J. R. Statist. Soc. B (2020)
82, *Part* 4, *pp.* 1–44

Quasi-stationary Monte Carlo methods and the ScaLE algorithm

Murray Pollock,

University of Warwick, Coventry, UK

Paul Fearnhead,

University of Lancaster, UK

and Adam M. Johansen and Gareth O. Roberts

University of Warwick, Coventry, UK

[*Read before* The Royal Statistical Society *at a meeting organized by the* Research Section *on Wednesday, April 22nd, 2020*, Professor G. P. Nason *in the Chair*]

Summary. The paper introduces a class of Monte Carlo algorithms which are based on the simulation of a Markov process whose quasi-stationary distribution coincides with a distribution of interest. This differs fundamentally from, say, current Markov chain Monte Carlo methods which simulate a Markov chain whose stationary distribution is the target. We show how to approximate distributions of interest by carefully combining sequential Monte Carlo methods with methodology for the exact simulation of diffusions. The methodology that is introduced here is particularly promising in that it is applicable to the same class of problems as gradient-based Markov chain Monte Carlo algorithms but entirely circumvents the need to conduct Metropolis–Hastings type accept–reject steps while retaining *exactness*: the paper gives theoretical guarantees ensuring that the algorithm has the correct limiting target distribution. Furthermore, this methodology is highly amenable to 'big data' problems. By employing a modification to existing naive subsampling and control variate techniques it is possible to obtain an algorithm which is still exact but has *sublinear* iterative cost as a function of data size.

Keywords: Control variates; Importance sampling; Killed Brownian motion; Langevin diffusion; Markov chain Monte Carlo sampling; Quasi-stationarity; Sequential Monte Carlo methods

1. Introduction

Advances in methodology for the collection and storage of data have led to scientific challenges and opportunities in a wide array of disciplines. This is particularly so in statistics as the complexity of appropriate statistical models often increases with data size. Many current state of the art statistical methodologies have algorithmic cost that scales poorly with increasing volumes of data. As noted by Jordan (2013),

> 'many statistical procedures either have unknown runtimes or runtimes that render the procedure unusable on large-scale data'

and this has resulted in a proliferation in the literature of methods

> '...which may provide no statistical guarantees and which in fact may have poor or even disastrous statistical properties'.

Address for correspondence: Gareth O. Roberts, Department of Statistics, University of Warwick, Zeeman Building, Coventry, CV4 7AL, UK.
E-mail: gareth.o.roberts@warwick.ac.uk

 1369–7412/20/82000

This is particularly keenly felt in computational and Bayesian statistics, in which the standard computational tools are Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC) methods and their many variants (see for example Robert and Casella (2004)). MCMC methods are *exact* in the (weak) sense that they construct Markov chains which have the correct limiting distribution. Although MCMC methodology has had considerable success in being applied to a wide variety of substantive areas, they are not well suited to this new era of 'big data' as their computational cost will increase at least linearly with the number of data points. For example, each iteration of the Metropolis–Hastings algorithm requires evaluating the likelihood, the calculation of which, in general, scales linearly with the number of data points. The motivation behind the work that is presented in this paper is on developing Monte Carlo methods that are exact, in the same sense as MCMC methods, but that have a computational cost per effective sample size that is sublinear in the number of data points.

To date, the success of methods that aim to adapt MCMC sampling to reduce its algorithmic cost has been mixed and has invariably led to a compromise on exactness—such methodologies generally construct a stochastic process with limiting distribution which is (at least hopefully) close to the desired target distribution. Broadly speaking these methods can be divided into three classes of approach: 'divide-and-conquer' methods, 'exact subsampling' methods and 'approximate subsampling' methods. Each of these approaches has its own strengths and weaknesses which will be briefly reviewed in the following paragraphs.

Divide-and-conquer methods (for instance, Neiswanger *et al.* (2014), Wang and Dunson (2013), Scott *et al.* (2016) and Minsker *et al.* (2014)) begin by splitting the data set into a large number of smaller data sets (which may or may not overlap). Inference is then conducted on these smaller data sets and resulting estimates are combined in some appropriate manner. A clear advantage of such an approach is that inference on each small data set can be conducted independently, and in parallel, and so if one had access to a large cluster of computing cores then the computational cost could be significantly reduced. The primary weakness of these methods is that the recombination of the separately conducted inferences is inexact. All current theory is asymptotic in the number of data points, n (Neiswanger *et al.*, 2014; Li *et al.*, 2017). For these asymptotic regimes the posterior will tend to a Gaussian distribution (Johnson, 1970), and it is questionable whether divide-and-conquer methods offer an advantage over simple approaches such as a Laplace approximation to the posterior (Bardenet *et al.*, 2017). Most results on convergence rates (e.g. Srivastava *et al.* (2016)) have rates that are of order $\mathcal{O}(m^{-1/2})$, where m is the number of data points in each subset. As such they are no stronger than convergence rates for analysing just a single batch. One exception is in Li *et al.* (2017), where convergence rates of $\mathcal{O}(n^{-1/2})$ are obtained, albeit under strong conditions. However, these results relate only to estimating marginal posterior distributions, rather than the full posterior.

Subsampling methods are designed so that each iteration requires access to only a subset of the data. Exact approaches in this vein typically require subsets of the data of random size at each iteration. One approach is to construct unbiased estimators of pointwise evaluations of the target density by using subsets of the data, which could then be embedded within the pseudomarginal MCMC framework developed by Andrieu and Roberts (2009). Unfortunately, the construction of such positive unbiased estimators is not possible in general (Jacob and Thiéry, 2015) and such methods often require both bounds on, and good analytical approximations of, the likelihood (Maclaurin and Adams, 2015).

More promising practical results have been obtained by approximate subsampling approaches. These methods use subsamples of the data to estimate quantities such as acceptance probabilities

(Nicholls *et al*., 2012; Korattikara *et al*., 2014; Bardenet *et al*., 2014), or the gradient of the posterior, that are used within MCMC algorithms. These estimates are then used in place of the true quantities. Although this can lead to increases in computational efficiency, the resulting algorithms no longer target the true posterior. The most popular of these algorithms is the stochastic gradient Langevin dynamics algorithm of Welling and Teh (2011). This approximately samples a Langevin diffusion which has the posterior as its stationary distribution. To do this requires first approximating the continuous time diffusion by a discrete time Markov process, and then using subsampling estimates of the gradient of the posterior within the dynamics of this discrete time process. This idea has been extended to approximations of other continuous time dynamics that target the posterior (Ahn *et al*., 2012; Chen *et al*., 2014; Ma *et al*., 2015).

Within these subsampling methods it is possible to tune the subsample size, and sometimes the algorithm's step size, to control the level of approximation. This leads to a trade-off, whereby increasing the computational cost of the algorithm can lead to samplers that target a closer approximation to the true posterior. There is also substantial theory quantifying the bias in, say, estimates of posterior means, that arise from these methods (Teh *et al*., 2016; Vollmer *et al*., 2016; Chen *et al*., 2015; Huggins and Zou, 2017; Dalalyan and Karagulyan, 2019), and how this depends on the subsample size and step size. However, although they often work well in practice it can be difficult to know just how accurate the results are for any given application. Furthermore, many of these algorithms still have a computational cost that increases linearly with data size (Bardenet *et al*., 2017; Nagapetyan *et al*., 2017; Baker *et al*., 2019).

The approach to the problem of big data that is proposed here is a significant departure from the current literature. Rather than building our methodology on the stationarity of appropriately constructed Markov chains, a novel approach based on the *quasi-limiting distribution* of suitably constructed stochastically weighted diffusion processes is developed. A quasi-limiting distribution for a Markov process X with respect to a Markov stopping time ζ is the limit of the distribution of $X_t|\zeta > t$ as $t \to \infty$, and such distributions are automatically *quasi-stationary distributions*; see Collet *et al*. (2013); this concept is completely unrelated to the popular area of quasi-Monte-Carlo methods. These *quasi-stationary Monte Carlo (QSMC) methods* that have been developed can be used for a broad range of Bayesian problems (of a similar type to MCMC methods) and exhibit interesting and differing algorithmic properties. The QSMC methods developed are *exact* in the same (weak) sense as MCMC methods, in that they give the correct (quasi-)limiting distribution. There are a number of possible implementations of the theory which open up interesting avenues for future research, in terms of branching processes, by means of stochastic approximation methods, or (as outlined in this paper) SMC methods. We note that the use of continuous time SMC and related algorithms to obtain approximations of large time limiting distributions of processes conditioned to remain alive has also been explored in settings in which a quantity of interest admits a natural representation of this form (see Del Moral and Miclo (2003), Rousset (2006) and related work in the physics literature, such as Giardina *et al*. (2011) and references therein); a substantial difference between these and the present work is that the QSMC methods that are described here construct a process for which quite a general distribution of interest is the quasi-stationary distribution and entirely avoid time discretization errors. One particularly interesting difference between our class of Monte Carlo and MCMC algorithms is that QSMC methods enable us to circumvent entirely the Metropolis–Hastings-type accept–reject steps, while still retaining theoretical guarantees that the correct limiting target distribution is recovered. In the case of big data problems, this removes one of the fundamental $\mathcal{O}(n)$ bottlenecks in computation.

QSMC methods can be applied in big data contexts by using a novel subsampling approach. We call the resulting algorithm *(ScaLE)*, for *scalable Langevin exact algorithm*. The name refers to the 'Langevin' diffusion which is used in the mathematical construction of the algorithm, although it should be emphazised that it is not explicitly used in the algorithm itself. As shown in Section 4, the approach to subsampling that is adopted here can potentially decrease the computational complexity of each iteration of a QSMC algorithm to be $\mathcal{O}(1)$. Furthermore, for a rejection sampler implementation of a QSMC sampling, the use of subsampling introduces no additional error—as the rejection sampler will sample from the same stochastic process, a killed Brownian motion, regardless of whether subsampling is used or not. There can be a computational cost of using subsampling, as the number of rejection sampler iterations that are needed to simulate the killed Brownian motion for a given time interval will increase. However, this paper will show that, by using control variates (Bardenet *et al.*, 2017) to reduce the variability of subsampling estimators of features of the posterior, the on-going algorithm computational cost will be $\mathcal{O}(1)$. Constructing the control variates involves a preprocessing step whose cost is $\mathcal{O}(n)$ (at least in the case of posterior contraction at rate $n^{-1/2}$) but after this preprocessing step the resulting cost of ScaLE per effective sample size can be $\mathcal{O}(1)$. The importance of using control variates to obtain a computational cost that is sublinear in n is consistent with other recent work on scalable Monte Carlo methods (Huggins and Zou, 2017; Bierkens *et al.*, 2019; Quiroz *et al.*, 2016; Dubey *et al.*, 2016; Nagapetyan *et al.*, 2017; Baker *et al.*, 2019).

The next section presents the main result that motivates development of QSMC methods. The following sections then provide detail on how to implement QSMC algorithms in practice, and how and why they are amenable to use with subsampling ideas. For clarity of presentation, much of the technical and algorithmic detail has been suppressed but can be found in the appendices.

2. Quasi-stationary Monte Carlo sampling

Given a target density π on $\mathbb{R}^d$, traditional (i.e. Metropolis–Hastings type) MCMC methods propose at each iteration from Markov dynamics with proposal density $q(\mathbf{x}, \mathbf{y})$, 'correcting' its trajectory by either accepting the move with probability

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{1, \frac{\pi(\mathbf{y})q(\mathbf{y}, \mathbf{x})}{\pi(\mathbf{x})q(\mathbf{x}, \mathbf{y})}\right\}, \tag{1}$$

or rejecting the move and remaining at state $\mathbf{x}$. In QSMC sampling, rather than rejecting a move and staying at $\mathbf{x}$, the algorithm *kills* the trajectory entirely, according to probabilities which relate to the target density.

Simulation of a Markov process with killing inevitably leads to death of the process. Thus it is natural to describe the long-term behaviour of the process through its conditional distribution given that the process is still alive. The limit of this distribution is called the quasi-stationary distribution (see, for example, Collet *et al.* (2013)). The idea of QSMC sampling is to construct a Markov process whose quasi-stationary distribution is the distribution, $\pi(\mathbf{x})$, from which the user wishes to sample from. Simulations from such a process can then be used to approximate expectations with respect to $\pi(\mathbf{x})$ just as in MCMC sampling.

Although in principle QSMC methods can be used with any Markov process, this paper will work exclusively with killed Brownian motion as it has some convenient properties that can be exploited. Therefore let $\{\mathbf{X}_t, t \geqslant 0\}$ denote d-dimensional Brownian motion initialized at $\mathbf{X}_0 = \mathbf{x}_0$. Suppose that $\kappa(\mathbf{x})$ denotes a non-negative hazard rate at which the Brownian motion is killed when it is in state $\mathbf{x}$, and let ζ be the killing time itself. Finally define

$$\mu_t(\mathrm{d}\mathbf{x}) := \mathbb{P}(\mathbf{X}_t \in \mathrm{d}\mathbf{x} | \zeta > t): \tag{2}$$

the distribution of $\mathbf{X}_t$ given that it has not yet been killed. The limit of this distribution as $t \to \infty$ is the quasi-stationary distribution of the killed Brownian motion.

The aim will be to choose κ in such a way that μ_t converges to π and, with this in mind, we introduce the function $\phi : \mathbb{R}^d \to \mathbb{R}$

$$\phi(\mathbf{x}) := \frac{\|\nabla \log\{\pi(\mathbf{x})\}\|^2 + \Delta \log\{\pi(\mathbf{x})\}}{2} = \frac{\Delta \pi(\mathbf{x})}{2\pi(\mathbf{x})}, \tag{3}$$

where '$\|\cdot\|$' denotes the usual Euclidean norm and Δ the Laplacian operator. By further imposing the following condition the first theorem can be proved.

Condition 1. There is a constant $\Phi > -\infty$ such that $\Phi \leqslant \phi(\mathbf{u}) \; \forall \; \mathbf{u} \in \mathbb{R}^d$.

Theorem 1. Under the regularity conditions (26) and (27) in Appendix A, suppose that condition 1 holds and set

$$\kappa(\mathbf{x}) := \phi(\mathbf{x}) - \Phi \geqslant 0; \tag{4}$$

then it follows that μ_t converges in L^1 and pointwise to π.

For a proof of theorem 1, see appendix A.

Note that the regularity conditions in Appendix A are largely technical smoothness and other weak regularity conditions that are common in stochastic calculus. In contrast, condition 1 is necessary for us to be able to construct QSMC methods. However, since non-pathological densities on $\mathbb{R}^d$ are typically convex in the tails, the second identity in expression (3) demonstrates that condition 1 is almost always satisfied in real examples.

Theorem 1 can be exploited for statistical purposes by noting that, for some sufficiently large t^*, $\mu_t \approx \pi$ for $t > t^*$. Thus, given samples from μ_t for $t > t^*$, one would have an (approximate) sample from π. This is analogous to MCMC sampling, with t^* being the burn-in period; the only difference being the need to simulate from the distribution of the process conditionally on its not having died.

The next two sections describe how to simulate from μ_t. Firstly a description of how to simulate a killed Brownian motion process exactly in continuous time is provided. A naive approach to sample from μ_t is to simulate independent realizations of this killed Brownian motion, and to use the values at time t of those processes which have not yet died by time t. In practice this is impracticable, as the probability of survival will, in general, decay exponentially with t. To overcome this SMC methods are employed.

Both these steps introduce additional challenges that are not present within standard MCMC problems. Thus a natural question is why use QSMC methods at all? This is addressed in Section 4 where it is shown that simulating the killing events can be carried out using just subsamples of data. In fact subsamples of size 2 can be used without introducing any approximation into the dynamics of the killed Brownian motion.

3. Implementing quasi-stationary Monte Carlo algorithms

3.1. Simulating killed Brownian motion

Theorem 1 relates a target distribution of interest to the quasi-stationary distribution of an appropriate killed Brownian motion. To be able to simulate from this quasi-stationary distribution it is necessary to be able to simulate from killed Brownian motion.

To help to convey the main ideas, first consider the case where the killing rate $\kappa(\mathbf{x})$ is bounded above by some constant, K say. In this case it is possible to use thinning (see, for example,

Kingman (1992)) to simulate the time at which the process will die. This involves simulating the Brownian motion independently of a Poisson process with rate K. Each event of the Poisson process is a potential death event, and an appropriate Bernoulli variable then determines whether or not the death occurs. For an event at time ξ the probability that death occurs depends on the state of the Brownian motion at time ξ and is equal to $\kappa(\mathbf{x}_\xi)/K$. Thus to simulate the killed Brownian motion to time t the first step is to simulate all events in the Poisson process up to time t. Then, by considering the events in time order, it is straightforward to simulate the Brownian motion at the first event time and as a result to determine whether death occurs. If death does not occur, the next event time can be considered. This is repeated until either the process dies or the process has survived the last potential death event in $[0, t]$. If the latter occurs, Brownian motion can be simulated at time t without any further conditions.

