UCI Machine Learning Repository* [Fanaee-T and Gama, 2014]. This dataset can be found at `https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset` and in particular, we used the hourly (`hour.csv`) recorded dataset from this link.

A.4 Smart grid stability data example

In Section 7.5.3.2, we looked the *'Smart Grid Stability'* dataset available on the *Kaggle* [Schäfer et al., 2016] at `https://www.kaggle.com/datasets/pcbreviglieri/smart-grid-stability`, and considered applying a logistic regression model to the data.

A.5 NYC flights data example

In Section 7.5.3.3, we applied a logistic regression model to the `nycflights13` dataset. This can be accessed by installing the `nycflights13` [Wickham, 2021] package in **R**. In this example, we used

the dataset to create $p = 20$ predictors (for predicting the binary outcome of the flight arriving late or not). In particular, we created the following variables:

- `dep_delayed`: binary variable with 1 if flight departed at least 1 minute late, or 0 if not,
- `weekday`: binary variable with 1 if flight was on a weekday (Monday-Friday), or 0 if not,
- `night`: binary variable with 1 if flight was at night (8PM-5AM), or 0 if not,
- `carrier`: categorical variable with 16 categories providing the carrier of the flight (resulting in 15 binary variables),
- `origin`: categorical variable with 3 categories providing the origin airport of the flight (resulting in 2 binary variables).

Appendix B

Calculations for examples

In this chapter, we provide the calculations necessary to implement the Fusion algorithms discussed in this thesis. In particular, to implement Monte Carlo Fusion [Dai et al., 2019] (see Section 5.1) or Bayesian Fusion [Dai et al., 2021] (see Section 5.2) we must be able to compute the function ϕ_c^{dl} (5.6). This requires the computation of the first and second order derivatives of the log-sub-posterior densities. Furthermore, we must be able to compute bounds of ϕ_c^{dl}. Similarly, in order to implement Generalised Monte Carlo Fusion (Section 6.1), Divide-and-Conquer Generalised Monte Carlo Fusion (Section 6.2), Generalised Bayesian Fusion (Section 7.1) and Divide-and-Conquer Generalised Bayesian Fusion (Section 7.2), we must be able to compute ϕ_c given in (6.9) and compute its bounds (as per Proposition 6.1.3 or otherwise). As such, we will provide the necessary calculations to implement the methodology in the various examples throughout this thesis.

B.1 Univariate distribution with light tails

In Section 5.1.3.1, we considered a univariate distribution where the sub-posterior densities were given by $f_c(x) = e^{-\frac{x^4}{2C}}$. The first and second derivatives of $\log f_c$ are given by

$$\frac{\mathrm{d}\log f_c(x)}{\mathrm{d}x} = -\frac{2x^3}{C},$$
$$\frac{\mathrm{d}^2\log f_c(x)}{\mathrm{d}x^2} = -\frac{6x^2}{C},$$

respectively. We applied Monte Carlo Fusion [Dai et al., 2019] (see Section 5.1) for this example, hence we must compute ϕ_c^{dl} given in (5.6). In one dimension, this is given by

$$\phi_c^{dl}(x) := \frac{1}{2}\left(\left(\frac{\mathrm{d}\log f_c(x)}{\mathrm{d}x}\right)^2 + \frac{\mathrm{d}^2\log f_c(x)}{\mathrm{d}x^2}\right). \tag{B.1}$$

Substituting in the first and second derivatives for this example, we arrive at (5.11). To compute lower and upper bounds, $L_X^{(c)}$ and $U_X^{(c)}$, of $\phi_c^{dl}(x)$ for $x \in R_c$, first note that in one dimension, R_c is simply an interval $[l, u]$. We can compute the derivative of $\phi_c^{dl}(x)$ to find that

$$\frac{\mathrm{d}\phi_c^{dl}(x)}{\mathrm{d}x} = \frac{12x^5}{C^2} - \frac{6x}{C}.$$

Setting this to 0, the turning points of ϕ_c^{dl} occur at $x = 0, -\left(\frac{C}{2}\right)^{1/4}, \left(\frac{C}{2}\right)^{1/4}$. To find focal bounds of ϕ_c^{dl}, we must evaluate ϕ_c^{dl} at $x = l$, $x = u$ and any of these turning points if they occur in the interval $[l, u]$. We obtain $L_X^{(c)}$ and $U_X^{(c)}$ by taking the minimum and maximum value of these points.

B.2 Univariate mixture Gaussian

In Section 5.1.3.2, we considered an example with *tempered* univariate mixture Gaussian sub-posterior distributions. In particular, we consider $f_c(x) \propto f(x)^\beta := \left(\sum_{k=1}^K w_k \cdot \mathcal{N}_1\left(x|\mu_k, \sigma_k^2\right)\right)^\beta$ where $\beta \in (0, 1)$, K denotes the number of components in the mixture, μ_k denotes the mean for component k, w_k denotes the weight for component k and $\mathcal{N}_1(x|\mu_k, \sigma_k^2)$ denotes the density of a univariate Normal distribution with mean μ_k and variance σ_k^2 for $k = 1, \ldots, K$. In our example in Section 5.1.3.2, we had $K = 3$, where the weights were $(0.35, 0.2, 0.45)$ with means $(-3, 2, 0.5)$ and variances $(1, 1.5^2, 0.5^2)$.

We applied Monte Carlo Fusion [Dai et al., 2019] (see Section 5.1) for this example, hence we must compute ϕ_c^{dl} given in (5.6). Here, since $f_c(x) \propto f(x)^\beta$, we have a one dimensional example with

$$\begin{aligned}\phi_c^{dl}(x) &:= \frac{1}{2}\left(\left(\frac{\mathrm{d}\log f_c(x)}{\mathrm{d}x}\right)^2 + \frac{\mathrm{d}^2 \log f_c(x)}{\mathrm{d}x^2}\right) \\ &= \frac{1}{2}\left(\left(\frac{f_c'(x)}{f_c(x)}\right)^2 + \frac{f_c(x) \cdot f_c''(x) - f_c'(x)^2}{f(x)^2}\right). \end{aligned} \tag{B.2}$$

We note that the derivatives of $f_c = f(x)^\beta$ where are given by

$$\begin{aligned} f_c'(x) &:= \frac{\mathrm{d}f(x)^\beta}{\mathrm{d}x} = \beta f'(x) \cdot f(x)^{\beta-1}, \\ f_c''(x) &:= \frac{\mathrm{d}^2 f(x)^\beta}{\mathrm{d}x^2} = \beta f''(x) \cdot f(x)^{\beta-1} + \beta(\beta-1) f'(x)^2 f(x)^{\beta-2}. \end{aligned}$$

We have

$$f(x) := \sum_{k=1}^K w_k \cdot \mathcal{N}_1\left(x|\mu_k, \sigma_k^2\right) = \sum_{k=1}^K w_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x-\mu_k)^2}{2\sigma_k^2}\right),$$

so the derivatives of $f(x)$ are simply the weighted sums of derivatives of Gaussian densities, in

particular:

$$f'(x) = \sum_{k=1}^{K} w_k \cdot \left(\frac{\mu_k - x}{\sigma_k^2} \right) \cdot \mathcal{N}_1 \left(x | \mu_k, \sigma_k^2 \right),$$
$$f''(x) = \sum_{k=1}^{K} w_k \cdot \left(\frac{(x-\mu_k)^2 - \sigma_k^2}{\sigma_k^4} \right) \cdot \mathcal{N}_1 \left(x | \mu_k, \sigma_k^2 \right).$$

We can use these results together to compute ϕ_c^{dl} in (B.2). To find upper and lower bounds of ϕ_c conditional on some simulated hypercube $R_c(x) := [l, u]$, we simply employ a numerical optimiser. In particular, we utilise the `optimise` function implemented in base in **R** (although note other programming languages will have similar numerical optimisation libraries available). This function searches the interval from l to u (which we input) for a minimum or maximum of the function and so returns $L_X^{(c)}$ and $U_X^{(c)}$ via Brent's method [Brent, 1973]. Since we are only optimising a function in one dimension, this is typically fast and accurate. We found this to be sufficient in our simulations and provided tight bounds for ϕ_c.