This can be viewed as a rejection sampler to simulate from $\mu_t(\mathbf{x})$, the distribution of the Brownian motion at time t conditional on its surviving to time t. Any realization that has been killed is 'rejected' and a realization that is not killed is a draw from $\mu_t(\mathbf{x})$. It is easy to construct an importance sampling version of this rejection sampler. Assume that there are k events in the Poisson process before time t, and these occur at times $\xi_1, \ldots, \xi_k$. The Brownian motion path is simulated at each event time and at time t. The output of the importance sampler is the realization at time t, $\mathbf{x}_t$, together with an importance sampling weight that is equal to the probability that the path survives each potential death event:

$$W_t := \prod_{i=1}^{k} \frac{K - \kappa(\mathbf{x}_{\xi_i})}{K}.$$

Given a positive lower bound on the killing rate, $\kappa(\mathbf{x}) \geqslant K^{\downarrow}$ for all $\mathbf{x}$, it is possible to improve the computational efficiency of the rejection sampler by splitting the death process into a death process of rate $K^{\downarrow}$ and one of rate $\kappa(\mathbf{x}) - K^{\downarrow}$. Actual death occurs at the first event in either of these processes. The advantage of this construction is that the former death process is independent of the Brownian motion. Thus it is possible first to simulate whether or not death occurs in this process. If it does not we can then simulate, using thinning as above, a killed Brownian motion with rate $\kappa(\mathbf{x}) - K^{\downarrow}$. The latter will have a lower intensity and thus be quicker to simulate. Using the importance sampling version instead, events in a Poisson process of rate $K - K^{\downarrow}$, $\xi_1, \ldots, \xi_k$ say, are simulated, and our realization at time t is assigned a weight

$$W_t := \exp(-K^{\downarrow} t) \prod_{i=1}^{k} \frac{K - \kappa(\mathbf{x}_{\xi_i})}{K - K^{\downarrow}}.$$

This is particularly effective as the $\exp(-K^{\downarrow} t)$-term is a constant which will cancel on normalization of the importance sampling weights.

3.2. Simulating killed Brownian motion by using local bounds

The approach in Section 3.1 is not applicable in the absence of an upper bound on the killing rate. Even in situations where a global upper bound does exist, the resulting algorithm may be inefficient if this bound is large. Both of these issues can be overcome by using local bounds on the rate. For this section we shall work with the specific form of the killing rate in theorem 1, namely $\phi(\mathbf{x}) - \Phi$. The bounds that are used will be expressed in terms of bounds on $\phi(\mathbf{x})$.

Given an initial value for the Brownian motion, $\mathbf{x}_0$, define a hypercube which contains $\mathbf{x}_0$. In practice this cube is defined to be centred on $\mathbf{x}_0$ with a user-chosen side length (which may depend on $\mathbf{x}_0$). Denote the hypercube by $\mathcal{H}_1$, and assume that upper and lower bounds $U_{\mathbf{X}}^{(1)}$ and $L_{\mathbf{X}}^{(1)}$ respectively can be found for $\phi(\mathbf{x})$ with $\mathbf{x} \in \mathcal{H}_1$. The thinning idea of the previous section can

be used to simulate the killed Brownian motion while the process stays within $\mathcal{H}_1$. Furthermore it is possible to simulate the time at which the Brownian motion first leaves $\mathcal{H}_1$ and the value of the process when this happens (see Appendix C). Thus our approach is to use our local bounds on $\phi(\mathbf{x})$, and hence on the killing rate, to simulate the killing process while $\mathbf{x}$ remains in $\mathcal{H}_1$. If the process leaves $\mathcal{H}_1$ before t it is then necessary to define a new hypercube, $\mathcal{H}_2$ say, to obtain new local bounds on $\phi(\mathbf{x})$ for $\mathbf{x} \in \mathcal{H}_2$ and to repeat simulating the killing process by using these new bounds until the process either first leaves the hypercube or time t is reached.

The details of this approach are now given, describing the importance sampling version which is used later—though a rejection sampler can be obtained by using similar ideas. The first step is to calculate the hypercube $\mathcal{H}_1$ and the bounds $L_{\mathbf{X}}^{(1)}$ and $U_{\mathbf{X}}^{(1)}$. We then simulate the time and position at which $\mathbf{x}$ first leaves $\mathcal{H}_1$. We call this the *layer information* and denote it as $R_{\mathbf{X}}^{(1)} = (\tau_1, \mathbf{x}_{\tau_1})$. The notion of a layer for diffusions was formalized in Pollock *et al.* (2016), and we refer the interested reader there for further details. Next the possible killing events on $[0, t \wedge \tau_1]$ are generated by simulating events of a Poisson process of rate $U_{\mathbf{X}}^{(1)} - L_{\mathbf{X}}^{(1)}$: $\xi_1, \ldots, \xi_k$ say. The next step involves simulating the values of the Brownian motion at these event times (the simulation of which is conditional on $R_{\mathbf{X}}^{(1)}$—see Appendix C.2 and algorithm 5 there for a description of how this can be done). An incremental importance sampling weight for this segment of time is given as

$$W^{(1)} := \exp\{-(L_{\mathbf{X}}^{(1)} - \Phi)(t \wedge \tau_1)\} \prod_{i=1}^{k} \frac{U_{\mathbf{X}}^{(1)} - \phi(\mathbf{x}_{\xi_i})}{U_{\mathbf{X}}^{(1)} - L_{\mathbf{X}}^{(1)}}. \tag{5}$$

If $\tau_1 < t$, then this process is repeated with a hypercube centred on $\mathbf{x}_{\tau_1}$ until simulation to time t has been achieved. This gives successive incremental weights $W^{(2)}, W^{(3)}, \ldots$. A simulated value for the Brownian motion at time t is given, again simulated conditionally on the layer information for the current segment of time, and an importance sampling weight that is the product of the incremental weights associated with each segment of time. At time t, $J(t)$ incremental weights have been simulated, leading to the cumulative weight

$$W_t = \prod_{j=1}^{J(t)} W^{(j)}. \tag{6}$$

Full algorithmic details of the description above are given in algorithm 1 in Table 1. In practice every sample $\mathbf{X}_t$ will have an importance weight that shares a common constant of $\exp(\Phi t)$ in equation (6). As such it is omitted from algorithm 1 and the weights are asterisked to denote this. It is straightforward to prove that this approach gives valid importance sampling weights in the following sense.

Theorem 2. For each $t \leqslant T$

$$\mathbb{E}[W_t | \mathbf{X}[0, T]] = \exp\left\{ -\int_0^t \phi(X_s) \mathrm{d}s \right\}.$$

Proof. First note that, by direct calculation of its Doob–Meyer decomposition conditionally on $\mathbf{X}[0, T]$, $W_t \exp\{\int_0^t \phi(X_s)\mathrm{d}s\}$ is a martingale; see for example Revuz and Yor (2013). Therefore $\mathbb{E}[W_t | \mathbf{X}[0, T]] \exp\{\int_0^t \phi(X_s)\mathrm{d}s\} = 1$ and the result follows.

3.3. Simulating from the quasi-stationary distribution

In theory we can use our ability to simulate from $\mu_t(\mathbf{x})$, with using either rejection sampling to simulate from the quasi-stationary distribution of our killed Brownian motion, or importance

Table 1. Algorithm 1: importance sampling killed Brownian motion algorithm

1 Initialize: input initial value $\mathbf{X}_0$, and time interval length t; set $i=1$, $j=0$, $\tau_0=0$, $w_0^*=1$
2 R: choose hypercube $\mathcal{H}_i$ and calculate $L_{\mathbf{X}}^{(i)}$ and $U_{\mathbf{X}}^{(i)}$; simulate layer information $R_{\mathbf{X}}^{(i)} \sim \mathcal{R}$ as per Appendix C, obtaining $\tau_i, \mathbf{x}_{\tau_i}$
3 E: simulate $E \sim \text{Exp}(U_{\mathbf{X}}^{(i)} - L_{\mathbf{X}}^{(i)})$
4 ξ_j: set $j=j+1$ and $\xi_j = (\xi_{j-1} + E) \wedge \tau_i \wedge t$
5 $w_{\xi_j}^*$: set $w_{\xi_j}^* = w_{\xi_{j-1}}^* \exp\{-L_{\mathbf{X}}^{(i)}[\xi_j - \xi_{j-1}]\}$
6 $\mathbf{X}_{\xi_j}$: simulate $\mathbf{X}_{\xi_j} \sim \text{MVN}(\mathbf{X}_{\xi_{j-1}}, \xi_j - \xi_{j-1}) \vert R_{\mathbf{X}}^{(i)}$ as per Appendix C.2 and algorithm 5 there.
7 τ_i: if $\xi_j = t$ then output $\mathbf{x}_t$ and w_t^*; otherwise, if $\xi_j = \tau_i$, set $i=i+1$, and return to step 2; otherwise set $w_{\xi_j}^* = w_{\xi_j}^* \{U_{\mathbf{X}}^{(i)} - \phi(\mathbf{X}_{\xi_j})\}/(U_{\mathbf{X}}^{(i)} - L_{\mathbf{X}}^{(i)})$ and return to step 3

sampling to approximate this distribution. We would need to specify a 'burn-in' period of length t^* say, as in MCMC sampling and then to simulate from $\mu_{t^*}(\mathbf{x})$. If t^* is chosen appropriately these samples would essentially be draws from the quasi-stationary distribution. Furthermore we can propagate these samples forward in time to obtain samples from $\mu_t(\mathbf{x})$ for $t > t^*$, and these would, marginally, be draws from essentially the quasi-stationary distribution.

However, in practice this simple idea is unlikely to work. We can see this most clearly with the rejection sampler, as the probability of survival will decrease exponentially with t—and thus the rejection probability will often be prohibitively large.

Various approaches have been suggested to overcome the inefficiency of this naive approach to simulating from a quasi-stationary distribution (see for example de Oliveira and Dickman (2005), Groisman and Jonckheere (2013), and the recent rebirth methodology of Blanchet *et al.* (2016)). Our approach is to use ideas from SMC methods. In particular, we shall discretize time into m intervals of length T/m for some chosen T and m. Defining $t_i := iT/m$ for $i = 1, \ldots, m$, we use our importance sampler to obtain an N-sample approximation of $\mu_{t_1}(\mathbf{x})$; this will give us N particles, i.e. realizations of $\mathbf{x}_{t_1}$, and their associated importance sampling weights. We normalize the importance sampling weights and calculate the empirical variance of these normalized weights at time t_1. If this is sufficiently large we resample the particles, by simulating N times from the empirical distribution that is defined by the current set of weighted particles. If we resample, we assign each of the new particles a weight $1/N$.

The set of weighted particles at time t_1 is then propagated to obtain a set of N weighted particles at time t_2. The new importance sampling weights are just the weights at time t_1, before propagation, multiplied by the (incremental) importance sample weight calculated when propagating the particle from time t_1 to t_2. The above resampling procedure is applied, and this whole iteration is repeated until we have weighted particles at time T. This approach is presented as the QSMC algorithm in algorithm 2 in Table 2 in which N_{eff} is the effective sample size (ESS) of the weights (Kong *et al.*, 1994), a standard way of monitoring the variance of the importance sampling weights within SMC sampling, and N_{th} is a user-chosen threshold which determines whether or not to resample. The algorithm outputs the weighted particles at the end of each iteration.

Given the output from algorithm 2, the target distribution π can be estimated as follows. After choosing a burn-in time, t^* ($\in (t_0, \ldots, t_m)$), sufficiently large to provide reasonable confidence that quasi-stationarity has been 'reached', the approximation to the law of the killed process is then simply the weighted occupation measures of the particle trajectories in the interval $[t^*, T]$.

Table 2. Algorithm 2: QSMC algorithm

1 *Initialization step* ($i=0$):

(a) input, starting distribution $f_{\mathbf{x}_0}$, number of particles, N, and set of m times $t_{1:m}$

(b) $\mathbf{X}_0^{(\cdot)}$: for k $1,\ldots,N$ simulate $\mathbf{X}_{t_0}^{(1:N)} \sim f_{\mathbf{x}_0}$ and set $w_{t_0}^{(1:N)} = 1/N$

2 *Iterative update steps* ($i=i+1$ *while* $i \leqslant m$):

(a) N_{eff}, if $N_{\text{eff}} \leqslant N_{\text{th}}$ then for k $1,\ldots,N$ resample $\mathbf{X}_{t_{i-1}}^{(k)} \sim \tilde{\pi}_{t_{i-1}}^N$, the empirical distribution defined by the current set of weighted particles, and set $w_{t_{i-1}}^{(k)} = 1/N$;

(b) for k $1,\ldots,N$,

(i) $\mathbf{X}_{t_i}^{(\cdot)}$, simulate $\mathbf{X}_{t_i}^{(k)} | \mathbf{X}_{t_{i-1}}^{(k)}$ along with unnormalized weight increment $w_{t_i - t_{i-1}}^*$ as per algorithm 1;

(ii) $w'^{(\cdot)}_{t_i}$, calculate unnormalized weights $w'^{(k)}_{t_i} = w_{t_{i-1}}^{(k)} \; w_{t_i - t_{i-1}}^*$;

(c) $w_{t_i}^{(\cdot)}$: for k $1,\ldots,N$ set $w_{t_i}^{(k)} = w'^{(k)}_{t_i} / \Sigma_{l=1}^N w'^{(l)}_{t_i}$;

(d) $\tilde{\pi}_{t_i}^N$: set $\tilde{\pi}_{t_i}^N(\mathrm{d}\mathbf{x}) := \Sigma_{k=1}^N w_{t_i}^{(k)} \; \delta_{\mathbf{X}_{t_i}^{(k)}}(\mathrm{d}\mathbf{x})$

More precisely, using the output of the QSMC algorithm,

$$\pi(\mathrm{d}\mathbf{x}) \approx \hat{\pi}(\mathrm{d}\mathbf{x}) := \frac{1}{m(T-t^*)/T} \sum_{i=m(T-t^*)/T}^{m} \sum_{k=1}^{N} w_{t_i}^{(k)} \delta_{\mathbf{X}_{t_i}^{(k)}}(\mathrm{d}\mathbf{x}). \tag{7}$$

For concreteness, for a suitable $L^1(\pi)$ function g, the Monte Carlo estimator can simply be set to

$$\widehat{\pi(g)} = \frac{1}{m(T-t*)/T} \sum_{i=m(T-t^*)/T}^{m} \sum_{k=1}^{N} w_{t_i}^{(k)} g(\mathbf{X}_{t_i}^{(k)}). \tag{8}$$

The general (g-specific) theoretical ESS is given by $\mathrm{var}_\pi g / \mathrm{var}\widehat{\pi(g)}$. Practical approximation of ESS is discussed in Appendix I.

4. Subsampling

We now return to the problem of sampling from the posterior in a big data setting and assume that we can write the target posterior as

$$\pi(\mathbf{x}) \; (= \pi_n(\mathbf{x})) \propto \prod_{i=0}^{n} f_i(\mathbf{x}), \tag{9}$$

where $f_0(\mathbf{x})$ is the prior and $f_1(\mathbf{x}),\ldots,f_n(\mathbf{x})$ are likelihood terms. To be consistent with our earlier notation $\mathbf{x}$ refers to the parameters in our model. The assumption of this factorization is quite weak and includes many classes of models exhibiting various types of conditional independence structure.

It is possible to sample from this posterior by using algorithm 2 by choosing $\phi(\mathbf{x})$, and hence $\kappa(\mathbf{x})$, which determines the death rate of the killed Brownian motion, as defined in expressions (3) and (4) respectively. In practice this will be computationally prohibitive as at every potential death event we determine acceptance by evaluating $\phi(\mathbf{x})$, which involves calculating derivatives of the log-posterior, and so requires accessing the full data set of size n. However, it is easy to

estimate $\phi(\mathbf{x})$ unbiasedly by using subsamples of the data as the log-posterior is a sum over the different data points. Here we show that we can use such an unbiased estimator of $\phi(\mathbf{x})$ while still simulating the underlying killed Brownian motion exactly.

4.1. Simulating killed Brownian motion with an unbiased estimate of the killing rate

To introduce the approach proposed we begin by assuming that we can simulate an auxiliary random variable $A \sim \mathcal{A}$, and (without loss of generality) construct a positive unbiased estimator $\tilde{\kappa}_A(\cdot)$ such that

$$\mathbb{E}_{\mathcal{A}}[\tilde{\kappa}_A(\cdot)] = \kappa(\cdot). \tag{10}$$

The approach relies on the following simple result which is stated in a general way as it is of independent interest for simulating from events of probability which are expensive to compute but that admit a straightforward unbiased estimator. Its proof is trivial and will be omitted.