B.3 Univariate Gaussian

In Sections 6.3.2 and 6.3.3, we considered examples involving combining univariate Gaussian sub-posteriors, i.e. $f_c(x) \propto \exp\left(-\frac{(x-\mu_c)^2}{2C\sigma_c^2}\right)$. The first and second derivatives of $\log f_c$ are given by

$$\frac{\mathrm{d} \log f_c(x)}{\mathrm{d}x} = -\frac{(x-\mu_c)}{C\sigma_c^2},$$
$$\frac{\mathrm{d}^2 \log f_c(x)}{\mathrm{d}x^2} = -\frac{1}{C\sigma_c^2},$$

respectively. In these examples, we considered applying Generalised Monte Carlo Fusion (see Section 6.1) and hence we must compute ϕ_c given in (6.9). In one-dimension, this is simply given by

$$\phi_c(x) := \frac{\Lambda_c}{2} \left(\left(\frac{\mathrm{d} \log f_c(x)}{\mathrm{d}x} \right)^2 + \frac{\mathrm{d}^2 \log f_c(x)}{\mathrm{d}x^2} \right), \tag{B.3}$$

where Λ_c is a scalar. Therefore, the ϕ_c function for univariate Gaussian sub-posteriors is given by

$$\phi_c(x) = \frac{\Lambda_c}{2} \left(\frac{(x-\mu_c)^2}{C^2\sigma_c^4} - \frac{1}{C\sigma_c^2} \right). \tag{B.4}$$

Since ϕ_c (B.4) is simply a quadratic with global minimum occuring at $x = \mu_c$, then to find lower and upper bounds, $L_X^{(c)}$ and $U_X^{(c)}$, of ϕ_c for $x \in R_c := [l, u]$, then we can simply evaluate ϕ_c at the corners of our hypercube R_c (i.e. evaluate at $x = l$ and $x = u$), and if the mean $\mu_c \in [l, u]$, then we also must compute ϕ_c at the mean too (since we would simply take $L_X^{(c)} = \mu_c$ in this case, as this is

where the global minimum occurs). By taking the minimum and maximum values of these points, we can obtain $L_X^{(c)}$ and $U_X^{(c)}$.

B.4 Multivariate Gaussian

In Sections 6.3.1, 7.4.1, 7.4.2 and 7.4.3, we considered examples involving combining multivariate Gaussian sub-posteriors (noting that the first three of these examples, we considered bivariate Gaussian ($d = 2$) sub-posteriors). If $f_c \sim \mathcal{N}_d(\boldsymbol{\mu}_c, C\boldsymbol{\Sigma}_c)$, then

$$\log f_c(\boldsymbol{x}) = -\frac{1}{2C}(\boldsymbol{x} - \boldsymbol{\mu}_c)^\intercal \boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x} - \mu_c) + \text{constant},$$

and

$$\begin{aligned}\nabla \log f_c(\boldsymbol{x}) &= -\frac{1}{C}\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c),\\ \nabla^2 \log f_c(\boldsymbol{x}) &= -\frac{1}{C}\boldsymbol{\Sigma}_c^{-1}.\end{aligned}$$

Then we can use these directly to compute ϕ_c in (6.9). As with the univariate Gaussian example in Appendix B.3, we can follow a similar approach since we have a quadratic in $\boldsymbol{x}$. In particular, in each dimension $k = 1, \ldots, d$, given the bounds in Z-space $[l_k, u_k]$ say where $\boldsymbol{z}_{t,k} \in [l_k, u_k]$ for $t \in [0, T]$ (or $t \in [t_{j-1}, t_j]$ more generally in the BF setting), then the minimum of ϕ_c occurs at either l_k, u_k or $\boldsymbol{\mu}_{c,k}$ if $\boldsymbol{\mu}_{c,k} \in [l_k, u_k]$. Therefore, in each dimension, we can simply check if the value of the mean is included in the bound. We collect all permutations of the potential points where the minimum or maximum can occur in each dimension, evaluate at these points and use these to find $L_X^{(c)}$ and $U_X^{(c)}$.

B.5 Logistic Regression

In Sections 6.4.1, 6.4.2 and 7.5.3, we considered applying our Fusion methodologies to a logistic regression example with Gaussian prior distributions for the parameters. In particular, our sub-posterior densities were given by the posterior for Bayesian logistic regression with $\mathcal{N}_d(\mu_j, C\sigma_{\beta_j}^2)$ prior for β_j for $j = 0, \ldots, p$ is given by

$$f_c(\boldsymbol{\beta}) := \pi(\boldsymbol{\beta}|\boldsymbol{y}) = \left[\prod_{i=1}^{n} \frac{e^{X_i\boldsymbol{\beta}\cdot y_i}}{1 + e^{X_i\boldsymbol{\beta}}}\right] \cdot \left[\prod_{j=0}^{p} \frac{1}{\sqrt{2\pi C\sigma_{\beta_j}^2}} \exp\left(-\frac{(\beta_j - \mu_j)^2}{2C\sigma_{\beta_j}^2}\right)\right] \tag{B.5}$$

where $X \in \mathbb{R}^{n\times(p+1)}$ is the design matrix so $X_i\boldsymbol{\beta} = \beta_0 + \beta_1 X_{i1} + \cdots \beta_p X_{ip}$. The log-posterior is given by

$$\log f_c(\boldsymbol{\beta}) = \sum_{i=1}^{n}\left[X_i\boldsymbol{\beta}\cdot y_i - \log\left(1 + e^{\boldsymbol{\beta}X_i}\right)\right] - \sum_{j=0}^{p}\frac{(\beta_j - \mu_j)^2}{2C\sigma_{\beta_j}^2} + \text{constant}. \tag{B.6}$$

The first derivative of the log-posterior with respect to β_k for $k = 0, \ldots, p$, is given by

$$\begin{aligned}\frac{\partial \log f_c(\boldsymbol{\beta})}{\partial \beta_k} &= \sum_{i=1}^{n} \left[X_{ik} \cdot y_i - \frac{X_{ik} e^{X_i \boldsymbol{\beta}}}{1 + e^{X_i \boldsymbol{\beta}}} \right] - \frac{(\beta_k - \mu_k)}{C \sigma^2_{\beta_k}} \\ &= \sum_{i=1}^{n} \left[X_{ik} \cdot \left(y_i - \frac{1}{1 + e^{-X_i \boldsymbol{\beta}}} \right) \right] - \frac{(\beta_k - \mu_k)}{C \sigma^2_{\beta_k}} \end{aligned} \tag{B.7}$$

and the second order derivatives of the log-posterior are given by

$$\frac{\partial^2 \log f_c(\boldsymbol{\beta})}{\partial \beta_k^2} = -\sum_{i=1}^{n} \frac{X_{ik}^2 e^{X_i \boldsymbol{\beta}}}{(1 + e^{X_i \boldsymbol{\beta}})^2} - \frac{1}{C \sigma^2_{\beta_k}}, \tag{B.8}$$

$$\frac{\partial^2 \log f_c(\boldsymbol{\beta})}{\partial \beta_k \partial \beta_l} = -\sum_{i=1}^{n} \frac{X_{ik} X_{il} e^{X_i \boldsymbol{\beta}}}{(1 + e^{X_i \boldsymbol{\beta}})^2} \text{ for } k \neq l, \tag{B.9}$$

for $k, l = 0, \ldots, p$. We can use these directly to compute ϕ_c given in (6.9).

B.5.1 Computing the bounds of ϕ_c

To compute the bounds of ϕ_c, we can utilise the bounds provided in Proposition 6.1.3 (or in (6.11) and (6.12)). To do so, we must be able to compute an upper bound of the matrix norm $\boldsymbol{\Lambda}_c \nabla^2 \log f_c(\boldsymbol{x})$ for $\boldsymbol{x} \in R_c$ where R_c denotes the simulated layer information, i.e. to compute (6.13) which occurs in the bounds. While this can be done by computing the matrix norm of the matrix which bounds the matrix $\boldsymbol{\Lambda}_c \nabla^2 \log f_c(\boldsymbol{x})$ element-wise, we noted in Section 6.1.2 that it is typically easier to find bounds on the matrix norm of $\nabla^2 \log f_c^{(z)}(\boldsymbol{z})$ where $\boldsymbol{z} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{\beta}$, and instead we can focus on finding a bound in the transformed space, i.e. compute (6.23).

In this logistic regression setting, let $\boldsymbol{z} = \boldsymbol{\Lambda}_c^{-\frac{1}{2}} \boldsymbol{\beta}$ then the transformed posterior density is given by

$$f_c^{(z)}(\boldsymbol{z}) = \pi(\boldsymbol{\beta} | X, \boldsymbol{y}) |J|, \tag{B.10}$$

where $J = \boldsymbol{\Lambda}_c^{-\frac{1}{2}}$ is the Jacobian matrix with elements $J_{ij} = \frac{\partial z_i}{\partial \beta_j} = \boldsymbol{\Lambda}_{c,ij}^{-\frac{1}{2}}$. We have

$$\log f_c^{(z)}(\boldsymbol{z}) = \log \pi(\boldsymbol{\beta} | X, \boldsymbol{y}) + \log |J|. \tag{B.11}$$