Proposition 1. Let $0 \leqslant p \leqslant 1$, and suppose that P is a random variable with $\mathbb{E}[P] = p$ and $0 \leqslant P \leqslant 1$ almost surely. Then, if $u \sim U[0, 1]$, the event $\{u \leqslant P\}$ has probability p.

We now adapt this result to our setting, noting that the randomness that is obtained by direct simulation of a p-coin, and that using proposition 1, is indistinguishable.

Recall that, in Section 3.1 to simulate a Poisson process of rate κ, Poisson thinning was used. The initial step is first to find, for the Brownian motion trajectory constrained to the hypercube $\mathcal{H}$, a constant $K_{\mathbf{X}} \in \mathbb{R}_+$ such that $\forall\, \mathbf{x} \in \mathcal{H}$, $\kappa(\mathbf{x}) \leqslant K_{\mathbf{X}}$ holds. Then a dominating Poisson process of rate $K_{\mathbf{X}}$ is simulated to obtain potential death events, and then in sequence each potential death event is accepted or rejected. A single such event, occurring at time ξ, will be accepted as a death with probability $\kappa(\mathbf{x}_\xi)/K_{\mathbf{X}}$.

An equivalent formulation would simulate a Poisson process of rate κ by using a dominating Poisson process of higher rate $\tilde{K}_{\mathbf{X}} \geqslant K_{\mathbf{X}}$. This is achieved by simply substituting $K_{\mathbf{X}}$ for $\tilde{K}_{\mathbf{X}}$ in the argument above. However, the penalty for doing this is an increase in the expected computational cost by a factor of $\tilde{K}_{\mathbf{X}}/K_{\mathbf{X}}$—therefore it is reasonable to expect to have a larger number of potential death events, each of which will have a smaller acceptance probability.

Now, suppose for our unbiased estimator $\tilde{\kappa}_A$ that it is possible to identify some $\tilde{K}_{\mathbf{X}} \in \mathbb{R}_+$ such that for $\mathcal{A}$ almost all A, and all $\mathbf{x} \in \mathcal{H}$, $0 \leqslant \tilde{\kappa}(\mathbf{x}) \leqslant \tilde{K}_{\mathbf{X}}$. Noting from equation (10) that we have an unbiased $[0, 1]$-valued estimator of the probability of a death event in the above argument (i.e. $\mathbb{E}_{\mathcal{A}}[\tilde{\kappa}_A(\mathbf{x})/\tilde{K}] = \kappa(\mathbf{x})/\tilde{K}$) and, by appealing to proposition 1, another (entirely equivalent) formulation of the Poisson thinning argument above is to use a dominating Poisson process of rate $\tilde{K}_{\mathbf{X}}$, and to determine acceptance or rejection of each potential death event by simulating $A \sim \mathcal{A}$ and accepting with probability $\tilde{\kappa}_A(\mathbf{x}_\xi)/\tilde{K}$ (instead of $\kappa(\mathbf{x}_\xi)/\tilde{K}$).

In the remainder of this section we exploit this extended construction of Poisson thinning (using an auxiliary random variable and unbiased estimator), to develop a scalable alternative to the QSMC approach that was introduced in algorithm 2. The key idea in doing so is to find an auxiliary random variable and unbiased estimator which can be simulated and evaluated without fully accessing the data set, while ensuring that the increased number of evaluations that is necessitated by the ratio $\tilde{K}_{\mathbf{X}}/K_{\mathbf{X}} \geqslant 1$ does not grow too severely.

4.2. Constructing a scalable replacement estimator

Noting from expressions (3) and (4) that the selection of $\kappa(\mathbf{x})$ that is required to sample from a posterior $\pi(\mathbf{x})$ is determined by $\phi(\mathbf{x})$, in this section we focus on finding a practical construction of a scalable unbiased estimator for $\phi(\mathbf{x})$. Recall that

$$\phi(\mathbf{x}) := [\|\nabla \log\{\pi(\mathbf{x})\}\|^2 + \Delta \log\{\pi(\mathbf{x})\}]/2, \tag{11}$$

and that as in algorithm 2, while staying within hypercube $\mathcal{H}_i$, it is possible to find constants $L_{\mathbf{X}}^{(i)}$ and $U_{\mathbf{X}}^{(i)}$ such that $L_{\mathbf{X}}^{(i)} \leqslant \phi(\mathbf{x}) \leqslant U_{\mathbf{X}}^{(i)}$. As motivated by Section 4.1, it is then possible to construct an auxiliary random variable $A \sim \mathcal{A}$, and an unbiased estimator ϕ_A such that

$$\mathbb{E}_{\mathcal{A}}[\phi_A(\cdot)] = \phi(\cdot), \tag{12}$$

and to determine constants $\tilde{U}_{\mathbf{X}}^{(i)} \geqslant U_{\mathbf{X}}^{(i)}$ and $\tilde{L}_{\mathbf{X}}^{(i)} \leqslant L_{\mathbf{X}}^{(i)}$ such that within the same hypercube we have $\tilde{L}_{\mathbf{X}}^{(i)} \leqslant \tilde{\phi}_A(\mathbf{x}) \leqslant \tilde{U}_{\mathbf{X}}^{(i)}$. To ensure the validity of our QSMC approach, as justified by theorem 1 in Section 3.3, it is necessary to substitute condition 1 with the following (similarly weak) condition.

Condition 2. There is a constant $\tilde{\Phi} > -\infty$ such that $\tilde{\Phi} \leqslant \tilde{\phi}_A(\mathbf{u})$ for $\mathcal{A}$-almost every A, $\forall\, \mathbf{u} \in \mathbb{R}^d$.

To ensure practicality and scalability it is crucial to focus on ensuring that the ratio

$$\frac{\tilde{\lambda}}{\lambda} = \frac{\tilde{U}_{\mathbf{X}}^{(i)} - \tilde{L}_{\mathbf{X}}^{(i)}}{U_{\mathbf{X}}^{(i)} - L_{\mathbf{X}}^{(i)}}, \tag{13}$$

where $\tilde{\lambda} := \tilde{U}_{\mathbf{X}}^{(i)} - \tilde{L}_{\mathbf{X}}^{(i)}$, does not grow too severely with the size of the data set (as this determines the multiplicative increase in the rate, and hence increased inefficiency, of the dominating Poisson process required within algorithm 2). To do this, our approach develops a tailored control variate, of a similar type to that which has since been successfully used within the concurrent work of two of the authors on MCMC methods (see Bierkens *et al.* (2019)).

To implement the control variate estimator, it is first necessary to find a point that is close to a mode of the posterior distribution π, denoted by $\hat{\mathbf{x}}$. In fact, for the scaling arguments to hold, $\hat{\mathbf{x}}$ should be within $\mathcal{O}(n^{-1/2})$ of the true mode, and achieving this is a less demanding task than actually locating the mode. Moreover we note that this operation is required to be done only once, and not at each iteration, and so can be done fully in parallel. In practice it would be possible to use a stochastic gradient optimization algorithm to find a value that is close to the posterior mode, and we recommend then starting the simulation of our killed Brownian motion from this value, or from some suitably chosen distribution centred at this value. Doing this substantially reduces the burn-in time of our algorithm. In the following section we describe a simpler method that is applicable when two passes of the full data set can be tolerated in the algorithm's initialization.

Addressing scalability for multimodal posteriors is a more challenging problem, and goes beyond what is addressed in this paper, but is of significant interest for future work. We do, however, make the following remarks. In the presence of multimodality, stochastic gradient optimization schemes may converge to the wrong mode. This is still sufficiently good as long as possible modes are separated by a distance which is $\mathcal{O}(n^{-1/2})$; when separate modes which are separated by more than $\mathcal{O}(n^{-1/2})$ exist, an interesting option would be to adopt multiple control variates.

Remembering that $\log\{\pi(\mathbf{x})\} = \Sigma_{i=0}^{n} \log\{f_i(\mathbf{x})\}$ and letting $\mathcal{A}$ be the law of $I \sim U\{0, \ldots, n\}$, our control variate estimator is constructed thus:

$$\mathbb{E}_{\mathcal{A}}\big[\underbrace{(n+1)[\nabla \log\{f_I(\mathbf{x})\} - \nabla \log\{f_I(\hat{\mathbf{x}})\}]}_{=:\tilde{\alpha}_I(\mathbf{x})}\big] = \underbrace{\nabla \log\{\pi(\mathbf{x})\} - \nabla \log\{\pi(\hat{\mathbf{x}})\}}_{=:\alpha(\mathbf{x})}. \tag{14}$$

As such, $\phi(\mathbf{x})$ can be re-expressed as

Table 3. Algorithm 3: ScaLE†

Initialize: choose $\hat{\mathbf{x}}$ and compute $\nabla\log\{\pi(\hat{\mathbf{x}})\}$ and $\Delta\log\{\pi(\hat{\mathbf{x}})\}$ 2(b)(i) On calling algorithm 1, (a) replace $L_{\mathbf{X}}^{(i)}$ and $U_{\mathbf{X}}^{(i)}$ in step 2 with $\tilde{L}_{\mathbf{X}}^{(i)}$ and $\tilde{U}_{\mathbf{X}}^{(i)}$; (b) replace step 7 with τ_i, if $\xi_j=\tau_i$, set $i=i+1$, and return to step 2; otherwise simulate $A_j=(I_j,J_j)$, with $I_j,J_j\sim^{\mathrm{IID}}U\{0,\ldots,n\}$, and set $w^*_{\xi_j}=w^*_{\xi_j}\{\tilde{U}_{\mathbf{X}}^{(i)}-\tilde{\phi}_{A_j}(\mathbf{X}_{\xi_j})\}/(\tilde{U}_{\mathbf{X}}^{(i)}-\tilde{L}_{\mathbf{X}}^{(i)})$ (where $\tilde{\phi}_{A_j}$ is defined as in equation (16)) and return to algorithm 1, step 3

†As per algorithm 2 unless stated otherwise.

$$\phi(\mathbf{x})=(\alpha(\mathbf{x})^{\mathrm{T}}[2\nabla\log\{\pi(\hat{\mathbf{x}})\}+\alpha(\mathbf{x})]+\mathrm{div}\{\alpha(\mathbf{x})\}/2+C, \tag{15}$$

where $C:=\|\nabla\log\{\pi(\hat{\mathbf{x}})\}\|^2/2+\Delta\log\{\pi(\hat{\mathbf{x}})\}/2$ is a constant. Letting $\mathcal{A}$ now be the law of $I,J\sim^{\mathrm{IID}}U\{0,\ldots,n\}$ the following unbiased estimator of ϕ can be constructed:

$$\mathbb{E}_{\mathcal{A}}\Big[\underbrace{(\tilde{\alpha}_I(\mathbf{x})^T[2\nabla\log\{\pi(\mathbf{x})\}+\tilde{\alpha}_I(\mathbf{x})]+\mathrm{div}\{\tilde{\alpha}_I(\mathbf{x})\})/2+C}_{=:\tilde{\phi}_A(\mathbf{x})}\Big]=\phi(\mathbf{x}). \tag{16}$$

The estimators $\tilde{\alpha}_I(\mathbf{x})$ and $\tilde{\phi}_A(\mathbf{x})$ are nothing more than classical *control variate* estimators, albeit in a fairly elaborate setting, and henceforth we shall refer to these accordingly.

The construction of the estimator requires evaluation of the constants $\nabla\log\{\pi(\hat{\mathbf{x}})\}$ and $\Delta\log\{\pi(\hat{\mathbf{x}})\}$. Although both are $\mathcal{O}(n)$ evaluations they must be computed only once and, furthermore, as mentioned above, can be calculated entirely in parallel.

The unbiased estimators $\tilde{\alpha}_I(\mathbf{x})$ and $\tilde{\phi}_A(\mathbf{x})$ use (respectively) single and double draws from $\{1,\ldots,n\}$ although it is possible to replace these by averaging over multiple draws (sampled with replacement), although this is not studied theoretically in the present paper and is exploited only in Section 7.5 of the empirical study.

Embedding our subsampling estimator described above within the QSMC algorithm of Section 3.3 results in algorithm 3 in Table 3, termed the *scalable Langevin exact algorithm ScaLE*. A similar modification could be made to the rejection sampling version; called R-QSMC, which was discussed in Section 3.3 and is detailed in Appendix F. This variant is termed the *rejection scalable Langevin exact algorithm R-ScaLE*, and full algorithmic details are provided in Appendix G.

4.3. Implementation details

In this section we detail some simple choices of the various algorithmic parameters which lead to a concrete implementation of ScaLE. These choices have been made on the bases of parsimony and convenience and are certainly not optimal.

In practice, we are likely to want to employ a suitable preconditioning transformation $\mathbf{X}'=\mathbf{\Lambda}^{-1}\mathbf{X}$ before applying the algorithm to equate roughly scales for different components. If we did not do this, it is likely that some components would mix particularly slowly. Obtaining a suitable $\hat{\mathbf{x}}$ and $\mathbf{\Lambda}$ is important. One concrete approach, and that used throughout our empirical study except where otherwise stated, is as follows. Divide a data set into a number of batches which are sufficiently small to be processed by using standard maximum likelihood estimation approaches and estimate the maximum likelihood estimate (MLE) and observed Fisher information for each batch; $\hat{\mathbf{x}}$ can then be chosen to be the mean of these MLEs and $\mathbf{\Lambda}^{-1}$ to be a diagonal matrix with elements equal to the square root of the sum of the diagonal elements of the estimated

information matrices. Better performance would generally be obtained by using a non-diagonal matrix, but this serves to illustrate a degree of robustness to the specification of these parameters. The constants that are required within the control variate can then be evaluated. For a given hypercube $\mathcal{H}$, a bound $\tilde{K}_{\mathbf{X}}$ on the dominating Poisson process intensity can then be obtained by simple analytic arguments facilitated by extending that hypercube to include $\hat{\mathbf{x}}$ and obtaining bounds on the modulus of continuity of $\tilde{\phi}_A$. In total, two passes of the full data set are required to obtain the necessary algorithmic parameters and to specify the control variate fully.

As discussed in Section 3.3, it is necessary to choose an execution time T for the algorithm and an auxiliary mesh $(t_0 := 0, t_1, \ldots, t_m := T)$ on which to evaluate g in the computation of the QSMC estimator (8). Within the algorithm the particle set is evolving according to killed Brownian motion with a preconditioning matrix $\mathbf{\Lambda}^{-1}$ chosen to match approximately the square root of the information matrix of the target posterior. As such, T should be chosen to match the time that is taken for preconditioned Brownian motion to explore such a space, which in the examples that are considered in this paper ranged from $T \approx 1$ to $T \approx 100$. The number of temporal mesh points, m, was chosen with computational considerations in mind—increasing m increases the cost of evaluating the estimator and leads to greater correlation between the particle set at consecutive mesh points but ensures when running the algorithm on a multiple-user cluster that the simulation is periodically saved and reduces the variance of the estimator. As the computational cost of the algorithm is entirely determined by the bounds on the discussed modulus of continuity of $\tilde{\phi}_A$, in each of the examples we later consider that our mesh size was loosely determined by the inverse of this quantity and ranged from $t_i - t_{i-1} \approx 10^{-3}$ to $t_i - t_{i-1} \approx 10^{-6}$.

The initial distribution $f_{\mathbf{x}_0}$ is not *too* critical, provided that it is concentrated reasonably close (within a neighbourhood of size $\mathcal{O}(n^{-1/2})$) to the mode of the distribution. The stability properties of the SMC implementation ensure that the initial conditions will be forgotten (see chapter 7 of Del Moral (2004) for a detailed discussion). The empirical results that are presented below were obtained by choosing, as $f_{\mathbf{x}}$, either a singular distribution concentrated at $\hat{\mathbf{x}}$ or a normal distribution centred at that location with a covariance matrix matching $\mathbf{\Lambda}\mathbf{\Lambda}^{\mathrm{T}}$; results were found to be insensitive to the particular choice.

5. Complexity of ScaLE

The computational cost of ScaLE will be determined by two factors: the speed at which μ_t approaches π and the computational cost of running the algorithm per unit algorithm time. Throughout the exposition of this paper, the proposal process is simple Brownian motion. Because of posterior contraction, as n grows, this proposal Brownian motion moves increasingly rapidly through the support of π. However, as n grows, killing rates will grow. In this subsection we shall explore in detail how the computational cost of ScaLE varies with n (its complexity) while bringing out explicitly the delicate link to the rate of posterior contraction and the effect of the choice of $\hat{\mathbf{x}}$.

We start by examining the speed of convergence of μ_t and in particular its dependence on posterior contraction. Being more explicit about posterior contraction, we say that $\{\pi_n\}$ are $\mathcal{O}(n^{-\eta/2})$ or have *contraction rate* $\eta/2$ for some $\eta > 0$ to a limit $\mathbf{x}_0$ if for all $\epsilon > 0$ there exists $K > 0$ such that, when $\mathbf{X}_n \sim \pi_n$, $\mathbb{P}(|\mathbf{X}_n - \mathbf{x}_0| > Kn^{-\eta/2}) < \epsilon$. It is necessary to extend the definition of μ_t to a setting where n increases; hence define

$$\mu_t^{n,\mathbf{u}}(\mathrm{d}\mathbf{x}) := \mathbb{P}(\mathbf{X}_t \in \mathrm{d}\mathbf{x} | \zeta > t, \mathbf{X}_0 = \mathbf{x}_0 + n^{-\eta/2}\mathbf{u}). \tag{17}$$

Since we are dealing with Markov processes that are essentially never uniformly ergodic, it is impossible to control convergence times uniformly. The specification of the initial value as

$\mathbf{X}_0 = \mathbf{x}_0 + n^{-\eta/2}\mathbf{u}$, which, as n increases, remains close to the centre of the posterior as specified through the contraction rate, goes as far as we can before incurring additional computational costs for bad starting values.

Set

$$T_{n,\mathbf{u},\epsilon} = \inf\{t \geqslant 0;\ \|\mu_t^{n,\mathbf{u}} - \pi_n\| < \epsilon\}$$

where '$\|\cdot\|$' represents total variation distance. It will be necessary to make the following technical assumption. For all $\epsilon, K > 0$

$$\limsup_{n\to\infty} \sup_{|\mathbf{u}|<K} n^{\eta} T_{n,\mathbf{u},\epsilon} < \infty. \tag{18}$$

At first sight, assumption (18) may seem strong, but it is very natural and is satisfied in reasonable situations. For example suppose that we have a contraction scaling limit:

$$\pi_n(dx) \approx h\left(\frac{\mathbf{x} - \mathbf{x}_0}{n^{\eta/2}}\right).$$

(A special case of this is the Bernstein–von Mises theorem with $\eta = 1$ and h being Gaussian, but our set-up is far broader.) If $\{\mathbf{X}_t^n\}$ denotes ScaLE on π_n, then by simple scaling and time change properties of Brownian motion it is easily checked that if $\mathbf{Y}_t = \mathbf{X}_{n^{-\eta}t}$ then $\mathbf{Y}$ is (approximately) ScaLE on h which is clearly independent of n. Thus to obtain a process which converges in $\mathcal{O}(1)$ we need to *slow down* $\mathbf{X}$ by a time scaling factor of

$$\text{time factor} = n^{\eta}. \tag{19}$$

Similar arguments have been used for scaling arguments of other Monte Carlo algorithms that use similar control variates; see for instance the concurrent work of Bierkens *et al.* (2019).

Although posterior contraction has a positive effect on computational cost, also, for large n, the rate at which a likelihood subsample needs to be calculated, as measured by $\tilde{\lambda}$, needs to increase. Since $\tilde{\lambda}$ depends on the current location in the state space, where we need to be precise we shall set $\tilde{\lambda}_{n,K}$ to be an available bound which applies uniformly for $|\mathbf{x} - \mathbf{x}_0| < Kn^{-\eta/2}$.

The following notion of *convergence cost* will be required: setting

$$\mathcal{C}_{\text{iter}} = \mathcal{C}_{\text{iter}}(n, K, \epsilon) = T_{n,K,\epsilon}\tilde{\lambda}_{n,K}$$

ScaLE is said to have *iteration complexity* n^{ϖ} or, equivalently, is $\mathcal{O}(n^{\varpi})$ if $\mathcal{C}_{\text{iter}}(n, K, \epsilon)$ is $\mathcal{O}(n^{\varpi})$ for all $K, \epsilon > 0$.

Therefore to understand iteration complexity of ScaLE it is necessary to understand the rate at which $\tilde{\lambda}_{n,K}$ grows with n. A general way to do this is to use global, or local, bounds on the second derivatives of the log-likelihood for each datum. To simplify the following exposition a global bound is assumed, so that

$$\rho[\nabla^2 \log\{f_I(\mathbf{x})\}] \leqslant P_n, \tag{20}$$

for some $P_n > 0$, where $\rho(\cdot)$ represents the spectral radius and ∇^2 is the Hessian matrix. For smooth densities with Gaussian and heavier tails, the Hessian of the log-likelihood is typically uniformly bounded (in both data and parameter). In such cases such a global bound would be expected, and in fact P_n would be constant in n.

Recalling the layer construction of Section 3.2 for a single trajectory of killed Brownian motion, we can ensure that over any finite time interval we have $\mathbf{x} \in \mathcal{H}$, some hypercube. Let the centre of the hypercube be $\mathbf{x}^*$.

In this section, eventually the assumption that the posterior contracts at a rate $n^{-\eta/2}$ will be made, i.e. that $\{n^{\eta/2}(\mathbf{x}-\mathbf{x}_0), n=1,2,\ldots\}$ is tight. The so-called *regular case* corresponds to the case where $\eta=1$, although there is no need to make any explicit assumptions about normality in what follows. The practitioner has complete freedom to choose $\mathcal{H}$, and it makes sense to choose this so that $\|\mathbf{x}-\mathbf{x}^*\| < C^* n^{-\eta/2}$ for some $C^*>0$ and for all $\mathbf{x}\in\mathcal{H}$.

It is possible to bound $\tilde{\phi}_A(\mathbf{x})$ both above and below if we can bound $|\tilde{\phi}_A(\mathbf{x})|$ over $\mathcal{A}$ almost all possible realizations of A. To bound $|\tilde{\phi}_A(\mathbf{x})|$, the approach here is first to consider the elementary estimator in expression (14). By imposing condition (20) we can then obtain

$$\max_{\mathbf{x}\in\mathcal{H}, I\in\{0,\ldots,n\}} |\tilde{\alpha}_I(\mathbf{x})| \leqslant (n+1) P_n \max_{\mathbf{x}\in\mathcal{H}} \|\mathbf{x}-\hat{\mathbf{x}}\|. \tag{21}$$

Thus it is possible to bound estimator (16) as follows:

$$2 \max_{\mathbf{x}\in\mathcal{H}, A\in\mathcal{A}} |\tilde{\phi}_A(\mathbf{x}) - C| \leqslant (n+1) P_n \max_{\mathbf{x}\in\mathcal{H}} \|\mathbf{x}-\hat{\mathbf{x}}\| [|2\nabla \log\{\pi(\hat{\mathbf{x}})\}| + P_n (n+1) \max_{\mathbf{x}\in\mathcal{H}} \|\mathbf{x}-\hat{\mathbf{x}}\|] + P_n d(n+1). \tag{22}$$

We can use the fact that $\max_{\mathbf{x}\in\mathcal{H}} \|\mathbf{x}-\hat{\mathbf{x}}\| \leqslant \|\mathbf{x}^*-\hat{\mathbf{x}}\| + C^* n^{-\eta/2}$ to bound the terms in this expression.

We now directly consider the iteration complexity of ScaLE. We note that the appropriate killing rate to ensure mixing in time $\mathcal{O}(1)$ involves slowing down by the time factor given in expression (19) and is therefore just $n^{-\eta}\tilde{\lambda}$. Assuming that $\eta \leqslant 1$, and using the bound on $|\tilde{\phi}_A(\mathbf{x}) - C|$ for the hypercube centred on $\mathbf{x}^*$, we have that, while we remain within the hypercube,

$$\frac{1}{n^\eta}\tilde{\lambda} = \mathcal{O}(P_n n^{1-3\eta/2}[P_n n^{1-\eta/2} + |\nabla \log\{\pi(\hat{\mathbf{x}})\}|]). \tag{23}$$

Here the assumption has been made that at stationarity $\mathbf{x}^*$ will be a draw from the support of the posterior, so that, under the assumption of posterior contraction at the $n^{-\eta/2}$-rate, then $\|\mathbf{x}^*-\hat{\mathbf{x}}\| = \mathcal{O}_p(n^{-\eta/2})$. This discussion is summarized in the following result.

Theorem 3. Suppose that assumptions (18) and (20) hold, posterior contraction occurs at rate $n^{-\eta/2}$ for $\eta \leqslant 1$, P_n is $\mathcal{O}(1)$ and $|\nabla \log\{\pi(\hat{\mathbf{x}})\}| = \mathcal{O}(n^\iota)$ for some $\iota > 0$. Then the iterative complexity of ScaLE is $\mathcal{O}(n^\varpi)$ where

$$\varpi := \max(1-\eta/2, \iota) + 1 - 3\eta/2.$$

In particular, where $\iota \leqslant 1-\eta/2$, $\varpi = 2-2\eta$. If $\eta=1$, then it follows that $\varpi = 0$ and the iterative complexity of ScaLE is $\mathcal{O}(1)$.

This result also illuminates the role that is played by $|\nabla \log\{\pi(\hat{\mathbf{x}})\}|$ in the efficiency of the algorithm. In the following discussion it is assumed that $\eta=1$. It is worth noting that although a completely arbitrary starting value for $\hat{\mathbf{x}}$ might make $|\nabla \log\{\pi(\hat{\mathbf{x}})\}|$ an $\mathcal{O}(n)$ quantity leading to an iterative complexity of the algorithm which is $\mathcal{O}(n^{1/2})$, to obtain $\mathcal{O}(1)$ it is simply required that $|\nabla \log\{\pi(\hat{\mathbf{x}})\}|$ be $\mathcal{O}(n^{1/2})$ which gives considerable leeway for any initial explorative algorithm to find a good value for $\hat{\mathbf{x}}$.

Note that, given bounds on the third derivatives, equation (23) can be improved by linearizing the divergence term in expression (16). This idea is exploited later in a logistic regression example (see Sections 7.2, 7.3 and 7.4).

In the absence of a global bound on the second derivatives, it is possible to replace P_n in the above arguments by any constant that bounds the second derivatives for all $\mathbf{x}$ such that $\|\mathbf{x}-\hat{\mathbf{x}}\| \leqslant \max_{\mathbf{x}\in\mathcal{H}} \|\mathbf{x}-\hat{\mathbf{x}}\|$. In this case, the *most extreme* rate at which $\tilde{\lambda}$ can grow is logarithmically with n,

for instance for light-tailed models where the data really come from the model being used. Where the tails are misspecified and light-tailed models are being used, the algorithmic complexity can be considerably worse. There is considerable scope for more detailed analyses of these issues in future work.

The above arguments give insight into the influence of our choice of $\hat{\mathbf{x}}$. It affects the bound on $\tilde{\lambda}$, and hence the computational efficiency of ScaLE, through the terms $\|\mathbf{x}^* - \hat{\mathbf{x}}\|$. Furthermore the main term in the order of $\tilde{\lambda}$ is the square of this distance. If $\hat{\mathbf{x}}$ is the posterior mean, then the square of this distance will, on average, be the posterior variance. By comparison, if $\hat{\mathbf{x}}$ is k posterior standard deviations away from the posterior mean, then on average the square distance will be $k^2 + a$ times the posterior variance (for some constant a), and the computational cost of ScaLE will be increased by a factor of roughly $k^2 + a$.

5.1. Overall complexity

Here we shall briefly discuss the overall complexity of ScaLE. The general set-up of theorem 3 describes the iteration complexity of ScaLE on the assumption that $|\nabla \log\{\pi(\hat{\mathbf{x}})\}|$ grows no worse than $\mathcal{O}(n^{\iota})$. However, there is a substantial initial computational cost in locating $\hat{\mathbf{x}}$ and calculating $\nabla \log\{\pi(\hat{\mathbf{x}})\}$ which is likely to be $\mathcal{O}(n)$ as there are n terms in the calculation of the latter. Therefore the *overall complexity* of ScaLE can be described as

$$\mathcal{C} = \mathcal{C}_{\text{init}} + \mathcal{C}_{\text{iter}} = \mathcal{O}(n) + \mathcal{O}(n^{\varpi} t)$$

where t represents algorithm time. This is in contrast with an MCMC algorithm for which the iteration cost would be $\mathcal{O}(n)$, leading to overall complexity tn. A Laplace approximation will involve an initial cost that is (at very least) $\mathcal{O}(n)$ but no further computation.

Since they both involve full likelihood calculations, finding the posterior mode and finding $\hat{x}$ are both likely to be $\mathcal{O}(n)$ calculations. This can be shown to be so for strongly log-concave posterior densities (Nesterov, 2013), though the cost may be higher if the log-posterior is not concave. However, the above discussion shows that to achieve $\mathcal{O}(1)$ scaling with data we typically only need to find $\hat{x}$ within $\mathcal{O}(n^{-1/2})$ of the posterior model. Thus finding $\hat{x}$ is certainly no more difficult than finding the posterior mode, as we can use the same mode finding algorithm, e.g. Bottou (2010), Nesterov (2013) and Jin *et al.* (2018), but have the option of stopping earlier.

If n is sufficiently large that the cost of initialization dominates the iteration cost, ScaLE will computationally be no more expensive to implement than the Laplace approximation. In this case we obtain an *exact approximate* algorithm (ScaLE) for at most the computational complexity of an approximate method (Laplace). These complexity considerations are summarized in Table 4.

Table 4. Complexity of algorithms for big data†

Algorithm	$\mathcal{C}_{\text{init}}$	$\mathcal{C}_{\text{iter}}$	$\mathcal{C}$
MCMC	0	tn	tn
Laplace approximation	n	0	n
ScaLE	n	tn^{ϖ}	$n + tn^{\varpi}$
ScaLE when $\eta = 1$	n	t	$n + t$

†This is split into the complexity of initiation, $\mathcal{C}_{\text{init}}$, and the cost of the iterative algorithm, $\mathcal{C}_{\text{iter}}$. Here n denotes sample size, t denotes algorithm time, and ϖ and η are as given in theorem 3.

6. Theoretical properties

SMC algorithms in both discrete and continuous time have been studied extensively in the literature (for related theory for approximating a fixed point distribution, including for algorithms with resampling implemented in continuous time, see Del Moral and Miclo (2000, 2003) and Rousset (2006); a discrete time algorithm to approximate a fixed point distribution in a different context was considered by Whiteley and Kantas (2017)). To avoid a lengthy technical diversion, we restrict ourselves here to studying a slightly simplified version of the problem to obtain the simplest and most interpretable possible form of results. The technical details of this construction are deferred to Appendix H and we give here only a qualitative description that is intended to guide intuition and the key result: that the resulting estimator satisfies a Gaussian central limit theorem with the usual Monte Carlo rate.

Consider a variant of the algorithm in which (multinomial) resampling occurs at times kh for $k \in \mathbb{N}$ where h is a time step resolution that is specified in advance and consider the behaviour of estimates that are obtained at these times. Extension to resampling at a random subset of these resampling times would be possible by using the approach of Del Moral *et al*. (2012), considering precisely the QSMC algorithm that was presented in algorithm 2 and ScaLE in algorithm 3 would require additional technical work that is somewhat beyond the scope of this paper; no substantial difference in behaviour was observed.

To employ standard results for SMC algorithms it is convenient to consider a discrete time embedding of the algorithms described. We consider an abstract formalism in which between the specified resampling times the trajectory of the Brownian motion is sampled, together with such auxiliary random variables as are required in any particular variant of the algorithm. Provided that the potential function that is employed to weight each particle before resampling has conditional expectation (given the path) proportional to the exact killing rate integrated over these discrete time intervals a valid version of ScaLE is recovered.

This discrete time formalism enables results on more standard SMC algorithms to be applied directly to ScaLE. We provide in the following proposition a straightforward corollary to a result in chapter 9 of Del Moral (2004), which demonstrates that estimates that are obtained from a single algorithmic time slice of the ScaLE satisfy a central limit theorem.

Proposition 2 (central limit theorem). In the context described, under mild regularity conditions (see references given in Appendix H),

$$\lim_{N\to\infty} \sqrt{N}\left(\frac{1}{N}\sum_{i=1}^{N} \varphi(X_{hk}^i) - \mathbb{E}_{\mathbb{K}_{hk}^x}[\varphi(X_{hk}^i)]\right) \Rightarrow \sigma_k(\varphi) Z$$

where $\varphi : \mathbb{R}^d \to \mathbb{R}$, Z is a standard normal random variable, '$\Rightarrow$' denotes convergence in distribution and $\sigma_k(\varphi)$ depends on the precise choice of subsampling scheme as well as the test function of interest and is specified in Appendix H along with the law $\mathbb{K}_{hk}^x$.