Since $\boldsymbol{\beta} = \boldsymbol{\Lambda}_c^{\frac{1}{2}} \boldsymbol{z}$, we have

$$\begin{aligned} f_c^{(z)}(\boldsymbol{z}) &:= \pi(\boldsymbol{\beta} | X, \boldsymbol{y}) \cdot |\boldsymbol{\Lambda}_c^{-\frac{1}{2}}| \\ &= \left[\prod_{i=1}^{n} \frac{e^{X_i \boldsymbol{\beta} \cdot y_i}}{1 + e^{X_i \boldsymbol{\beta}}} \right] \cdot \left[\prod_{j=0}^{p} \frac{1}{\sqrt{2\pi C \sigma^2_{\beta_j}}} \exp\left(-\frac{(\beta_j - \mu_j)^2}{2C \sigma^2_{\beta_j}} \right) \right] \cdot |\boldsymbol{\Lambda}_c^{-\frac{1}{2}}| \end{aligned}$$

$$= \left[\prod_{i=1}^{n} \frac{e^{X_i(\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})\cdot y_i}}{1+e^{X_i(\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})}}\right] \cdot \left[\prod_{j=0}^{p} \frac{1}{\sqrt{2\pi C\sigma^2_{\beta_j}}} \exp\left(-\frac{\left((\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_j - \mu_j\right)^2}{2C\sigma^2_{\beta_j}}\right)\right] \cdot |\boldsymbol{\Lambda}_c^{-\frac{1}{2}}|, \tag{B.12}$$

so

$$\log f_c^{(z)}(\boldsymbol{z}) = \sum_{i=1}^{n}\left[X_i(\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})\cdot y_i - \log\left(1+e^{X_i(\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})}\right)\right] - \sum_{j=0}^{p}\frac{\left((\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_j - \mu_j\right)^2}{2C\sigma^2_{\beta_j}} + \text{constant}. \tag{B.13}$$

We first note that since $\boldsymbol{\beta} = \boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z}$, then $\beta_i = (\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_i = \sum_k \Lambda_{ik}^{\frac{1}{2}} z_k$. So we have

$$\begin{aligned}
\frac{\partial(X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}}{\partial z_k} &= \frac{\partial}{\partial z_k}\sum_j X_{ij}\beta_j \\
&= \frac{\partial}{\partial z_k}\sum_j X_{ij}\left(\sum_k \Lambda_{jk}^{\frac{1}{2}} z_k\right) \\
&= \sum_j X_{ij}\Lambda_{jk}^{\frac{1}{2}} \\
&= (X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{ik}
\end{aligned} \tag{B.14}$$

and also we have

$$\frac{\partial(\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_i}{\partial z_k} = \frac{\partial}{\partial z_k}\sum_j \Lambda_{ij}^{\frac{1}{2}} z_j = \Lambda_{ik}^{\frac{1}{2}} \tag{B.15}$$

Using (B.14) and (B.15), then the first derivative of the log-transformed posterior with respect to β_k for $k = 0, \ldots, p$, is given by

$$\frac{\partial \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k} = \sum_{i=1}^{n}\left[(X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{ik}\cdot\left(y_i - \frac{1}{1+e^{-(X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}}}\right)\right] - \sum_{j=0}^{p}\frac{\Lambda_{jk}^{\frac{1}{2}}\left((\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_j - \mu_j\right)}{C\sigma^2_{\beta_j}}. \tag{B.16}$$

Then the second order derivatives are given by

$$\frac{\partial^2 \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k \partial z_l} = -\sum_{i=1}^{n}\frac{(X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{ik}(X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{il}e^{(X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}}}{\left(1+e^{(X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}}\right)^2} - \sum_{j=0}^{p}\frac{\Lambda_{jk}^{\frac{1}{2}}\Lambda_{jl}^{\frac{1}{2}}}{C\sigma^2_{\beta_j}}, \tag{B.17}$$

for $k, l = 0, \ldots, p$.

To find bounds for ϕ_c, we must now try to find bounds on the second derivatives given above and compute the matrix norm of the matrix made up of these bounds (which ultimately bounds $\nabla^2 \log f_c^{(z)}(\boldsymbol{z})$ element-wise). For this example, we can find global and lower bounds of the second derivatives. Note however, we typically will expect better performance with the local bounds on $P^{\boldsymbol{\Lambda}_c}$ (6.23) (as this will typically lead to the expected number of points we need to evaluate while performing Poisson thinning, κ_c, to be lower) despite these bounds being slightly more expensive to compute in practice.

B.5.1.1 Global bounds of $P^{\boldsymbol{\Lambda}_c}$

We first note that $\frac{e^x}{(1+e^x)^2} \leq \frac{1}{4}$ for all x (and this maximum occurs at $x = 0$). We can utilise this to obtain a global bound:

$$\sup \left[\left| \frac{\partial^2 \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k \partial z_l} \right| \right] = \sum_{i=1}^{n} \frac{|X\boldsymbol{\Lambda}_c^{\frac{1}{2}}|_{ik} \cdot |X\boldsymbol{\Lambda}_c^{\frac{1}{2}}|_{il}}{4} + \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}} \Lambda_{jl}^{\frac{1}{2}}}{C\sigma_{\beta_j}^2}. \tag{B.18}$$

B.5.1.2 Local bounds of $P^{\boldsymbol{\Lambda}_c}$

Local bounds can be obtained if we can find local bounds for

$$G_1(\boldsymbol{z}) := \frac{e^{(X_i \boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}}}{\left(1 + e^{(X_i \boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}}\right)^2}, \tag{B.19}$$

for $i = 1, \ldots, n$. In that case, we have

$$\sup_{\boldsymbol{z} \in R^{(z)}} \left[\left| \frac{\partial^2 \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k \partial z_l} \right| \right] = \sum_{i=1}^{n} \left[|X\boldsymbol{\Lambda}_c^{\frac{1}{2}}|_{ik} \cdot |X\boldsymbol{\Lambda}_c^{\frac{1}{2}}|_{il} \cdot \max_{\boldsymbol{z} \in R^{(z)}} \{G_1(\boldsymbol{z})\} \right] + \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}} \Lambda_{jl}^{\frac{1}{2}}}{C\sigma_{\beta_j}^2}. \tag{B.20}$$

To compute $\max_{\boldsymbol{z} \in R^{(z)}} \{G_1(\boldsymbol{z})\}$, see Section B.7.1.2 and Algorithm B.7.1 and set $r = 1$.

B.6 Robust Regression

In Section 7.5.1, we considered a robust regression example (using a student-t distribution) with Gaussian prior distributions for the parameters. In particular, our sub-posterior densities were given by the posterior for Bayesian robust regression with $\mathcal{N}_d(\mu_j, C\sigma_{\beta_j}^2)$ prior for β_j for $j = 0, \ldots, p$ is given by

$$f_c(\boldsymbol{\beta}) = \pi(\boldsymbol{\beta}|X, \boldsymbol{y}) := \left[\prod_{i=1}^{n} \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})\sqrt{\pi\nu}\sigma} \left(1 + \frac{1}{\nu} \left(\frac{y_i - X_i\boldsymbol{\beta}}{\sigma} \right)^2 \right)^{-\left(\frac{\nu+1}{2}\right)} \right]$$

$$\cdot \left[\prod_{j=0}^{p} \frac{1}{\sqrt{2\pi C\sigma_{\beta_j}^2}} \exp\left(-\frac{(\beta_j - \mu_j)^2}{2C\sigma_{\beta_j}^2} \right) \right]. \tag{B.21}$$

The log-posterior is given by

$$\log f_c(\boldsymbol{\beta}) = -\left(\frac{\nu+1}{2}\right) \sum_{i=1}^{n} \log\left(1 + \frac{1}{\nu\sigma^2}\left(y_i - X_i\boldsymbol{\beta}\right)^2\right) - \sum_{j=0}^{p} \frac{(\beta_j - \mu_j)^2}{2C\sigma_{\beta_j}^2} + \text{constant}. \tag{B.22}$$

The first derivative of the log-posterior with respect to β_k for $k = 0, \ldots, p$ is given by

$$\begin{aligned} \frac{\partial \log \pi(\boldsymbol{\beta}|X, \boldsymbol{y})}{\partial \beta_k} &= -\left(\frac{\nu+1}{2}\right) \sum_{i=1}^{n} \frac{-\frac{2X_{ik}}{\nu\sigma^2}(y_i - X_i\boldsymbol{\beta})}{1 + \frac{1}{\nu\sigma^2}(y_i - X_i\boldsymbol{\beta})^2} - \frac{(\beta_k - \mu_k)}{C\sigma_{\beta_k}^2} \\ &= (\nu+1) \sum_{i=1}^{n} \frac{X_{ik}(y_i - X_i\boldsymbol{\beta})}{\nu\sigma^2 + (y_i - X_i\boldsymbol{\beta})^2} - \frac{(\beta_k - \mu_k)}{C\sigma_{\beta_k}^2}, \end{aligned} \tag{B.23}$$

and the second order derivatives of the log-posterior are given by

$$\frac{\partial^2 \log \pi(\boldsymbol{\beta}|X, \boldsymbol{y})}{\partial \beta_k^2} = (\nu+1) \sum_{i=1}^{n} \frac{X_{ik}^2 \left((y_i - X_i\boldsymbol{\beta})^2 - \nu\sigma^2\right)}{\left(\nu\sigma^2 + (y_i - X_i\boldsymbol{\beta})^2\right)^2} - \frac{1}{C\sigma_{\beta_k}^2}, \tag{B.24}$$

$$\frac{\partial^2 \log \pi(\boldsymbol{\beta}|X, \boldsymbol{y})}{\partial \beta_k \partial \beta_l} = (\nu+1) \sum_{i=1}^{n} \frac{X_{ik}X_{il} \left((y_i - X_i\boldsymbol{\beta})^2 - \nu\sigma^2\right)}{\left(\nu\sigma^2 + (y_i - X_i\boldsymbol{\beta})^2\right)^2} \text{ for } k \neq l, \tag{B.25}$$

for $k, l = 0, \ldots, p$. We can use these derivatives directly to compute ϕ_c given in (6.9).