7. Examples

In this section we present five example applications of the methodology that is developed in this paper, each highlighting a different aspect of ScaLE and contrasted with appropriate competing algorithms. In particular, in Section 7.1 we consider a simple pedagogical example which has a skewed target distribution, contrasted with MCMC sampling; Section 7.2 considers the performance of a logistic regression model in which significantly less information is available from the data about one of the covariates than the others; in Section 7.3 we apply both ScaLE and the stochastic gradient Langevin diffusion algorithm SGLD to a regression problem based on

the American Statistical Association's data expo airline on-time performance data, which are of moderately large size (of the order of 10^8); Section 7.4 considers ScaLE applied to a very large logistic regression problem, with a data set of size $n = 2^{34} \approx 10^{10.2}$, along with consideration of scalability with respect to data size; finally, in Section 7.5, parameter inference for a contaminated regression example is given, motivated by a big data application with $n = 2^{27} \approx 10^{8.1}$, and illustrating the potential of an *approximate* implementation of ScaLE even when misinitialized.

All simulations were conducted in R on an Xeon X5660 central processor unit running at 2.8 GHz. For the purposes of presenting the ScaLE methodology as cleanly as possible, in each example no prior has been specified. In practice, a prior can be simply included within the methodology as described in Section 4.

7.1. Skewed target distribution

To illustrate ScaLE applied to a simple non-Gaussian target distribution, we constructed a small data set of size $n = 10$, to which we applied a logistic regression model

$$y_i = \begin{cases} 1 & \text{with probability } \exp(\mathbf{x}_i^{\mathrm{T}}\beta)/\{1+\exp(\mathbf{x}_i^{\mathrm{T}}\beta)\}, \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

The data were chosen to induce a skewed target, with $\mathbf{y}^{\mathrm{T}} = (1, 1, 0, \ldots, 0)$ and $\mathbf{x}_i^{\mathrm{T}} = (1, (-1)^i/i)$.

We used the `glm` R package to obtain the maximum likelihood estimate ($\beta^* \approx (-1.5598, -1.3971)$) and observed Fisher information, to (mis)initialize the particles in ScaLE. In total $N = 2^{10}$ particles were used, along with a subsampling mechanism of size 2 and a control variate computed as in Section 4.2 by setting $\hat{\mathbf{x}} = \beta^*$. For comparison we ran a random-walk Metropolis algorithm on the same example initialized at β^* by using the `MCMClogit` function provided by MCMCpack (Martin *et al.*, 2011), computed the posterior marginals based on 1 million iterations thinned to 100000 and after discarding a burn-in of 10000 iterations, and overlaid them together those estimated by ScaLE in Fig. 1. These are accompanied by the `glm` fit used to misinitialize ScaLE.

It is clear from Fig. 1 that the posterior that was obtained by simulating ScaLE matches that of MCMC sampling, and both identify the skew which would be overlooked by a simple normal approximation. The particle set in ScaLE quickly recovers from its misinitialization, and only a modest burn-in period is required. In practice, we would of course not advocate using ScaLE for such a small data setting—the computational and implementational complexity of ScaLE does not compete with MCMC sampling in this example. However, as indicated in Section 5 and the subsequent examples, ScaLE is robust to increasing data size whereas simple MCMC sampling will scale at best linearly.

7.2. Heterogeneous logistic regression

For this example a synthetic data set of size $n = 10^7$ was produced from the logistic regression model (24). Each record contained three covariates, in addition to an intercept. The covariates were simulated independently from a three-dimensional normal distribution with identity covariance truncated to $[-0.001, 0.001] \times [-1, 1] \times [-1, 1]$, and with the true $\beta = (0, 2, -2, 2)$ (where the first co-ordinate corresponds to the intercept). The specification of this data set is such that significantly less information is available from the data about the second covariate than about the others. Data were then generated from model (24) by using the simulated covariates.

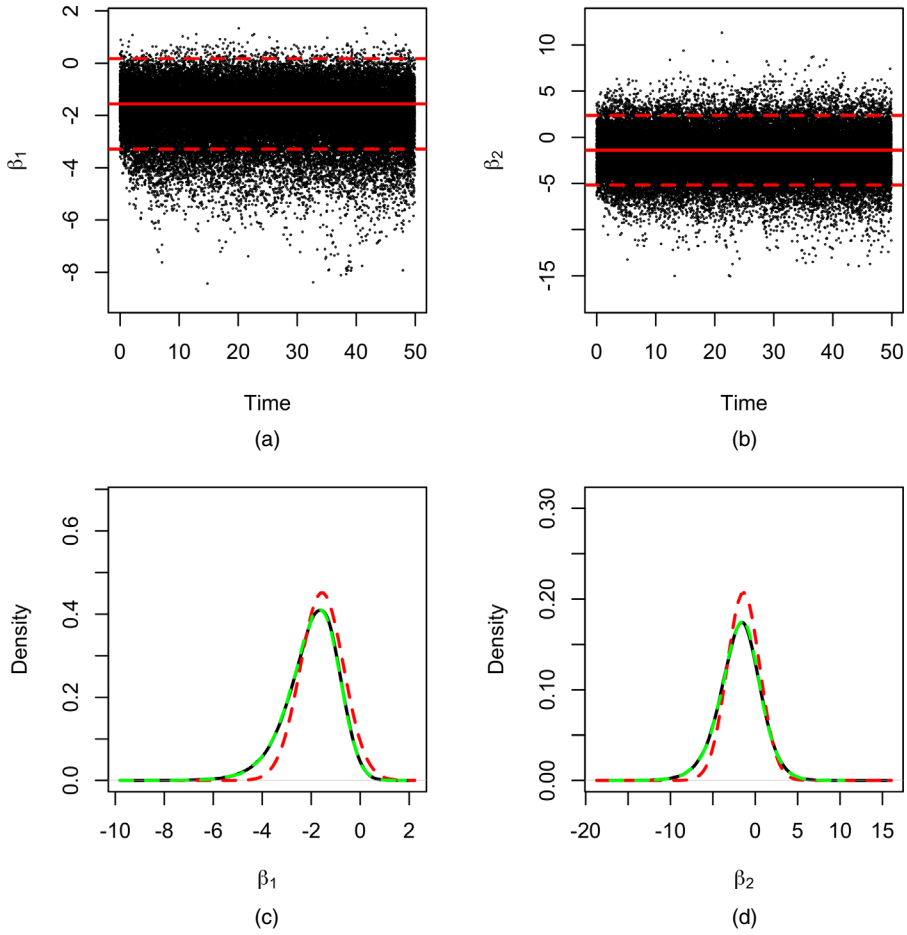

Fig. 1. (a), (b) Trace trajectories of ScaLE applied to the skewed target distribution example of Section 7.1 (———, parameter values fitted using the glm R package; — — —, 95% confidence intervals imputed by using the covariance matrix estimated from the glm package) and (c), (d) marginal densities obtained by ScaLE (— — —) (overlaid are the normal approximation from the glm R package (— — —), and from the MCMC run (— — —)): (a), (c) β_1; (b), (d) β_2

As before, the `glm` R package was used to obtain the MLE and observed Fisher information, which was used within ScaLE to set $\beta^* = \hat{\mathbf{x}} \approx (2.3581 \times 10^{-4}, 2.3407, -2.0009, 1.9995)$ and $\mathbf{\Lambda} \approx \text{diag}(7.6238 \times 10^{-4}, 1.3202, 1.5137 \times 10^{-3}, 1.5138 \times 10^{-3})$ respectively. For the control variate $\nabla \log\{\pi(\hat{\mathbf{x}})\} \approx (2.0287 \times 10^{-9}, 2.2681 \times 10^{-9}, -2.3809 \times 10^{-6}, -2.3808 \times 10^{-6})$ was calculated by using the full data set and as expected (and required for computational considerations) is extremely small, along with $\Delta \log\{\pi(\hat{\mathbf{x}})\}$.

ScaLE was then applied to this example, using $N = 2^{10}$ particles initialized by using a normal approximation given by the computed $\hat{\mathbf{x}}$ and $\mathbf{\Lambda}$, and a subsampling mechanism of size 2. The simulation was run for 20 h, in which time 84935484 individual records of the data set were accessed (equivalent to roughly 8.5 full data evaluations). Trace plots for the simulation can be found in Fig. 2, along with posterior marginals given by the output (after discarding as burn-in a tenth of the simulation). The posterior marginals are overlaid with the normal approximation given by the R `glm` fit.

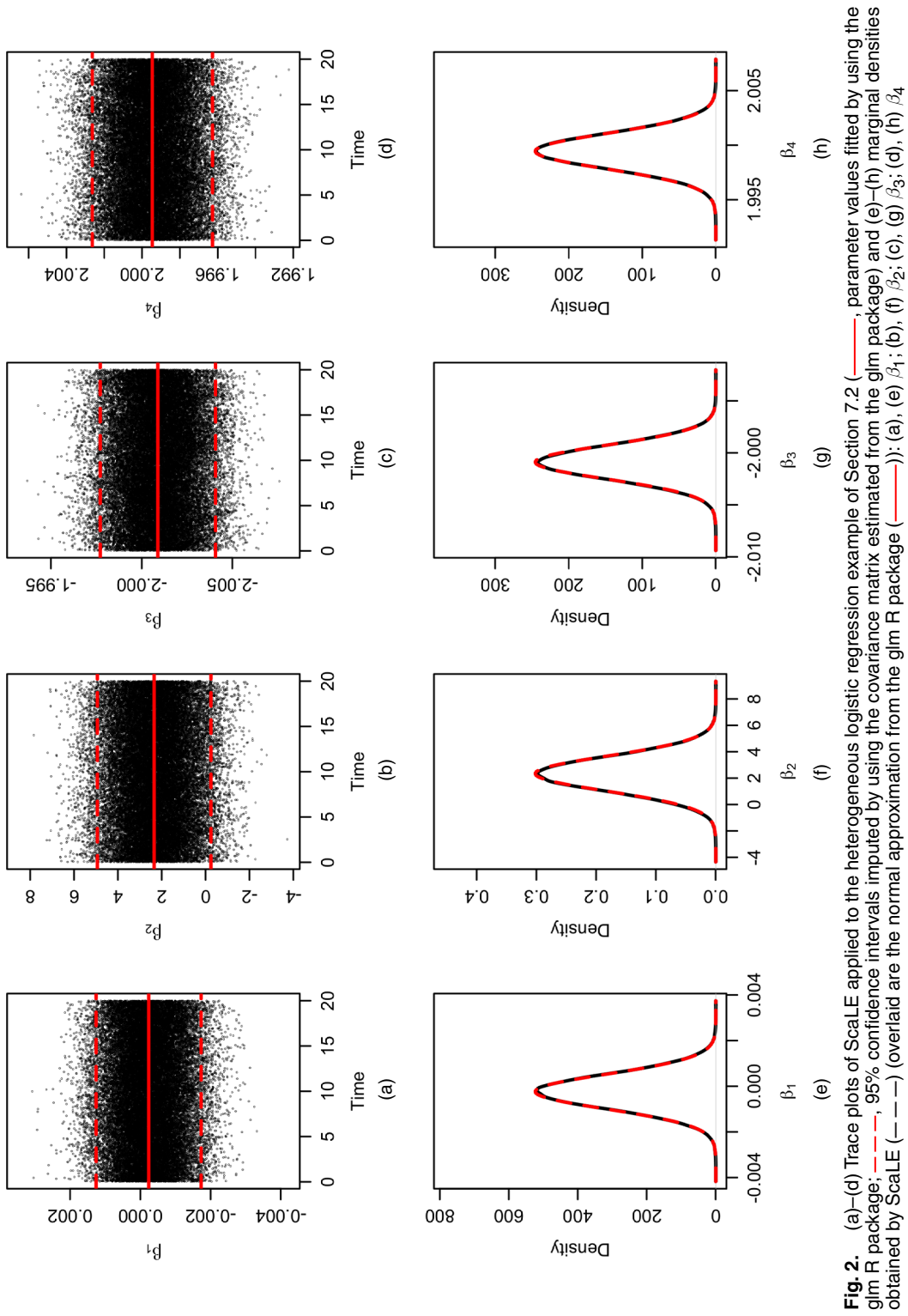

Fig. 2. (a)–(d) Trace plots of ScaLE applied to the heterogeneous logistic regression example of Section 7.2 (———, parameter values fitted by using the glm R package; – – –, 95% confidence intervals imputed by using the covariance matrix estimated from the glm package) and (e)–(h) marginal densities obtained by ScaLE (— — —) (overlaid are the normal approximation from the glm R package (———)): (a), (e) β_1; (b), (f) β_2; (c), (g) β_3; (d), (h) β_4

The estimated means and standard deviations for the regression parameters were $\mathbf{x} \approx (-2.3194 \times 10^{-4}, 2.3197, -2.0009, 1.9995)$ and $\sigma_{\mathbf{x}} \approx (7.6703 \times 10^{-4}, 1.3296, 1.6386 \times 10^{-4}, 1.6217 \times 10^{-4})$ respectively. This is in contrast with β^* and standard deviations of approximately $(7.6238 \times 10^{-4}, 1.3203, 1.6237 \times 10^{-4}, 1.6233 \times 10^{-4})$ from the `glm` output.

To assess the quality of the output we adopted a standard method for estimating the ESS for a single parameter. In particular, we first estimated a marginal ESS associated with the particles from ScaLE at a single time point, with this defined as the average of the ratio of the variance of the estimator of the parameter by using these particles to the posterior variance of the parameter (Carpenter *et al.*, 1999). To calculate the overall ESS, the dependence of these estimators over time is accounted for by modelling this dependence as an AR(1) process. Full details of this approach are given in Appendix I. The resulting average ESS per parameter by using this approach was found to be 352.

The ScaLE output is highly stable and demonstrates that, despite the heterogeneity in the information for different parameters, the Bernstein–von Mises limit (Laplace approximation) proves here to be an excellent fit. Although the generalized linear model fit is therefore excellent in this case, ScaLE can be effectively used to verify this. This is in contrast with the example in Section 7.1 where ScaLE demonstrates that the generalized linear model–Laplace approximation is a poor approximation of the posterior distribution.

7.3. Airline data set

To demonstrate our methodology applied to a real (and moderately large) data set we consider the 'Airline on-time performance' data set which was used for the 2009 American Statistical Association data expo and can be obtained from `http://stat-computing.org/dataexpo/2009/`. The 'airline' data set consists in its entirety of a record of all flight arrival and departure details for all commercial flights within the USA from October 1987 to April 2008. In total the data set comprises 123534969 such flights together with 29 covariates.

For the purposes of this example we selected a number of covariates to investigate what effect (if any) they may have on whether a flight is delayed. The Federal Aviation Administration considers an arriving flight to be late if it arrives more than 15 min later than its scheduled arrival time. As such we take the *flight arrival delay* as our observed data (given by ArrDelay in the Airline data) and treat it as binary, taking a value of 1 for any flight delayed in excess of the Federal Aviation Administration definition.

In addition to an intercept, we determine three further covariates which may reasonably affect flight arrival: a *weekend* covariate, which we obtain by treating DayOfWeek as binary, taking a value of 1 if the flight operated on a Saturday or Sunday, a *night flight* covariate, which we obtain by taking DepTime (departure time) and treating it as binary, taking a value of 1 if the departure is between 8 p.m. and 5 a.m., and flight *distance*, which we obtain by taking Distance and normalizing by subtracting the minimum distance and dividing by the range.

The resulting data set that was obtained by the above process contained some missing entries, and so all such flights were omitted from the data set (in total 2786730 rows), leaving $n = 120748238$ rows. We performed logistic regression taking the *flight arrival delay* variable as the response and treating all other variables as covariates.

To allow computation of $\hat{\mathbf{x}}$ and $\boldsymbol{\Lambda}$ as required by ScaLE the data were first divided into 13 subsets each of size 9288326 and the MLE and observed information matrix for each were obtained by using the R `glm` package. The airline data set is highly structured, and so for robustness the order of the flight records was first permuted before applying `glm` to the data subsets. An estimate for the MLE and observed information matrix for the full data set was obtained by simply taking the mean for each coefficient of the subset MLE fits, and summing

the subset information matrices. The centring point $\hat{\mathbf{x}} \approx (-1.5609, -0.1698, 0.2823, 0.9865)$ was chosen to be the computed MLE fit, and for simplicity $\mathbf{\Lambda}^{-1}$ was chosen to be the square root of the diagonal of the computed information matrix ($\mathbf{\Lambda} \approx \text{diag}(2.309470 \times 10^{-4}, 4.632830 \times 10^{-4}, 6.484359 \times 10^{-4}, 1.2231 \times 10^{-5})$). As before, and as detailed in Section 4.2, we use the full data set to compute $\nabla \log\{\pi(\hat{\mathbf{x}})\} \approx (0.00249, 0.0018, 0.0021, 0.0029)$ (which again is small as suggested by the theory, and required for efficient implementation of ScaLE) and $\Delta \log\{\pi(\hat{\mathbf{x}})\} \approx -3.999$.