B.6.1 Computing the bounds of ϕ_c

Following in the same approach as Section B.5.1, we can compute the bounds of ϕ_c (in (6.9)) by utilising the bounds provided in (6.11) and (6.12). As noted in Section B.5.1, we must be able to find an upper bound on the matrix norm of $\nabla^2 \log f_c^{(z)}(\boldsymbol{z})$ where $\boldsymbol{z} := \boldsymbol{\Lambda}_c^{-\frac{1}{2}}\boldsymbol{\beta}$, i.e. compute (6.23). To do so, we can compute the matrix norm of the matrix which bounds $\nabla^2 \log f_c^{(z)}(\boldsymbol{z})$ element-wise. Now, let $\boldsymbol{z} = \boldsymbol{\Lambda}_c^{-\frac{1}{2}}\boldsymbol{\beta}$ then $f_c^{(z)}(\boldsymbol{z}) = \pi(\boldsymbol{\beta}|X, \boldsymbol{y})|J|$, where $J = \boldsymbol{\Lambda}_c^{-\frac{1}{2}}$ is the Jacobian matrix, so we have

$$\begin{aligned} f_c^{(z)}(\boldsymbol{z}) &:= \pi(\boldsymbol{\beta}|X, \boldsymbol{y}) \cdot |\boldsymbol{\Lambda}_c^{-\frac{1}{2}}| \\ &= \left[\prod_{i=1}^{n} \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})\sqrt{\pi\nu}\sigma} \left(1 + \frac{1}{\nu}\left(\frac{y_i - X_i(\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})}{\sigma}\right)^2\right)^{-\left(\frac{\nu+1}{2}\right)} \right] \end{aligned}$$

$$\cdot \left[\prod_{j=0}^{p} \frac{1}{\sqrt{2\pi C\sigma_{\beta_j}^2}} \exp\left(-\frac{\left((\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_j - \mu_j \right)^2}{2C\sigma_{\beta_j}^2} \right) \right] \cdot |\boldsymbol{\Lambda}_c^{-\frac{1}{2}}|. \tag{B.26}$$

so

$$\begin{aligned} \log f_c^{(z)}(\boldsymbol{z}) = &-\left(\frac{\nu+1}{2}\right) \sum_{i=1}^{n} \log\left(1 + \frac{1}{\nu\sigma^2}\left(y_i - X_i(\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})\right)^2\right) \\ &- \sum_{j=0}^{p} \frac{\left((\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_j - \mu_j \right)^2}{2C\sigma_{\beta_j}^2} + \text{constant}. \end{aligned} \tag{B.27}$$

Recall from (B.14) and (B.15), then The first derivative of the log-transformed posterior with respect to β_k for $k = 0, \ldots, p$ is given by

$$\begin{aligned} \frac{\partial \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k} &= \left(\frac{\nu+1}{2}\right) \sum_{i=1}^{n} \frac{-\frac{2(X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{ik}}{\nu\sigma^2}\left(y_i - (X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)}{1 + \frac{1}{\nu\sigma^2}\left(y_i - (X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)^2} - \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}}\left((\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_j - \mu_j \right)}{C\sigma_{\beta_j}^2} \\ &= (\nu+1) \sum_{i=1}^{n} \frac{(X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{ik}\left(y_i - (X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)}{\nu\sigma^2 + \left(y_i - (X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)^2} - \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}}\left((\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_j - \mu_j \right)}{C\sigma_{\beta_j}^2} \end{aligned} \tag{B.28}$$

Then the second order derivatives are given by

$$\frac{\partial^2 \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k \partial z_l} = (\nu+1) \sum_{i=1}^{n} \frac{(X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{ik}(X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{il}\left(\left(y_i - (X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)^2 - \nu\sigma^2\right)}{\left(\left(y_i - (X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)^2 + \nu\sigma^2\right)^2} - \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}}\Lambda_{jl}^{\frac{1}{2}}}{C\sigma_{\beta_j}^2} \tag{B.29}$$

for $k, l = 0, \ldots, p$.

B.6.1.1 Global bounds of $P^{\boldsymbol{\Lambda}_c}$

To compute $P^{\boldsymbol{\Lambda}_c}$ for this example, first note that we can write

$$\frac{\partial^2 \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k \partial z_l} = (\nu+1) \sum_{i=1}^{n} (X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{ik}(X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{il}\left[\frac{1}{E_i + b} - \frac{2b}{(E_i+b)^2}\right] - \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}}\Lambda_{jl}^{\frac{1}{2}}}{C\sigma_{\beta_j}^2}, \tag{B.30}$$

for $k, l = 0, \ldots, p$, where $b = \nu\sigma^2$ and $E_i = \left(y_i - (X_i \boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)^2$. Now let

$$K(E_i) = \frac{1}{E_i + b} - \frac{2b}{(E_i + b)^2},$$

then the derivative is given by

$$K'(E_i) = -\frac{1}{(E_i + b)^2} + \frac{4b}{(E_i + b)^3}.$$

Setting $K'(E_i) = 0$ gives $E_i = 3b$, and we have $K(E_i = 3b) = \frac{1}{8b}$. So the supremum of the second derivative is given by

$$\sup\left[\left|\frac{\partial^2 \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k \partial z_l}\right|\right] = \frac{(\nu+1)}{8\nu\sigma^2}\sum_{i=1}^{n} |X\boldsymbol{\Lambda}_c^{\frac{1}{2}}|_{ik} \cdot |X\boldsymbol{\Lambda}_c^{\frac{1}{2}}|_{il} - \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}}\Lambda_{jl}^{\frac{1}{2}}}{C\sigma_{\beta_j}^2}. \tag{B.31}$$

We can therefore use this to compute $P^{\boldsymbol{\Lambda}_c}$ to compute bounds for ϕ_c as per (6.11) and (6.12).

B.7 Negative Binomial Regression

In Section 7.5.2, we considered a negative Binomial regression example with Gaussian prior distributions for the parameters. In particular, our sub-posterior densities were given by the posterior density with $\mathcal{N}_d(\mu_j, C\sigma_{\beta_j}^2)$ priors for β_j for $j = 0, \ldots, p$, is given by

$$\begin{aligned} f_c(\boldsymbol{\beta}) &:= \pi(\boldsymbol{\beta}|X, \boldsymbol{y}) \\ &= \left[\prod_{i=1}^{n} \frac{\Gamma(y_i + r)}{y_i!\Gamma(r)} \left(\frac{\mu_i}{\mu_i + r}\right)^{y_i} \left(\frac{r}{\mu_i + r}\right)^{r}\right] \cdot \left[\prod_{j=0}^{p} \frac{1}{\sqrt{2\pi C\sigma_{\beta_j}^2}} \exp\left(-\frac{(\beta_j - \mu_j)^2}{2C\sigma_{\beta_j}^2}\right)\right] \\ &= \left[\prod_{i=1}^{n} \frac{\Gamma(y_i + r)}{y_i!\Gamma(r)} \frac{\exp(X_i\boldsymbol{\beta} \cdot y_i) \cdot r^r}{(\exp(X_i\boldsymbol{\beta}) + r)^{y_i + r}}\right] \cdot \left[\prod_{j=0}^{p} \frac{1}{\sqrt{2\pi C\sigma_{\beta_j}^2}} \exp\left(-\frac{(\beta_j - \mu_j)^2}{2C\sigma_{\beta_j}^2}\right)\right] \end{aligned} \tag{B.32}$$

The log-posterior is given by

$$\log f_c(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[X_i\boldsymbol{\beta} \cdot y_i - (y_i + r)\log\left(\exp(X_i\boldsymbol{\beta}) + r\right)\right] - \sum_{j=0}^{p} \frac{(\beta_j - \mu_j)^2}{2C\sigma_{\beta_j}} + \text{constant}. \tag{B.33}$$

The first order derivative of the log-posterior with respect to β_k for $k = 0, \ldots, p$, is given by

$$\frac{\partial \log f_c(\boldsymbol{\beta})}{\partial \beta_k} = \sum_{i=1}^{n}\left[X_{ik}y_i - \frac{(y_i + r)X_{ik}\exp(X_i\boldsymbol{\beta})}{\exp(X_i\boldsymbol{\beta}) + r}\right] - \frac{(\beta_k - \mu_k)}{C\sigma_{\beta_k}^2},$$

$$= \sum_{i=1}^{n} \left[X_{ik} \cdot \left(y_i - \frac{(y_i + r)\exp(X_i\boldsymbol{\beta})}{\exp(X_i\boldsymbol{\beta}) + r} \right) \right] - \frac{(\beta_k - \mu_k)}{C\sigma_{\beta_k}^2}, \tag{B.34}$$

and the second order derivatives of the log-posterior are given by

$$\frac{\partial^2 \log f_c(\boldsymbol{\beta})}{\partial \beta_k^2} = -\sum_{i=1}^{n} \frac{(y_i + r) r X_{ik}^2 \exp(X_i\boldsymbol{\beta})}{\left(\exp(X_i\boldsymbol{\beta}) + r\right)^2} - \frac{1}{C\sigma_{\beta_k}^2}, \tag{B.35}$$

$$\frac{\partial^2 \log f_c(\boldsymbol{\beta})}{\partial \beta_k \partial \beta_l} = -\sum_{i=1}^{n} \frac{(y_i + r) r X_{ik} X_{il} \exp(X_i\boldsymbol{\beta})}{\left(\exp(X_i\boldsymbol{\beta}) + r\right)^2} \quad \text{for } k \neq l, \tag{B.36}$$

for $k, l = 0, \ldots, p$. We can use these directly to compute ϕ_c given in (6.9).

B.7.1 Computing the bounds of ϕ_c

Following in the same approach as Section B.5.1, we can compute the bounds of ϕ_c (in (6.9)) by utilising the bounds provided in (6.11) and (6.12). As noted in Section B.5.1, we must compute (6.23). To do so, we can compute the matrix norm of the matrix which bounds $\nabla^2 \log f_c^{(z)}(\boldsymbol{z})$ element-wise. We have

$$\begin{aligned} f_c^{(z)}(\boldsymbol{z}) &:= \pi(\boldsymbol{\beta}|X, \boldsymbol{y}) \cdot |\boldsymbol{\Lambda}_c^{-\frac{1}{2}}| \\ &= \left[\prod_{i=1}^{n} \frac{\Gamma(y_i + r)}{y_i! \Gamma(r)} \frac{\exp\left(X_i(\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z}) \cdot y_i \right) \cdot r^r}{\left(\exp\left(X_i(\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z}) \right) + r \right)^{y_i + r}} \right] \\ &\quad \cdot \left[\prod_{j=0}^{p} \frac{1}{\sqrt{2\pi C\sigma_{\beta_j}^2}} \exp\left(- \frac{\left((\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_j - \mu_j \right)^2}{2C\sigma_{\beta_j}^2} \right) \right] \cdot |\boldsymbol{\Lambda}_c^{-\frac{1}{2}}|, \end{aligned} \tag{B.37}$$

and

$$\begin{aligned} \log f_c^{(z)}(\boldsymbol{z}) = \sum_{i=1}^{n} &\left[(X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z} \cdot y_i - (y_i + r) \log\left(\exp\left((X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z} \right) + r \right) \right] \\ &- \sum_{j=0}^{p} \frac{\left((\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_j - \mu_j \right)^2}{2C\sigma_{\beta_j}} + \text{constant}. \end{aligned} \tag{B.38}$$

Recall from (B.14) and (B.15) that we have $\frac{\partial (X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}}{\partial z_k} = (X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{ik}$ and $\frac{\partial (\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_i}{\partial z_k} = \Lambda_{ik}^{\frac{1}{2}}$. Then first

derivative of the log-transformed posterior with respect to β_k is given by

$$\frac{\partial \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k} = \sum_{i=1}^{n} \left[(X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{ik} \cdot \left(y_i - \frac{(y_i + r)\exp\left((X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)}{\exp\left((X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right) + r} \right) \right] - \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}} \left((\boldsymbol{\Lambda}_c^{\frac{1}{2}}\boldsymbol{z})_j - \mu_j \right)}{C\sigma_{\beta_j}^2} \tag{B.39}$$

and the second order derivatives are given by

$$\frac{\partial^2 \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k \partial z_l} = -\sum_{i=1}^{n} \frac{(y_i + r) r (X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{ik} (X\boldsymbol{\Lambda}_c^{\frac{1}{2}})_{il} \exp\left((X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)}{\left(\exp\left((X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right) + r\right)^2} - \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}} \Lambda_{jl}^{\frac{1}{2}}}{C\sigma_{\beta_j}^2}. \tag{B.40}$$

To find bounds for r_c, we must now try to find bounds on the second derivatives given above and compute the matrix norm of the matrix made up of these bounds (which ultimately bounds $\nabla^2 \log f_c^{(z)}(\boldsymbol{z})$ element-wise). For this example, we can find global and lower bounds of the second derivatives. Note however, we typically will expect better performance with the local bounds on $P^{\boldsymbol{\Lambda}_c}$ (as this will typically lead to the expected number of points we need to evaluate while performing Poisson thinning, κ_c, to be lower) despite these bounds being slightly more expensive to compute in practice.

B.7.1.1 Global bounds of $P^{\boldsymbol{\Lambda}_c}$

Note that $\frac{e^{ax}}{(e^{ax}+r)^2} \leq \frac{1}{4r}$ for all x (where a is some constant), so we can use this to obtain global bounds on the matrix norm in the transformed space. Note that this maximum occurs at $x = \frac{1}{a}\log(r)$. To find global bounds, we can use

$$\sup\left[\left|\frac{\partial^2 \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k \partial z_l}\right|\right] = \sum_{i=1}^{n} \frac{(y_i + r) r |X\boldsymbol{\Lambda}_c^{\frac{1}{2}}|_{ik} \cdot |X\boldsymbol{\Lambda}_c^{\frac{1}{2}}|_{il}}{4r} + \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}} \Lambda_{jl}^{\frac{1}{2}}}{C\sigma_{\beta_j}^2}. \tag{B.41}$$

B.7.1.2 Local bounds of $P^{\boldsymbol{\Lambda}_c}$

Local bounds can be obtained if we can find local bounds for

$$G_r(\boldsymbol{z}) := \frac{\exp\left((X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)}{\left(\exp\left((X_i\boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right) + r\right)^2} \tag{B.42}$$

for $\boldsymbol{z} \in R^{(z)}$ and $i = 1, \ldots, n$. In that case, we have

$$\sup_{\boldsymbol{z} \in R^{(z)}} \left[\left| \frac{\partial^2 \log f_c^{(z)}(\boldsymbol{z})}{\partial z_k \partial z_l} \right| \right] = \sum_{i=1}^{n} \left[(y_i + r) r |X \boldsymbol{\Lambda}_c^{\frac{1}{2}}|_{ik} \cdot |X \boldsymbol{\Lambda}_c^{\frac{1}{2}}|_{il} \cdot \max_{\boldsymbol{z} \in R^{(z)}} \{G_r(\boldsymbol{z})\} \right] + \sum_{j=0}^{p} \frac{\Lambda_{jk}^{\frac{1}{2}} \Lambda_{jl}^{\frac{1}{2}}}{C \sigma_{\beta_j}^2}. \quad \text{(B.43)}$$

We can obtain bounds for $G_r(\boldsymbol{z})$ by noting that $\frac{\exp(x)}{(r+\exp(x))^2} \leq \frac{1}{4r}$ for all x and this maximum is attained at $x = \log(r)$. Further note that $\frac{\exp(x)}{(r+\exp(x))^2} \leq \frac{1}{4r}$ is a uni-modal function (with mode at $x = \log(r)$ as noted). Now let

$$F_i(\boldsymbol{z}) := (X_i \boldsymbol{\Lambda}_c^{\frac{1}{2}}) \boldsymbol{z} = \sum_{j=1}^{d} (X_i \boldsymbol{\Lambda}_c^{\frac{1}{2}})_j \boldsymbol{z}_j, \quad \text{(B.44)}$$

then let $F_i^{\downarrow} := \min_{\boldsymbol{z} \in R^{(z)}} F_i(\boldsymbol{z})$ and $F_i^{\uparrow} := \max_{\boldsymbol{z} \in R^{(z)}} F_i(\boldsymbol{z})$ denote the minimum and maximum of $F_i(\boldsymbol{z})$ for $\boldsymbol{z} \in R^{(z)}$ respectively. Then we note that this can simply be computed in with a linear cost with d. Now, noting that $F_i(\boldsymbol{z})$ is linear and $\frac{\exp(x)}{(r+\exp(x))^2} \leq \frac{1}{4r}$ is uni-modal, after computing $F_i^{\downarrow}$ and $F_i^{\uparrow}$, there are two cases:

1. If we have $\log(r) \in [F_i^{\downarrow}, F_i^{\uparrow}]$, then we know that for this hypercube $R^{(z)}$, we will attain the maximum $\frac{1}{4r}$.

2. If $\log(r) \notin [F_i^{\downarrow}, F_i^{\uparrow}]$, then the maximum of $G_r(x)$ occurs at which ever point is the closest to $\log(r)$.