ScaLE was initialized by using the normal approximation that is available from the `glm` fit. In total $N = 2^{12}$ particles were used in the simulation, and for computing the unbiased estimator $\tilde{\phi}_A(\mathbf{x})$ we used a subsample of size 2. The algorithm was executed so that n individual records of the data set were accessed (i.e. a single access to the full data set), which took 36 h of computational time. The first tenth of the simulation trajectories were discarded as burn-in and the remainder used to estimate the posterior density. The trace plots and posterior densities for each marginal for the simulation can be found in Fig. 3.

For comparison, we also ran SGLD (Welling and Teh, 2011). This algorithm approximately simulates from a Langevin diffusion which has the posterior distribution as its stationary distribution. The approximation comes from both simulating an Euler discretized version of the Langevin diffusion and from approximating gradients of the log-posterior at each iteration. The approximation error can be controlled by tuning the step size of the Euler discretization—with smaller step sizes meaning less approximation but slower mixing. We implemented SGLD by using a decreasing step size, as recommended by the theoretical results of Teh *et al.* (2016) and used pilot runs to choose the smallest scale for the step size schedule which still led to a well mixing algorithm. As such, the preprocessing expenditure matched that of ScaLE. The accuracy of the estimate of the gradient is also crucial to the performance of SGLD (Dalalyan and Karagulyan, 2019), and we employed an estimator that used control variates (similar to that developed in ScaLE) and a mini-batch size of 1000, following the guidance of Baker *et al.* (2019), Nagapetyan *et al.* (2017) and Brosse *et al.* (2018). For comparable results we ensured that SGLD had the same number of log-likelihood evaluations as ScaLE (i.e. equivalent to one single access to the full data set) and initiated SGLD from the centring value that was used for the control variates. In total the SGLD simulation took 4 h to execute. The first tenth was discarded as burn-in and the remainder was used to estimate the marginal posteriors, which are overlaid with those estimated by ScaLE in Fig. 3.

As can be seen in Fig. 3, SGLD estimates seem to be unstable here, with the algorithm struggling to mix effectively under the decreasing step size constraint, particularly for the fourth covariate. Indeed, the marginal posteriors should be convex and SGLD deviates strongly from this. This unstable behaviour was confirmed in replicate SGLD runs, and indeed it would be difficult to separate out bias from Monte Carlo error for SGLD without much longer runs. This is in contrast with ScaLE which produces far more stable output in this example.

7.4. Large data scenario

In this subsection we consider an application of ScaLE to a five-dimensional logistic regression model, considering data sets of up to size $n = 2^{34} \approx 10^{10.2}$. Logistic regression is a model that is frequently employed within big data settings (Scott *et al.*, 2016), and here the scalability of ScaLE is illustrated for this canonical model. In this example, we generate a data set of size 2^{34} from model (24) by first constructing a design matrix in which the ith entry $\mathbf{x}_i := (1, \zeta_{i,1}, \ldots, \zeta_{i,4})^{\mathrm{T}}$, where $\zeta_{1,1}, \ldots, \zeta_{n,4}$ are independent and identically distributed truncated normal random variables with support $[-1, 1]$. In the big data setting it is natural to assume such control on the

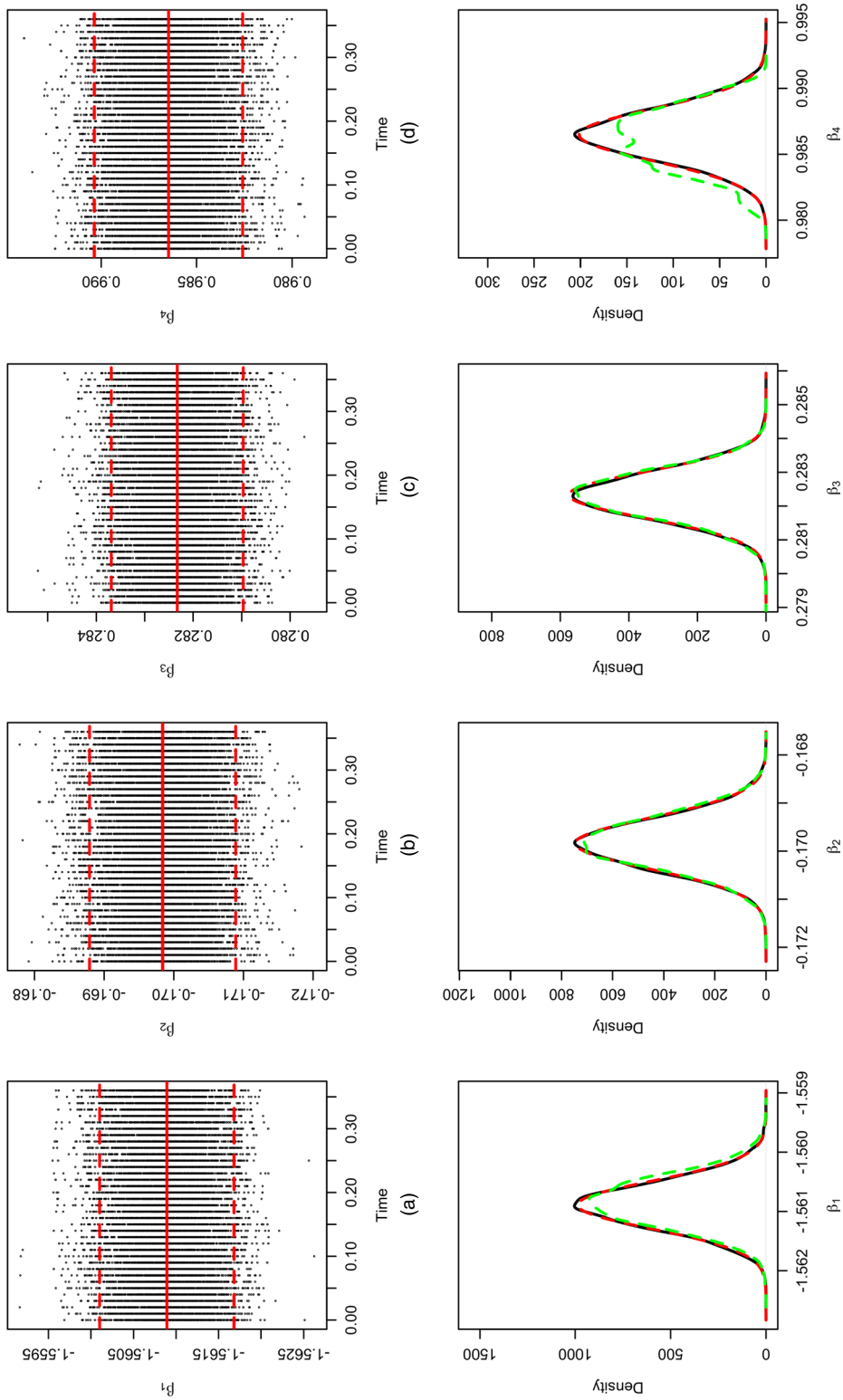

Fig. 3. (a)–(d) Trace plots of ScaLE applied to the airline data set (———, parameter values fitted by using the glm R package; – – –, 95% confidence intervals imputed by using the covariance matrix estimated from the glm package) and (e)–(h) marginal densities obtained by ScaLE (— — —) (overlaid are the normal approximation from the glm R package (– – –) and from the comparable SGLD run (— – —)): (a), (e) β_1; (b), (f) β_2; (c), (g) β_3; (d), (h) β_4

extreme entries of the design matrix, either through construction or physical limitation. On simulating the design matrix, binary observations are obtained by simulation using the parameters $\beta = (1, 1, -1, 2, -2)^{\mathrm{T}}$. Because of the extreme size of the data we realized observations only as they were required to avoid storing the entire data set; see the code provided for implementation details.

First considering the data set of size $n = 2^{34}$, then following the approach that was outlined in Section 7.3, $\hat{\mathbf{x}}$ and $\boldsymbol{\Lambda}$ were chosen by breaking the data into a large number of subsets, fitting the R `glm` package to each subset, then appropriately pooling the fitted MLE and observed Fisher information matrices. In total the full data set was broken into 2^{13} subsets of size 2^{21}, and the `glm` fitting and pooling were conducted entirely in parallel on a network of 100 cores. Consequently, $\hat{\mathbf{x}} = \beta^* \approx (0.9999943, 0.9999501, -0.9999813, 1.999987, -1.999982)$ and $\boldsymbol{\Lambda} \approx \mathrm{diag}(1.9710 \times 10^{-5}, 3.6921 \times 10^{-5}, 3.6910 \times 10^{-5}, 3.8339 \times 10^{-5}, 3.8311 \times 10^{-5})$. On computing $\hat{\mathbf{x}}$ an additional pass of the 2^{13} subsets of the data of size 2^{21} was conducted in parallel to compute $\nabla \log\{\pi(\hat{\mathbf{x}})\} \approx (-0.0735, -0.0408, 0.0428, -0.09495, 0.0987)$ and $\Delta \log\{\pi(\hat{\mathbf{x}})\} \approx -5.006$ for construction of the control variate. Fully utilizing the 100 cores that were available the full suite of preprocessing steps required for executing ScaLE (i.e. the computation of both the `glm` fit and control variate) took 27 h of wall clock time.

ScaLE was applied to this example by using $N = 2^{10}$ particles initialized by using a normal approximation given by the available `glm` fit, and a subsampling mechanism of size 2. The simulation was run for 70 h, in which time 49665450 individual records of the data set were accessed (equivalent to roughly 0.0029 full data evaluations). Trace plots for the simulation can be found in Fig. 4. The first tenth of the simulation trajectories was discarded as burn-in and the remainder used to estimate the posterior density of each marginal. These can also be found in Fig. 4, together with the normal approximation to the posterior marginals given by the R `glm` fit, which is again very accurate here, agreeing closely with the ScaLE output. Using the ESS approach that was described in Section 7.2 and Appendix I, the average ESS per parameter was found to be 553.

To investigate scaling with data size for this example, we considered the same model using the same process as outlined above with data sets varying in size by a factor of 2 from $n = 2^{21}$ to $n = 2^{33}$. Computing explicitly the dominating intensity $\tilde{\lambda}_{n,K}$ over the support of the density the relative cost of ScaLE for each data set with respect to the data set of size $n = 2^{34}$ can be inferred. This is shown in Fig. 5.

7.5. Contaminated mixture

In this subsection we consider parameter inference for a contaminated mixture model. This is motivated by big data sets obtained from Internet applications, in which the large data sets are readily available, but the data are of low quality and corrupted with noisy observations. In particular, in our example each datum comprises two features and a model is fitted in which the likelihood of an individual observation (y_i) is

$$F_i := \frac{1-p}{\sqrt{(2\pi\sigma^2)}} \exp\left\{ -\frac{1}{2\sigma^2}(\alpha x_{i,1} + \beta x_{i,2} - y_i)^2 \right\} + \frac{p}{\sqrt{(2\pi\phi^2)}} \exp\left(-\frac{1}{2\phi^2} y_i^2 \right). \tag{25}$$

In this model p represents the level of corruption and ϕ the variance of the corruption. A common approach uses MCMC sampling with data augmentation (Tanner and Wong, 1987). However, for large data sets this is not feasible as the dimensionality of the auxiliary variable vector will be $\mathcal{O}(n)$. For convenience a transformation of the likelihood was made so that each transformed parameter is on $\mathbb{R}$. The details have been omitted, and the results that are presented are given under the original parameterization.

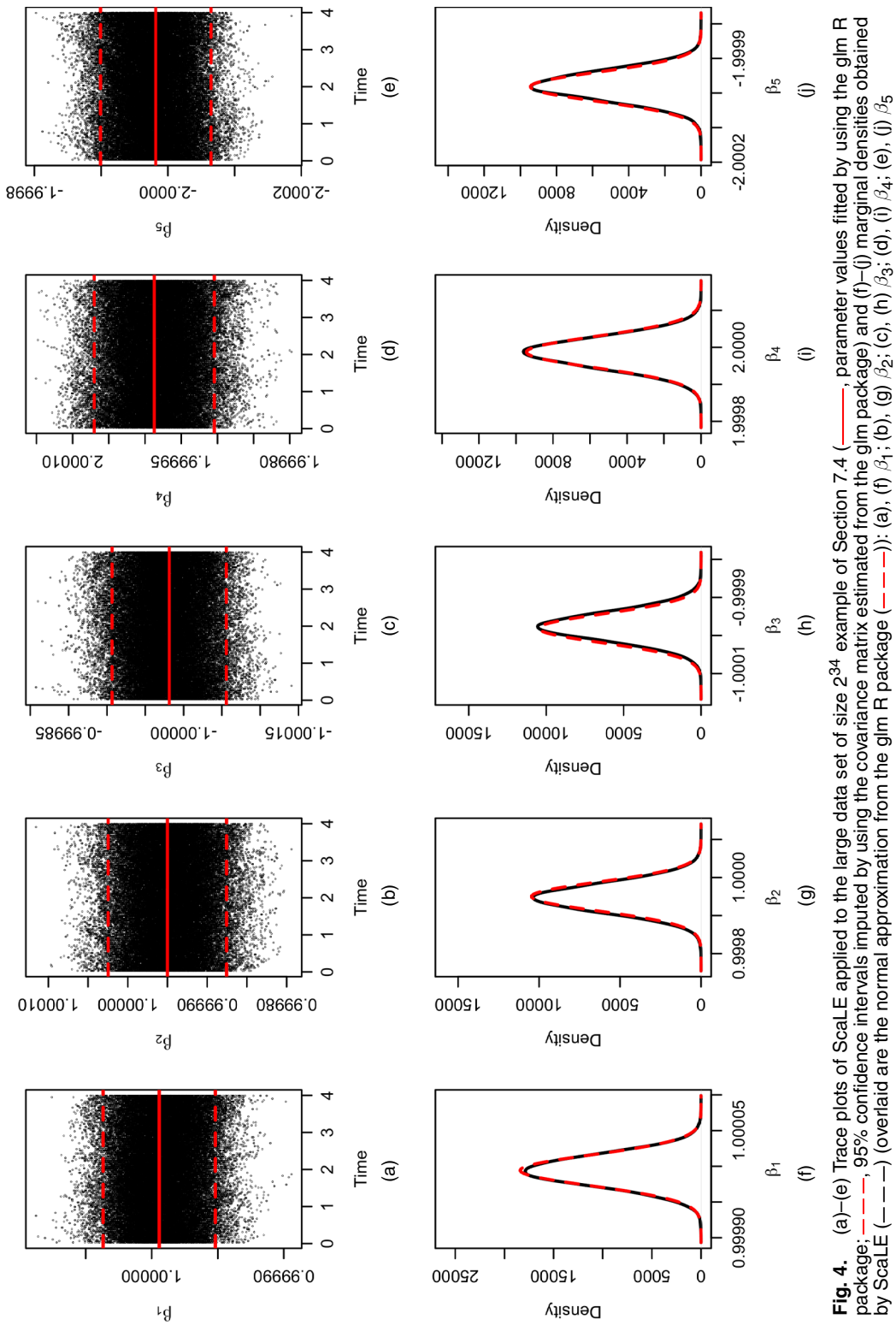

Fig. 4. (a)–(e) Trace plots of ScaLE applied to the large data set of size 2^{34} example of Section 7.4 (———, parameter values fitted by using the glm R package; – – –, 95% confidence intervals imputed by using the covariance matrix estimated from the glm package) and (f)–(j) marginal densities obtained by ScaLE (— — —) (overlaid are the normal approximation from the glm R package (– – –)): (a), (f) β_1; (b), (g) β_2; (c), (h) β_3; (d), (i) β_4; (e), (j) β_5

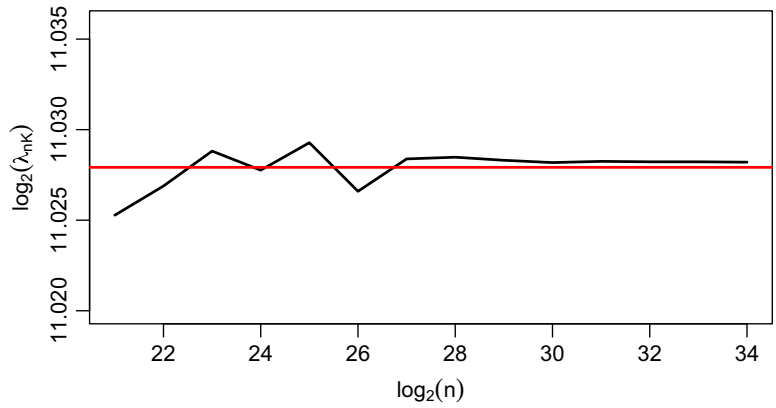

Fig. 5. Comparison of the bounding intensities and comparative cost for executing ScaLE for increasing data set sizes in the large data example of Section 7.4

A data set of size $n = 2^{27} \approx 10^{8.1}$ was generated from the model with parameters $\mu = [\alpha, \beta, \sigma, \phi, p] = [2, 5, 1, 10, 0.05]$. To illustrate a natural future direction for the ScaLE methodology, in this example we instead implemented an approximate version of ScaLE (as opposed to the exact version that was illustrated in the other examples of Section 7). In particular, the primary implementational and computational bottleneck in ScaLE is the formal 'localization procedure' to obtain almost sure bounds on the killing rate by constraining Brownian motion to a hypercube (as fully detailed in Section 3.2 and Appendix C). Removing the localization procedure results in the Brownian motion trajectories being unconstrained, and the resulting dominating intensity $\tilde{\lambda}$ being infinite. However, in practice the probability of such an excursion by Brownian motion outside a suitably chosen hypercube can be made vanishingly small (along with the consequent effect on the Monte Carlo output) by simply adjusting the temporal resolution at which the ergodic average is obtained from the algorithm (noting that Brownian motion scaling is $\mathcal{O}(\sqrt{t})$, and inflating the bounds on the Hessian for computing the intensity. The resulting 'approximate' algorithm is approximate in a different (more controllable and monitorable) sense than for instance SGLD, but results in substantial (10–50 times) computational speed-ups over the available (but expensive) 'exact' ScaLE.

In contrast with the other examples of Section 7, rather than fitting an approximate model to initialize the algorithm, instead in this example a single point mass to initialize the algorithm was chosen ($\mu = [2.00045, 5.00025, 0.999875, 10.005\ 0.0499675]$), and this was also used as the point to compute our control variate (described in Section 4.2). The preprocessing for executing ScaLE took approximately 6 h o f computational time (and is broadly indicative of the length of time that a single iteration of an alternative MCMC scheme such as the Metropolis adjusted Langevin algorithm would require). As discussed in Section 5, this 'misinitialization' impacts the efficiency of the algorithm by a constant factor but is, however, representative of what one in practice may conceivably be able to do (i.e. find by means of an optimization scheme a point within the support of the target posterior close to some mode and conduct a single $\mathcal{O}(n)$ calculation). The forgetting of this initialization is shown in Fig. 6.

Applying ScaLE for this application we used a particle set of size $N = 2^{11}$ and ran the algorithm for a diffusion time of $T = 200$, with observations of each trajectory at a resolution of $t_i - t_{i-1} = 0.1$. Again, the choice of N was made as in Section 7.4 as it provided the required stability. The choice of T was made as it corresponded approximately to a computational budget of 1 week.

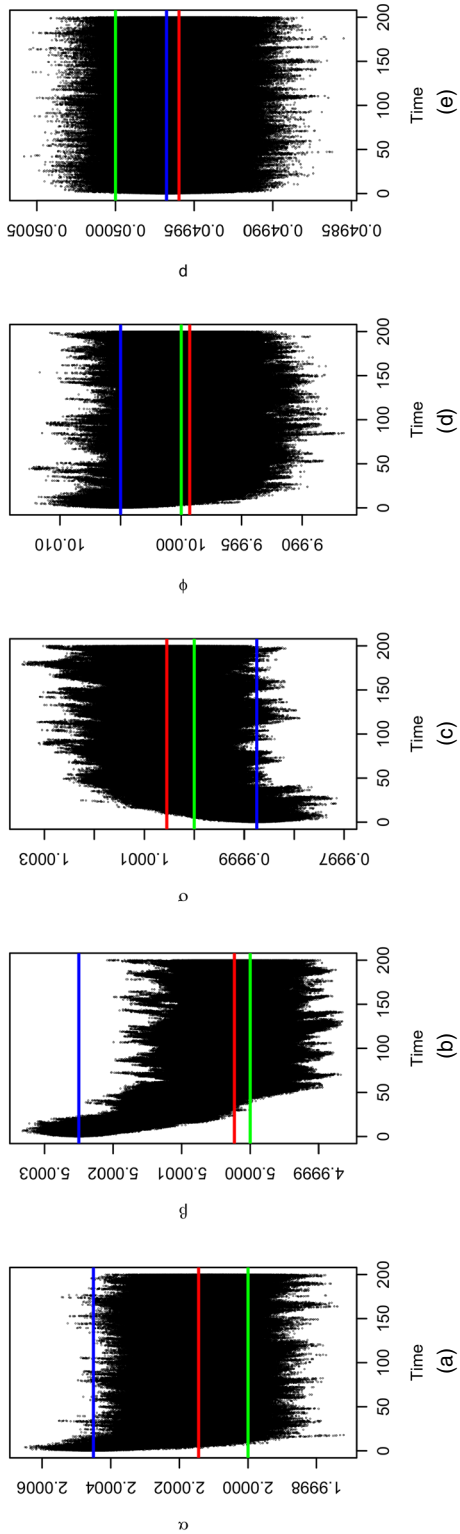

Fig. 6. (a)–(e) Trace trajectories of ScaLE applied to the contaminated mixture data set and (f)–(j) marginal densities (——, |, parameter values used to initialize the algorithm; ——, |, parameter values that the associated data set was generated from; ——, |, mean of the marginal densities): (a), (f) α; (b), (g) β; (c), (h) σ; (d), (i) ϕ; (e), (j) ρ

Each particle trajectory at each time $t \in [0, T]$ was associated with a sub-sample of the full data set of size 32, rather than 2, with the resulting likelihood estimates combined by simple averaging. This size was chosen as it provided balance with other components of the algorithm but allowed stabilization of the importance weights which was beneficial for the approximate algorithm. In total the entire run required accessing 500 million individual data points, which correspond to approximately four full evaluations of the data set.

An example of a typical run can be found in Fig. 6. A burn-in period of 100 was chosen, and alongside the trace plots in Fig. 6 an estimate of the marginal density of the parameters is provided by using the occupation measure of the trajectories in the interval $t \in [100, 200]$.

To assess the quality of the simulation, the same batch mean method is employed to estimate the marginal ESS for the run post burn-in as detailed in Section 7.4. The mean ESS per dimension for this run was around 930. An analysis of the Metropolis adjusted Langevin algorithm (for a necessarily much smaller run, indicated that it is possible to achieve an ESS of around $T/3$, where T corresponds to the run length subsequent to burn-in. As indicated above, and neglecting burn-in, this would mean an achievable ESS for a comparable computational budget for the Metropolis adjusted Langevin algorithm would be around 10–15.

8. Conclusions

In this paper we have introduced a new class of QSMC methods which are genuinely continuous time algorithms for simulating from complex target distributions. We have emphasized its particular effectiveness in the context of *big data* by developing novel subsampling approaches and the scalable Langevin exact algorithm ScaLE. Unlike its immediate competitors, our subsampling approach within ScaLE is essentially computationally free and does not result in any approximation to the target distribution. Our methodology is embedded within an SMC framework, supported by underpinning theoretical results. In addition, examples to which ScaLE is applied demonstrate its robust scaling properties for large data sets.

We show through theory and examples that the computational cost of ScaLE is more stable to data set size than *gold standard* MCMC approaches. Moreover we have seen it substantially outperform other approaches such as SGLD which are designed to be robust to data size at the cost of bias and serial correlation. ScaLE can both confirm that simpler approaches such as Laplace approximation are accurate and identify when such approximations are poor (as we see in the examples). We see this as a first step in a fruitful new direction for computational statistics. Many ideas for variations and extensions to our implementation exist and will stimulate further investigation.

Firstly, the need to simulate a quasi-stationary distribution creates particular challenges. Although quasi-stationarity is underpinned by an elegant mathematical theory, the development of numerical methods for quasi-stationarity is understudied. We have presented an SMC methodology for this problem, but alternatives exist. For instance, Blanchet, *et al.* (2016) have suggested alternative approaches.

Even within an SMC framework for extracting the quasi-stationary distribution, there are interesting alternatives that we have not explored. For example, by a modification of our reweighting mechanism it is possible to relate the target distribution of interest to the limiting *smoothing* distribution of the process, as opposed to the filtering distribution as we do here. Within the quasi-stationary literature this is often termed the type II quasi-stationary distribution. As such, the rich SMC literature offers many other variations on the procedures that were adopted here.

Using SMC methods benefits from the rich theory that they have. However, the use of QSMC methods actually demands new questions of SMC methods. Theorem 1 gives convergence as

$T \to \infty$, whereas proposition 2 gives a precise description of the limit as the number of particles N increases. Theoretical and practical questions are associated with letting both N and T tend to ∞ together. Within the examples in this paper *ad hoc* rules are used to assign computational effort to certain values of N and T. However, the general question of how to choose these parameters seems completely open.

Throughout the paper, we have concentrated on so-called *exact approximate* QSMC methods. Of course in many cases good approximations are sufficiently good and frequently computationally less demanding. However, for many approximate methods it will be difficult to quantify the systematic error being created by the approximation. Moreover, we emphasize that there are different strategies for creating effective approximations that emanate directly from exact approximate methods, and where the approximation error can be well understood. We have given an example of this in section 7.5 but other options are possible also.

There are interesting options for parallel implementation of SMC algorithms in conjunction with ScaLE. For instance an appealing option would be to implement the island particle filter (Del Moral *et al*., 2016) which could have substantial effects on the efficiency of our algorithms where large numbers of particles are required. Alternatively one could attempt to embed our scheme in other divide-and-conquer schemes as described in Section 1.

The approach in this paper has concentrated solely on killed (or reweighted) Brownian motion, and this strategy has been demonstrated to have robust convergence properties. However, given existing methodology for the exact simulation of diffusions in Beskos and Roberts (2005), Beskos *et al*. (2006, 2008), Pollock (2013, 2015) and Pollock *et al*. (2016), there is scope to develop methods which use proposal measures which much better *mimic* the shape of the posterior distribution.

The subsampling and control variate approaches that were developed here offer dramatic computational savings for tall data as we see from the examples and from the theory of results like theorem 3. We have not presented ScaLE as a method for high dimensional inference, and the problem of large n and d will inevitably lead to additional challenges. However, there may be scope to extend the ideas of ScaLE still further in this direction. For instance, it might be possible to subsample dimensions and thus to reduce the dimensional complexity for implementing each iteration.

We conclude by noting that, as a by-product, the theory behind our methodology offers new insights into problems concerning the existence of quasi-stationary distributions for diffusions killed according to a state-dependent hazard rate, complementing and extending current state of the art literature (Steinsaltz and Evans, 2007).

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (grants EP/D002060/1, EP/K014463/1, EP/N031938/1, EP/R018561/1 and EP/R034710/1) and two Alan Turing Institute programmes; the Lloyd's Register Foundation programme on 'Data-centric engineering' and the UK Government's 'Defence and security programme'. The authors thank Louis Aslett, Tigran Nagapetyan and Andi Q. Wang for useful discussions on aspects of this paper. In addition, we thank the referees for some very helpful suggestions and questions which have improved the exposition of this paper.

Appendix A: Proof of theorem 1

Here we present a proof of theorem 1. However, we first formally state the required regularity conditions. We suppose that

$$\pi(x) \text{ is bounded,} \tag{26}$$

and, defining $\nu(x) = \pi^2(x)$, we further assume that, for some $\gamma > 0$,

$$\liminf_{\mathbf{x}\to\infty} \left\{ \frac{\Delta\nu(\mathbf{x})}{\nu^{\gamma+1/2}(\mathbf{x})} - \frac{\gamma\|\nabla\nu(\mathbf{x})\|^2}{\nu^{\gamma+3/2}(\mathbf{x})} \right\} > 0, \tag{27}$$

where Δ represents the Laplacian.

Proof. Consider the diffusion with generator given by

$$\mathfrak{A}f(\mathbf{x}) = \tfrac{1}{2}\Delta f(\mathbf{x}) + \tfrac{1}{2}\nabla \log\{\nu(\mathbf{x})\}\nabla f(\mathbf{x}).$$

As ν is bounded, we assume without loss of generality that its upper bound is 1. Our proof will proceed by checking the conditions of corollary 6 of Fort and Roberts (2005), which establishes the result. In particular, we need to check that the following conditions are satisfied.

Condition 3. For all $\delta > 0$, the discrete time chain $\{X_{n\delta}, n = 0, 1, 2, \ldots\}$ is irreducible.

Condition 4. All closed bounded sets are petite.

Condition 5. We can find a drift function $V(\mathbf{x}) = \nu(\mathbf{x})^{-\gamma}$, for some $\gamma > 0$, that satisfies the condition

$$\mathfrak{A}V^{\eta}(\mathbf{x}) \leqslant -c_{\eta}V(\mathbf{x})^{\eta-\alpha} \tag{28}$$

for $\mathbf{x}$ outside some bounded set, for each $\eta \in [\alpha, 1]$ with associated positive constant c_{η}, and where $\alpha = 1 - (2\gamma)^{-1}$.

Condition 3 holds for any regular diffusion since the diffusion has positive continuous transition densities over time intervals $t > 0$; and positivity and continuity of the density also imply condition 4. For the final condition we require that

$$\limsup_{|\mathbf{x}|\to\infty} \frac{\mathfrak{A}V^{\eta}(\mathbf{x})}{V^{\eta-\alpha}(\mathbf{x})} < 0. \tag{29}$$

Now by direct calculation

$$\mathfrak{A}V^{\eta}(\mathbf{x}) = \frac{\eta\gamma}{2}\nu(\mathbf{x})^{-\gamma\eta-2}\{\eta\gamma\|\nabla\nu(\mathbf{x})\|^2 - \nu(\mathbf{x})\Delta\nu(\mathbf{x})\}, \tag{30}$$

so that

$$\frac{\mathfrak{A}V^{\eta}(\mathbf{x})}{V(\mathbf{x})^{\eta-\alpha}} = \frac{\eta\gamma\nu(\mathbf{x})^{-3/2-\gamma}}{2}\{\eta\gamma\|\nabla\nu(\mathbf{x})\|^2 - \nu(\mathbf{x})\Delta\nu(\mathbf{x})\}. \tag{31}$$

Therefore inequality (29) will hold whenever inequality (27) is true since we have the constraint that $\eta \leqslant 1$ and $\|\nabla\nu(\mathbf{x})\|^2$ is clearly non-negative. As such the result holds as required. □

Note that the condition in inequality (27) is essentially a condition on the tail of ν. This will hold even for heavy-tailed distributions, and we show that this is so for a class of one-dimensional target densities in Appendix B.

Appendix B: Polynomial tails

In this appendix we examine condition (27) which we use within theorem 1. This is essentially a condition on the tail of ν, and so we examine the critical case in which the tails of ν are heavy. More precisely, we demonstrate for polynomial tailed densities in one dimension that condition (27) essentially amounts to requiring that $\nu^{1/2}$ is integrable. Recall that by construction $\nu^{1/2}$ will be integrable as we have chosen $\nu^{1/2} = \pi$.

For simplicity, suppose that ν is a density on $[1, \infty)$ such that $\nu(x) = x^{-p}$. In this case we can easily compute that, for $p > 1$,

$$\nabla\nu(x) = -px^{-p-1},$$
$$\Delta\nu(x) = p(p+1)x^{-p-2}$$

from which we can easily compute the quantity whose limit is taken in inequality (27) as

$$x^{p(\gamma-1/2)-2}\{p(p+1)-\gamma p^2\}.$$

As such, we have that condition (27) holds if and only if

$$p+1-\gamma p>0 \tag{32}$$

and

$$p(\gamma-\tfrac{1}{2})-2\geqslant 0. \tag{33}$$

Now we shall demonstrate that we can find such γ for all $p>2$. For instance, suppose that $p=2+\epsilon$. The case $\epsilon\geqslant 2$ can be handled by just setting $\gamma=1$, so suppose otherwise and set $\gamma=\frac{3}{2}-\epsilon/4$. In this case, inequality (33) just gives $\epsilon/2-\epsilon^2/4\geqslant 0$. Moreover expression (32) becomes $3\epsilon/2+\epsilon^2>0$, completing our argument.

Appendix C: Simulation of a path space layer and intermediate points

In this appendix we present the methodology and algorithms that are required for simulating an individual proposal trajectory of (layered) killed multivariate Brownian motion, which is what is required in Section 3. Our exposition is as follows. In Appendix C.1 we present the work of Devroye (2009), in which a highly efficient rejection sampler was developed (based on the earlier work of Burq and Jones (2008)) for simulating the first-passage time for univariate standard Brownian motion for a given symmetric boundary, extending it to consider the case of the univariate first-passage times of d-dimensional standard Brownian motion with non-symmetric boundaries. This construction enables us to determine an interval (given by the first, first-passage time) and layer (a hypercube inscribed by the user-specified univariate boundaries) in which the sample path is almost surely constrained, and by application of the strong Markov property can be applied iteratively to find, for any interval of time, a layer (a concatenation of hypercubes) which almost surely constrains the sample path. In Appendix C.2 we present a rejection sampler enabling the simulation of constrained univariate standard Brownian motion as developed in Appendix C.1, at any desired intermediate point. As motivated in Section 3 these intermediate points may be at some random time (corresponding to a proposed killing point of the proposed sample path), or a deterministic time (in which the sample path is extracted for inclusion within the desired Monte Carlo estimator of QSMC (7)). Finally, in Appendix C.3 we present the full methodology that is required in Sections 3 and 4 in which we simulate multivariate Brownian motion at any desired time marginal, with d-dimensional hypercubes inscribing intervals of the state space in which the sample path almost surely lies.