Therefore local bounds can be obtained by minimising and maximising $F_i(\boldsymbol{z})$ for $\boldsymbol{z} \in R^{(z)}$. If this interval includes $\log(r)$, then the local maximum attains the global maximum, otherwise, the local maximum occurs at either of these intervals (which ever is closer to $\log(r)$).

This method for finding local bounds requires two optimisations of $F_i(\boldsymbol{z})$, but we note that we can actually obtain the bounds by only performing one optimisation. In particular, we can evaluate $F_i(\boldsymbol{z})$ at any arbitrary value $\hat{\boldsymbol{z}} \in R^{(z)}$ (we can simply take this to be the centre of the hypercube). If we have $F_i(\hat{\boldsymbol{z}}) > \log(r)$, then we just need only need minimise the function $F_i(\boldsymbol{z})$, since if we have $F_i^{\downarrow} < \log(r)$, then we know that $\log(r) \in [F_i^{\downarrow}, F_i^{\uparrow}]$, so the global maximum is attained. If $F_i^{\downarrow} > \log(r)$, then the maximum of $G(x)$ just occurs at $F_i^{\downarrow}$ and we can avoid the need to maximise the function $F_i(\boldsymbol{z})$. However, if conversely, we evaluate $F_i(\boldsymbol{z})$ at $\boldsymbol{z} = \hat{\boldsymbol{z}}$ and we have $F_i(\hat{\boldsymbol{z}}) < \log(r)$, then we just need to maximise $F_i(\boldsymbol{z})$ for $\boldsymbol{z} \in R^{(z)}$ and apply the inverse of the same trick. To summarise, in order to find $\max_{\boldsymbol{z} \in R^{(z)}} \{G_r(\boldsymbol{z})\}$, we can apply Algorithm B.7.1.

Algorithm B.7.1 Computing the local bounds of $G_r(\boldsymbol{z})$ given in (B.42) for $\boldsymbol{z} \in R^{(z)}$.

1. Compute $F_i(\hat{\boldsymbol{z}})$ at some arbitrary value $\hat{\boldsymbol{z}} \in R^{(z)}$.
2. If $F_i(\hat{\boldsymbol{z}}) > \log(r)$:
   (a) Compute $F_i^{\downarrow} := \min_{\boldsymbol{z} \in R^{(z)}} F_i(\boldsymbol{z})$.
   (b) $$\max_{\boldsymbol{z} \in R^{(z)}} \left\{ \frac{\exp\left((X_i \boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)}{\left(\exp\left((X_i \boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)+r\right)^2} \right\} = \begin{cases} \frac{1}{4r} & \text{if } F_i^{\downarrow} < \log(r), \\ G(F_i^{\downarrow}) = \frac{\exp\left(F_i^{\downarrow}\right)}{\left(\exp\left(F_i^{\downarrow}\right)+r\right)^2} & \text{otherwise.} \end{cases}$$
3. Else (if $F_i(\hat{\boldsymbol{z}}) < \log(r)$):
   (a) Compute $\max_{\boldsymbol{z} \in R^{(z)}} F_i(\boldsymbol{z})$.
   (b) $$\max_{\boldsymbol{z} \in R^{(z)}} \left\{ \frac{\exp\left((X_i \boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)}{\left(\exp\left((X_i \boldsymbol{\Lambda}_c^{\frac{1}{2}})\boldsymbol{z}\right)+r\right)^2} \right\} = \begin{cases} \frac{1}{4r} & \text{if } F_i^{\uparrow} > \log(r), \\ G(F_i^{\uparrow}) = \frac{\exp\left(F_i^{\uparrow}\right)}{\left(\exp\left(F_i^{\uparrow}\right)+r\right)^2} & \text{otherwise.} \end{cases}$$

Bibliography

Yacine Aït-Sahalia. Closed-form likelihood expansions for multivariate diffusions, 2008.

Isabelle Albert, Sophie Donnet, Chantal Guihenneuc-Jouyaux, Samantha Low-Choy, Kerrie Mengersen, and Judith Rousseau. Combining Expert Opinions in Prior Elicitation. *Bayesian Analysis*, 7(3):503–532, 2012.

Christophe Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

Soren Asmussen, Peter Glynn, and Jim Pitman. Discretization error in simulation of one-dimensional reflecting Brownian motion. *The Annals of Applied Probability*, 5(4):875–896, 1995.

Jack Baker, Paul Fearnhead, Emily B. Fox, and Christopher Nemeth. Control variates for stochastic gradient MCMC. *Statistics and Computing*, 29(3):599–615, 2019.

Stefan Banach and Hugo Steinhaus. Sur le principe de la condensation de singularités. *Fundamenta Mathematicae*, 1(9):50–61, 1927.

Rémi Bardenet, Arnaud Doucet, and Christopher C. Holmes. On Markov chain Monte Carlo methods for tall data. *Journal of Machine Learning Research*, 18(47), 2017.

Atilim G. Baydin, Barak A. Pearlmutter, Alexey A. Radul, and Jeffrey M. Siskind. Automatic Differentiation in Machine Learning: a Survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, New York, 1980.

Alexandros Beskos and Gareth O. Roberts. Exact Simulation of Diffusions. *The Annals of Applied Probability*, 15(4):2422–2444, 2005.

Alexandros Beskos, Omiros Papaspiliopoulos, and Gareth O. Roberts. Retrospective exact simulation of diffusion sample paths with applications. *Bernoulli*, 12(6):1077–1098, 2006a.

Alexandros Beskos, Omiros Papaspiliopoulos, Gareth O. Roberts, and Paul Fearnhead. Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68 (3):333–382, 2006b.

Alexandros Beskos, Omiros Papaspiliopoulos, and Gareth O. Roberts. A Factorisation of Diffusion Measure and Finite Sample Path Constructions. *Methodology and Computing in Applied Probability*, 10(1):85–104, 2008.

Alexandros Beskos, Stefano Peluchetti, and Gareth O. Roberts. *varepsilon*-Strong simulation of the Brownian path. *Bernoulli*, 18(4):1223–1248, 2012.

Joris Bierkens, Paul Fearnhead, and Gareth O. Roberts. The Zig-Zag Process and Super-Efficient Sampling for Bayesian Analysis of Big Data. *The Annals of Statistics*, 47(3):1288–1320, 2019.

Joris Bierkens, Sebastiano Grazzi, Kengo Kamatani, and Gareth O. Roberts. The Boomerang Sampler. In *International Conference on Machine Learning*, pages 908–918. PMLR, 2020.

Fischer Black and Myron Scholes. The Pricing of Options and Corporate Liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Miodrag Bolic, Petar M. Djuric, and Sangjin Hong. Resampling algorithms and architectures for distributed particle filters. *IEEE Transactions on Signal Processing*, 53(7):2442–2450, 2005.

Alexandre Bouchard-Côté, Sriram Sankararaman, and Michael I. Jordan. Phylogenetic Inference via Sequential Monte Carlo. *Systematic Biology*, 61(4):579–593, 2012.

Alexandre Bouchard-Côté, Sebastian J. Vollmer, and Arnaud Doucet. The Bouncy Particle Sampler: A Non-Reversible Rejection-Free Markov Chain Monte Carlo Method. *Journal of the American Statistical Association*, 113(522):855–867, 2018.

George E.P. Box and Mervin E. Muller. A note on the generation of normal random deviates. *Annals of Mathemaical Statistics*, 29:610–613, 1958.

Richard P. Brent. *Algorithms for minimization without derivatives*. Prentice-Hall, 1973.

Robert H. Cameron and William T. Martin. Transformations of Weiner integrals under translations. *Annals of Mathematics*, 45(2):386–396, 1944.

Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1), 2017.

James Carpenter, Peter Clifford, and Paul Fearnhead. An improved particle filter for nonlinear problems. *IEE Proceedings-Radar, Sonar and Navigation*, 146(1):2–7, 1999.

Bruno Casella and Gareth O. Roberts. Exact simulation of jump-diffusion processes with monte carlo applications. *Methodology and Computing in Applied Probability*, 13:449–473, 2011.

Kalok C. Chan, G. Andrew Karolyi, Francis A. Longstaff, and Anthony B. Sanders. An empirical comparison of alternative models of the short-term interest rate. *The Journal of Finance*, 47(3): 1209 –1227, 1992.

Ryan S.Y. Chan, Murray Pollock, Adam M. Johansen, and Gareth O. Roberts. Divide-and-Conquer Monte Carlo Fusion. Statistics e-print 2110.07265, arXiv, 2021.

Wu Changye and Christian P. Robert. Parallelising MCMC via Random Forests. Statistics e-print 1911.09698, arXiv, 2021.

Nan Chen and Zhengyu Huang. Localization and Exact Simulation of Brownian Motion Driven Stochastic Differential Equations. *Mathematics of Operations Research*, 38(3):591–616, 2013.

Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

Nicolas Chopin. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *The Annals of Statistics*, 32(6):2385–2411, 2004.

Nicolas Chopin and Omiros Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*, volume 4. Springer, 2020.

Sebastian Claici, Mikhail Yurochkin, Soumya Ghosh, and Justin Solomon. Model Fusion with Kullback-Leibler Divergence. In *International Conference on Machine Learning*, pages 2038–2047. PMLR, 2020.

Samuel N. Cohen and Robert J. Elliott. *Stochastic Calculus and Applications*, volume 2. Springer, 2015.

Alice Corbella, Simon E.F. Spencer, and Gareth O. Roberts. Automatic Zig-Zag sampling in practice. Statistics e-print 2206.11410, arXiv, 2022.

David R. Cox and Valerie Isham. *Point Processes*. Chapman and Hall, 1st edition, 1980.

Drew Creal. A Survey of Sequential Monte Carlo Methods for Economics and Finance. *Econometric Reviews*, 31(3):245–296, 2012.

Marco Cuturi and Arnaud Doucet. Fast computation of Wasserstein barycenters. In *International Conference on Machine Learning*, pages 685–693. PMLR, 2014.

Didier Dacunha-Castelle and Danielle Florens-Zmirou. Estimation of the Coefficients of a Diffusion from Discrete Observations. *Stochastics: An International Journal of Probability and Stochastic Processes*, 19(4):263–284, 1986.

Hongsheng Dai. Exact simulation for diffusion bridges: An adaptive approach. *Journal of Applied Probability*, 51(2):346–358, 2014.

Hongsheng Dai, Murray Pollock, and Gareth O. Roberts. Monte Carlo Fusion. *Journal of Applied Probability*, 56(1):174–191, 2019.

Hongsheng Dai, Murray Pollock, and Gareth O. Roberts. Bayesian Fusion: Scalable unification of distributed statistical analyses. Statistics e-print 2102.02123, arXiv, 2021.

Daryl J. Daley and David Vere-Jones. *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. Springer, 2003.

Daryl J. Daley and David Vere-Jones. *An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure*. Springer, 2008.

Luc Devroye. Generating the maximum of independent identically distributed random variables. *Computers & Mathematics with Applications*, 6(3):305–315, 1980.

Luc Devroye. *Non-Uniform Random Variate Generation*. Springer, New York, 1986.

Randal Douc, Olivier Cappé, and Eric Moulines. Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 64–69. IEEE, 2005.

Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In *The Oxford Handbook of Nonlinear Filtering*, pages 656–704. Oxford University Press, 2011.

Arnaud Doucet and Anthony Lee. Sequential Monte Carlo Methods. In *Handbook of Graphical Models*, pages 165–188. CRC Press, 2018.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to Sequential Monte Carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

Dirk Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013.

Albert Einstein. Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. *Annalen der physik*, 822(8):549–560, 1905.

Reihaneh Entezari, Radu V. Craiu, and Jeffrey S. Rosenthal. Likelihood Inflating Sampling Algorithm. *Canadian Journal of Statistics*, 46(1):147–175, 2018.

Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2):113–127, 2014.

Paul Fearnhead. *Sequential Monte Carlo methods in filter theory.* PhD thesis, Merton College, University of Oxford, 1998.

Paul Fearnhead, Omiros Papaspiliopoulos, and Gareth O. Roberts. Particle Filters for Partially Observed Diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):755–777, 2008.

Paul Fearnhead, Omiros Papaspiliopoulos, Gareth O. Roberts, and Andrew M. Stuart. Random-weight particle filtering of continuous time processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):497–512, 2010.

Axel Finke, Arnaud Doucet, and Adam M. Johansen. Limit Theorems for Sequential MCMC Methods. *Advances in Applied Probability*, 52(2):377–403, 2020.

Joseph L. Fleiss. The statistical basis of meta-analysis. *Statistical Methods in Medical Research*, 2 (2):121–145, 1993.

Christian Genest and James V. Zidek. Combining Probability Distributions: A Critique and an Annotated Bibliography. *Statistical Science*, 1(1):114–135, 1986.

Mathieu Gerber, Nicolas Chopin, and Nick Whiteley. Negative association, ordering and convergence of resampling methods. *Annals of Statistics*, 47(4):2236–2260, 2019.

Charles J. Geyer. Markov chain Monte Carlo maximum likelihood. *Computing Science and Statistics*, 23:156–163, 1991.

Charles J. Geyer and Elizabeth A. Thompson. Annealing Markov Chain Monte Carlo with Applications to Ancestral Inference. *Journal of the American Statistical Association*, 90(431):909–920, 1995.

Kay Giesecke and Dmitry Smelov. Exact sampling of jump diffusions. *Operations Research*, 61(4): 894–907, 2013.

Wally R. Gilks, Andrew Thomas, and David J. Spiegelhalter. A language and program for complex Bayesian modelling. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 43(1): 169–177, 1994.

Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

Igor Vladimirovich Girsanov. On transforming a certain class of stochastic processes by absolutely continuous substitution of measures. *Theory of Probability & Its Applications*, 5(3):285–301, 1960.

Simon Godsill, Peter Rayner, and Olivier Cappé. Digital Audio Restoration. In *Applications of Digital Signal Processing to Audio and Acoustics*, pages 133–194. Springer, 2002.

Gerald Goertzel. Quota sampling and importance functions in stochastic solution of particle problems. Technical report, Oak Ridge, Tennessee: U.S. Atomic Energy Commission, Technical Information Division, 1949. URL `https://www.osti.gov/biblio/4405957`.

Andrew Golightly and Darren J. Wilkinson. Bayesian sequential inference for stochastic kinetic biochemical network models. *Journal of Computational Biology*, 13(3):838–851, 2006.

Neil J. Gordon, David J. Salmond, and Adrian F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2): 107–113, 1993.

Robert J.B. Goudie, Anne M. Presanis, David Lunn, Daniela De Angelis, and Lorenz Wernisch. Joining and splitting models with markov melding. *Bayesian Analysis*, 14(1):81, 2019.

W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Tomoyuki Higuchi. Monte Carlo filter using the genetic algorithm operators. *Journal of Statistical Computation and Simulation*, 59(1):1–23, 1997.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *Journal of Machine Learning Research*, 2013.

Jean Jacod and Philip Protter. *Discretization of processes*, volume 67. Springer Science & Business Media, 2012.

Richard A. Johnson. Asymptotic expansions associated with posterior distributions. *The Annals of Mathematical Statistics*, pages 851–864, 1970.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, 1999.

Herman Kahn. Stochastic (Monte Carlo) Attenuation Analysis. Technical report, The RAND Corporation, Santa Monica, California, 1949.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Ioannis Karatzas and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, New York, 2nd edition, 1991.

Ioannis Karatzas and Steven E. Shreve. *Methods of mathematical finance*, volume 39. Springer, 1998.

Heysem Kaya, Pınar Tüfekci, and Fikret S. Gürgen. Local and global learning methods for predicting power of a combined gas & steam turbine. In *Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE*, pages 13–18, 2012.

Motoo Kimura and Tomoko Ohta. *Theoretical Aspects of Population Genetics*. Princeton University Press, Princeton, 4th edition, 1971.

John F.C. Kingman. *Poisson Processes*. Clarendon Press, Oxford, 1st edition, 1992.

Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

Peter E. Kloeden and Eckhard Platen. Stochastic differential equations. In *Numerical Solution of Stochastic Differential Equations*, pages 103–160. Springer, 1992.

Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, Berlin, 1995.

Augustine Kong. A note on importance sampling using standardized weights. *University of Chicago, Dept. of Statistics, Tech. Rep*, 348, 1992.

Augustine Kong, Jun S. Liu, and Wing Hung Wong. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.

Xiangshun Kong and Wei Zheng. Design Based Incomplete U-Statistics. *Statistica Sinica*, 31: 1593–1618, 2021.

Juan Kuntz, Francesca R. Crucinio, and Adam M. Johansen. Product-form estimators: exploiting independence to scale up Monte Carlo. e-print 2102.11575, arXiv, 2021a.

Juan Kuntz, Francesca R. Crucinio, and Adam M. Johansen. The Divide-and-Conquer Sequential Monte Carlo algorithm: Theoretical properties and limit theorems. e-print 2110.15782, arXiv, 2021b.

Lucien Le Cam. *Asymptotic Methods in Statistical Decision Theory*. Springer Science & Business Media, New York, 1986.

Anthony Lee, Christopher Yau, Michael B. Giles, Arnaud Doucet, and Christopher C. Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 19(4):769–789, 2010.

Cheng Li, Sanvesh Srivastava, and David B. Dunson. Simple, Scalable and Accurate Posterior Interval Estimation. *Biometrika*, 104(3):665–680, 2017.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated Learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

Fredrik Lindsten, Adam M. Johansen, Christian A. Naesseth, Bonnie Kirkpatrick, Thomas B. Schön, John A.D. Aston, and Alexandre Bouchard-Côté. Divide-and-Conquer with Sequential Monte Carlo. *Journal of Computational and Graphical Statistics*, 26(2):445–458, 2017.

Jun S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer Science & Business Media, 2001.

Jun S. Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

Hedibert F. Lopes and Ruey S. Tsay. Particle Filters and Bayesian Inference in Financial Econometrics. *Journal of Forecasting*, 30(1):168–209, 2011.

Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient MCMC. *Advances in Neural Information Processing Systems*, 28:2917–2925, 2015.

Dougal Maclaurin and Ryan P. Adams. Firefly Monte Carlo: Exact MCMC with Subsets of Data. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2014.

Andrew A. Manderson and Robert J.B. Goudie. A numerically stable algorithm for integrating Bayesian models using Markov melding. *Statistics and Computing*, 32(2):1–13, 2022.

Enzo Marinari and Giorgio Parisi. Simulated Tempering: a new Monte Carlo scheme. *EPL (Europhysics Letters)*, 19(6):451, 1992.

Gisiro Maruyama. Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo*, 4(1):48–90, 1955.

Simon Maskell and Neil Gordon. A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking. In *IEE Target Tracking: Algorithms and Applications (Ref. No. 2001/174)*, pages 2–1. IET, 2002.

Harley H. McAdams and Adam Arkin. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Sciences*, 94(3):814–819, 1997.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

Robert C. Merton. Theory of rational option pricing. *The Bell Journal of Economics and Management Science*, pages 141–183, 1973.

Robert C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1-2):125–144, 1976.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Stanislav Minsker, Sanvesh Srivastava, Lizhen Lin, and David B. Dunson. Scalable and Robust Bayesian Inference via the Median Posterior. In *International Conference on Machine Learning*, pages 1656–1664, 2014.

Stanislav Minsker, Sanvesh Srivastava, Lizhen Lin, and David B. Dunson. Robust and scalable Bayes via a median of subset posterior measures. *The Journal of Machine Learning Research*, 18(1):4488–4527, 2017.

Alexey Miroshnikov and Erin M. Conlon. ParallelMCMCcombine: an R package for Bayesian Methods for Big Data and Analytics. *PloS one*, 9(9):e108425, 2014.

Pierre Del Moral. Feynman-Kac Formulae. In *Feynman-Kac Formulae*, pages 47–93. Springer, 2004.

Lawrence M. Murray, Anthony Lee, and Pierre E. Jacob. Parallel resampling in the particle filter. *Journal of Computational and Graphical Statistics*, 25(3):789–805, 2016.

Willie Neiswanger, Chong Wang, and Eric P. Xing. Asymptotically Exact, Embarrassingly Parallel MCMC. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 623–632, Arlington, Virginia, USA, 2014. AUAI Press.

Christopher Nemeth and Chris Sherlock. Merging MCMC Subposteriors through Gaussian-process Approximations. *Bayesian Analysis*, 13(2):507–530, 2018.

Bernt Øksendal. *Stochastic Differential Equations*. Springer, 6th edition, 2007.

Rihui Ou, Deborshee Sen, and David B. Dunson. Scalable Bayesian inference for time series via divide-and-conquer. Statistics e-print 2106.11043, arXiv, 2021.

Gabriel Peyré and Marco Cuturi. Computational Optimal Transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

Martyn Plummer et al. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, volume 124, pages 1–10. Vienna, Austria., 2003.

Murray Pollock. *Some Monte Carlo Methods for Jump Diffusions*. PhD thesis, Department of Statistics, University of Warwick, 2013.

Murray Pollock, Adam M. Johansen, and Gareth O. Roberts. On the exact and ε-strong simulation of (jump) diffusions. *Bernoulli*, 22(2):794–856, 2016.

Murray Pollock, Paul Fearnhead, Adam M. Johansen, and Gareth O. Roberts. Quasi-stationary Monte Carlo and the ScaLE algorithm. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 82(5):1167–1221, 2020.

Klaus Pötzelberger and Liqun Wang. Boundary crossing probability for Brownian motion. *Journal of Applied Probability*, 38(1):152–164, 2001.

Matias Quiroz, Robert Kohn, Mattias Villani, and Minh-Ngoc Tran. Speeding up MCMC by efficient data subsampling. *Journal of the American Statistical Association*, 2018.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022. URL `http://www.R-project.org/`.

Maxim Rabinovich, Elaine Angelino, and Michael I. Jordan. Variational Consensus Monte Carlo. *Advances in Neural Information Processing Systems*, 28, 2015.

Carl Edward Rasmussen. Gaussian Processes in Machine Learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

Carl Edward Rasmussen and Christopher K. Williams. *Gaussian Processes for Machine Learning*. MIT Press Cambridge, MA, 2006.

Lewis J. Rendell, Adam M. Johansen, Anthony Lee, and Nick Whiteley. Global Consensus Monte Carlo. *Journal of Computational and Graphical Statistics*, 30(2):249–259, 2020.

Daniel Revuz and Marc Yor. *Continuous Martingales and Brownian Motion*, volume 293. Springer-Verlag, Berlin, 3rd edition, 1991.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Christian Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Science & Business Media, 2013.

L. Chris G. Rogers and David Williams. *Diffusions, Markov processes and martingales: Volume 2, Itô calculus*, volume 2. Cambridge University Press, 2000.

Benjamin Schäfer, Carsten Grabow, Sabine Auer, Jürgen Kurths, Dirk Witthaut, and Marc Timme. Taming instabilities in power grid networks by decentralized control. *The European Physical Journal Special Topics*, 225(3):569–582, 2016.

David W. Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, New York, 1992.

Steven L. Scott. Comparing consensus Monte Carlo strategies for distributed Bayesian computation. *Brazilian Journal of Probability and Statistics*, 31(4):668–685, 2017.

Steven L. Scott, Alexander W. Blocker, Fernando V. Bonassi, Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayes and Big Data: The Consensus Monte Carlo Algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88, 2016.

Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

Sidak Pal Singh and Martin Jaggi. Model Fusion via Optimal Transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

Sanvesh Srivastava, Volkan Cevher, Quoc Dinh, and David B. Dunson. WASP: Scalable Bayes via barycenters of subset posteriors. In *Artificial Intelligence and Statistics*, pages 912–920, 2015.

Sanvesh Srivastava, Cheng Li, and David B. Dunson. Scalable Bayes via barycenter in Wasserstein space. *The Journal of Machine Learning Research*, 19(1):312–346, 2018.

T-J Stieltjes. Recherches sur les fractions continues. *Annales de la Faculté des sciences de Toulouse: Mathématiques*, 8(4):1–122, 1894.

Nicholas G. Tawn and Gareth O. Roberts. Accelerating parallel tempering: Quantile Tempering Algorithm (QuanTA). *Advances in Applied Probability*, 51(3):802–834, 2019.

Pınar Tüfekci. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, 60:126–140, 2014.

Jaco Vermaak, Christophe Andrieu, Arnaud Doucet, and Simon J. Godsill. Particle methods for Bayesian modelling and enhancement of speech signals. *IEEE Transactions on Speech and Audio Processing*, 10(3):173–185, 2002.

Cédric Villani. *Optimal Transport: Old and New*, volume 338. Springer, 2009.

John Von Neumann. Various techniques used in connection with random digits. *National Bureau of Standards - Applied Mathematics Series*, 12(36-38):3, 1951.

Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

Chunlei Wang and Sanvesh Srivastava. Divide-and-Conquer Bayesian Inference in Hidden Markov Models. Statistics e-print 2105.14395, arXiv, 2021.

Liqun Wang and Klaus Pötzelberger. Boundary crossing probability for Brownian motion and general boundaries. *Journal of Applied Probability*, 34(1):54–65, 1997.

Xiangyu Wang and David B. Dunson. Parallelizing MCMC via Weierstrass Sampler. Statistics e-print 1312.4605, arXiv, 2013.

Xiangyu Wang, Fangjian Guo, Katherine A. Heller, and David B. Dunson. Parallelizing MCMC with random partition trees. *Advances in Neural Information Processing Systems*, 28, 2015.

Karl Weierstrass. Über die analytische Darstellbarkeit sogenannter willkürlicher Functionen einer reellen Veränderlichen. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin*, 2:633–639, 1885.

Max Welling and Yee W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688. Citeseer, 2011.

Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, 1994.

Hadley Wickham. *nycflights13: Flights that Departed NYC in 2013*, 2021.

Jingnan Xue and Faming Liang. Double-parallel Monte Carlo for Bayesian analysis of big data. *Statistics and Computing*, 29(1):23–32, 2019.

Toshio Yamada and Shinzo Watanabe. On the uniqueness of solutions of stochastic differential equations. *Journal of Mathematics of Kyoto University*, 11(1):155–167, 1971.

I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2): 2473–2480, 2009.

Sinan Yıldırım and Beyza Ermiş. Exact MCMC with differentially private moves. *Statistics and Computing*, 29(5):947–963, 2019.

Yan Zhou, Adam M. Johansen, and John A.D. Aston. Toward Automatic Model Comparison: An Adaptive Sequential Monte Carlo Approach. *Journal of Computational and Graphical Statistics*, 25(3):701–726, 2016.