C.1. Simulating the first-passage times of univariate and multivariate standard Brownian motion

To begin with we restrict our attention to the (ith) dimension of multivariate standard Brownian motion initialized at 0, and the first-passage time of the level $\theta^{(i)}$ (which is specified by the user). In particular we denote

$$\tau^{(i)} := \inf\{t\in\mathbb{R}+ : |W_t^{(i)} - W_0^{(i)}|\geqslant\theta^{(i)}\}. \tag{34}$$

Recalling the self-similarity properties of Brownian motion (Karatzas and Shreve (1991), section 2.9), we can further restrict our attention to the simulation of the first-passage time of univariate Brownian motion of the level 1, noting that $\tau^{(i)} =^{\mathcal{D}} (\theta^{(i)})^2\bar{\tau}$ where

$$\bar{\tau} := \inf\{t\in\mathbb{R}+ : |W_t - W_0|\geqslant 1\}, \tag{35}$$

noting that, at this level,

$$\mathbb{P}(W_\tau = W_0+1) = \mathbb{P}(W_\tau = W_0-1) = \tfrac{1}{2}. \tag{36}$$

Denoting by $f_{\bar{\tau}}$ the density of $\bar{\tau}$ (which cannot be evaluated pointwise), the approach that was outlined in Devroye (2009) for drawing random samples from $f_{\bar{\tau}}$ is a series sampler. In particular, an accessible dominating density of $f_{\bar{\tau}}$ is found (denoted $g_{\bar{\tau}}$) from which exact proposals can be made; then upper and lower monotonically convergent bounding functions are constructed ($\lim_{n\to\infty} f^{\uparrow}_{\bar{\tau},n} \to f_{\bar{\tau}}$ and $\lim_{n\to\infty} f^{\downarrow}_{\bar{\tau},n} \to f_{\bar{\tau}}$ such that for any $t \in \mathbb{R}_+$ and $\epsilon > 0$ $\exists n^*(t,\epsilon)$ such that $\forall n \geqslant n^*(t,\epsilon)$ we have $f^{\uparrow}_{\bar{\tau},n}(t) - f^{\downarrow}_{\bar{\tau},n}(t) < \epsilon$), and then evaluated to sufficient precision such that acceptance or rejection can be made while retaining exactness. A minor complication arises in that no known, tractable dominating density is uniformly efficient on $\mathbb{R}_+$, and furthermore no single representation of the bounding function converges monotonically to the target density pointwise on $\mathbb{R}_+$. As such, the strategy that was deployed by Devroye (2009) is to exploit a dual representation of $f_{\bar{\tau}}$ given by Ciesielski and Taylor (1962) to construct a hybrid series sampler, using one representation of $f_{\bar{\tau}}$ for the construction of a series sampler on the interval $(0, t_1]$ and the other representation for the interval $[t_2, \infty)$ (fortunately we have $t_1 > t_2$, and so we have freedom to choose a threshold $t^* \in [t_2, t_1]$ in which to splice the series samplers). In particular, as shown in Ciesielski and Taylor (1962) $f_{\bar{\tau}}(t) = \pi \Sigma_{k=0}^{\infty} (-1)^k a_k(t)$ where the elements of the two expansions are given by

$$a_k(t) = \begin{cases} \left(\frac{2}{\pi t}\right)^{3/2} \left(k + \frac{1}{2}\right) \exp\left\{-\frac{2}{t}\left(k+\frac{1}{2}\right)^2\right\}, & \text{(37a)} \\ \left(k + \frac{1}{2}\right) \exp\left\{-\frac{1}{2}\left(k+\frac{1}{2}\right)^2 \pi^2 t\right\}, & \text{(37b)} \end{cases}$$

and so by consequence upper and lower bounding sequences can be constructed by simply taking either representation and truncating the infinite sum to have an odd or even number of terms respectively (and thresholding to between 0 and the proposal $g_{\bar{\tau}}$, introduced below). More precisely,

$$\begin{aligned} f^{\downarrow}_{\bar{\tau},n}(t) &:= \left(\pi \sum_{k=0}^{2n+1} (-1)^k a_k(t)\right)_+ , \\ f^{\uparrow}_{\bar{\tau},n}(t) &:= \left\{\pi \sum_{k=0}^{2n} (-1)^k a_k(t)\right\} \wedge g_{\bar{\tau}}(t). \end{aligned} \tag{38}$$

As shown in lemma 1 of Devroye (2009), the bounding sequences based on the representation of $f_{\bar{\tau}}(t)$ in equation (37a) are monotonically converging for $t \in (0, 4/\log(3)]$, and for equation (37b) monotonically converging for $t \in [\log(3)/\pi^2, \infty)$. After choosing a suitable threshold $t^* \in [4/\log(3), \log(3)/\pi^2]$ for which to splice the series samplers, then by simply taking the first term in each representation of $f_{\bar{\tau}}(t)$ a dominating density can be constructed as follows:

$$f_{\bar{\tau}}(t) \leqslant g_{\bar{\tau}}(t) \propto \underbrace{\frac{2}{\pi t^{3/2}} \exp\left(-\frac{1}{2t}\right) \mathbb{1}_{t \leqslant t^*}}_{\propto g^{(1)}_{\bar{\tau}}(t)} + \underbrace{\frac{\pi}{2} \exp\left(-\frac{\pi^2 t}{8}\right) \mathbb{1}_{t \geqslant t^*}}_{\propto g^{(2)}_{\bar{\tau}}(t)}. \tag{39}$$

Devroye (2009) empirically optimized the choice of $t^* = 0.64$ to minimize the normalizing constant of expression (39). With this choice $M_1 := \int g^{(1)}_{\bar{\tau}}(t)\mathrm{d}t \approx 0.422599$ (to six decimal places) and $M_2 := \int g^{(2)}_{\bar{\tau}}(t)\mathrm{d}t \approx 0.578103$ (to six decimal places), and so we have a normalizing constant $M = M_1 + M_2 \approx 1.000702$ (to six decimal places) which equates to the expected number of proposal random samples drawn from $g_{\bar{\tau}}$ before we would expect an accepted draw (the algorithmic 'outer loop'). Now considering the iterative algorithmic 'inner loop'—in which the bounding sequences are evaluated to precision sufficient to determine acceptance or rejection—as shown in Devroye (2009), the exponential convergence of the sequences ensures that the number of iterations that are required is uniformly bounded in expectation by 3.

Simulation from $g_{\bar{\tau}}$ is possible by either simulating $\bar{\tau} \sim g^{(1)}_{\bar{\tau}}$ with probability M_1/M, or else or $\bar{\tau} \sim g^{(2)}_{\bar{\tau}}$. Simulating $\bar{\tau} \sim g^{(1)}_{\bar{\tau}}$ can be achieved by noting that, for $t \sim g^1_{\bar{\tau}}$, $t =^{\mathcal{D}} t^* + 8X/\pi^2$, where $X \sim \mathrm{Exp}(1)$. Simulating $\bar{\tau} \sim g^{(2)}_{\bar{\tau}}$ can be achieved by noting that as outlined in Devroye (1986), section IX.1.2, for $t \sim g^2_{\bar{\tau}}$, $t =^{\mathcal{D}} t^*/(1 + t^* X)^2$, where

$$X := \inf_i \{\{X_i\}_{i=1}^{\infty} \overset{\text{IID}}{\sim} \mathrm{Exp}(1) : (X_i)^2 \leqslant 2X_{i+1}/t^*, (i-1)/2 \in \mathbb{Z}\}.$$

Table 5. Algorithm 4: simulating $(\tau, W_\tau^{(i)})$, where $\tau := \inf\{t \in \mathbb{R}+ : |W_t^{(i)} - W_0^{(i)}| \geqslant \theta^{(i)}\}$ (Devroye, 2009)

1 Input $W_0^{(i)}$ and $\theta^{(i)}$
2 $g_{\bar{\tau}}$: simulate $u \sim U[0,1]$:
(a) $g_{\bar{\tau}}^{(1)}$, if $u \leqslant M_1/M$, then simulate $X \sim \text{Exp}(1)$ and set $\bar{\tau} := t^* + 8X/\pi^2$;
(b) $g_{\bar{\tau}}^{(2)}$, if $u > M_1/M$, then set $X := \inf_i\{\{X_i\}_{i=1}^{\infty} \sim^{\text{IID}} \text{Exp}(1) : (X_i)^2 \leqslant 2X_{i+1}/t^*, (i-1)/2 \in \mathbb{Z}\}$ and set $\bar{\tau} := t^*/(1+t^*X)^2$
3 u: simulate $u \sim U[0,1]$ and set $n = 0$
4 $f_{\bar{\tau},n}$: while $u g_{\bar{\tau}}(\bar{\tau}) \in (f_{\bar{\tau},n}^{\downarrow}(\bar{\tau}), f_{\bar{\tau},n}^{\uparrow}(\bar{\tau}))$, set $n = n+1$
5 $f_{\bar{\tau}}$: if $u g_{\bar{\tau}}(\bar{\tau}) \leqslant f_{\bar{\tau},n}^{\downarrow}(\bar{\tau})$ accept; otherwise reject and return to step 2
6 τ: set $\tau := (\theta^{(i)})^2 \bar{\tau}$
7 $W_\tau^{(i)}$: with probability $\frac{1}{2}$ set $W_\tau^{(i)} = W_0^{(i)} + \theta^{(i)}$; otherwise set $W_\tau^{(i)} = W_0^{(i)} - \theta^{(i)}$
8 Return $(\tau, W_\tau^{(i)})$

A summary of the above procedure for simulating jointly the first-passage time and location of the ith dimension of Brownian motion of the threshold level $\theta^{(i)}$ is provided in algorithm 4 in Table 5.

Generalizing to the case where we are interested in the first-passage time of Brownian motion of a non-symmetric barrier, in particular for $l^{(i)}, \upsilon^{(i)} \in \mathbb{R}_+$,

$$\tau^{(i)} := \inf\{t \in \mathbb{R}+ : W_t^{(i)} - W_0^{(i)} \notin (W_0^{(i)} - l^{(i)}, W_0^{(i)} + \upsilon^{(i)})\}, \tag{40}$$

is trivial algorithmically. In particular, using the strong Markov property we can iteratively apply algorithm 4, setting $\theta^{(i)} := \min(l^{(i)}, \upsilon^{(i)})$ and simulating intermediate first-passage times of lesser barriers, halting whenever the desired barrier is attained. We suppress this (desirable) flexibility in the remainder of the paper to avoid the resulting notational complexity.

C.2. Simulating intermediate points of multivariate standard Brownian motion conditioned on univariate first-passage times

Clearly in addition to being able to simulate the first-passage times of a single dimension of Brownian motion, we want to be able to simulate the remainder of the dimensions of Brownian motion at that time, or indeed the sample path at times other than its first-passage times. As the dimensions of standard Brownian motion are independent (and so Brownian motion can be composed by considering each dimension separately), we can restrict our attention to simulating a single dimension of the sample path for an intermediate time $q \in [s, \tau]$ given W_s, the extremal value W_τ, and constrained such that, $\forall\, u \in [s, \tau]$, $W_u \in [W_s - \theta, W_s + \theta]$. Furthermore, as we are interested in only the forward simulation of Brownian motion, by application of the strong Markov property we need to consider only the simulation of a single intermediate point (although by application of Pollock *et al*. (2016), section 7, simulation at times conditional on future information is possible).

To proceed, note that (as outlined in Asmussen *et al*. (1995), proposition 2) the law of a univariate Brownian motion sample path in the interval $[s, \tau]$ (where $s < \tau$) initialized at (s, W_s), and constrained to attain its extremal value at (τ, W_τ), is simply the law of a three-dimensional Bessel bridge. We require the additional constraint that, $\forall\, u \in [s, \tau]$, $W_u \in [W_s - \theta, W_s + \theta]$, which can be imposed in simulation by deploying a rejection sampling scheme in which a Bessel bridge sample path is simulated at a single required point (as above) and accepted if it meets the imposed constraint at either side of the simulated point, and rejected otherwise.

As presented in Beskos *et al*. (2006) and Pollock (2013), the law of a Bessel bridge sample path (parameterized as above) coincides with that of an appropriate time rescaling of three independent Brownian bridge sample paths of unit length conditioned to start and end at the origin (denoted by $\{b^{(i)}\}_{i=1}^3$). Supposing that we require the realization of a Bessel bridge sample path at some time $q \in [s, \tau]$, then by simply realizing three independent Brownian bridge sample paths at that time marginal ($\{b_q^{(i)}\}_{i=1}^3$), we have

$$W_q = W_s + (-1)^{\mathbb{1}(W_\tau < W_s)} \sqrt{\left((\tau - s)\left[\left\{\frac{\theta(\tau - q)}{(\tau - s)^{3/2}} + b_q^{(1)}\right\}^2 + (b_q^{(2)})^2 + (b_q^{(3)})^2\right]\right)}. \tag{41}$$

The method by which the proposed Bessel bridge intermediate point is accepted or rejected (recall, to impose the constraint that, $\forall\, u \in [s, \tau]$, $W_u \in [W_s - \theta, W_s + \theta]$) is non-trivial as a closed form representation of the required probability does not exist (which we shall denote in this appendix by p). Instead, as shown in theorem 4, a representation for p can be found as the product of two infinite series, which as a consequence of this form cannot be evaluated directly to make the typical acceptance–rejection comparison (i.e. determining whether $u \leqslant p$ or $u > p$, where $u \sim U[0, 1]$). The strategy that we deploy to retain exactness and to accept with the correct probability p is that of a retrospective Bernoulli sampler (Pollock *et al.* (2016), section 6.0). In particular, in corollary 1 we construct monotonically convergent upper and lower bounding probabilities ($p_n^{\uparrow}$ and $p_n^{\downarrow}$ respectively) with the property that $\lim_{n\to\infty} p_n^{\uparrow} \to p$ and $\lim_{n\to\infty} p_n^{\downarrow} \to p$ such that for any $u \in [0, 1]$ and $\epsilon > 0$ $\exists\, n^*(t)$ such that $\forall n \geqslant n^*(t)$ we have $p_n^{\uparrow} - p_n^{\downarrow} < \epsilon$, which are then evaluated to sufficient precision to make the acceptance–rejection decision, taking almost surely finite computational time.

Theorem 4. The probability that a three-dimensional Bessel bridge sample path $W \sim \mathbb{W}_{s,\tau}^{W_s, W_\tau}|(W_\tau, W_q)$ for $s < q < \tau$ attaining its boundary value at (τ, W_τ), remains in the interval $[W_s - \theta, W_s + \theta]$, can be represented by the following product of infinite series (where we denote by $m := \mathbb{1}(W_\tau > W_s) - \mathbb{1}(W_\tau < W_s)$),

$$\mathbb{P}(W_{[s,\tau]} \in [W_s - \theta, W_s + \theta] | W_s, W_q, W_\tau) = \underbrace{\frac{1 - \sum_{j=1}^{\infty}\{\varsigma_{q-s}(j; W_s - W_q, \theta) - \varphi_{q-s}(j; W_s - W_q, \theta)\}}{1 - \exp[-2\theta\{m(W_s - W_q) + \theta\}/(q - s)]}}_{=:p_1}$$
$$\times \underbrace{\left[1 + \sum_{j=1}^{\infty}\{\psi_{\tau-q}(j; W_q - W_\tau, \theta, m) + \chi_{\tau-q}(j; W_q - W_\tau, \theta, m)\}\right]}_{=:p_2}, \tag{42}$$

where

$$\varsigma_\Delta(j; \delta, \theta) := 2\exp\left\{-\frac{2\theta^2(2j-1)^2}{\Delta}\right\}\cosh\left\{\frac{2(2j-1)\theta\delta}{\Delta}\right\}, \tag{43}$$

$$\varphi_\Delta(j; \delta, \theta) := 2\exp\left(-\frac{8\theta^2 j^2}{\Delta}\right)\cosh\left(\frac{4\theta\delta j}{\Delta}\right), \tag{44}$$

$$\psi_\Delta(j; \delta, \theta, m) := \chi_\Delta(j; \delta, \theta, -m) := \frac{4\theta j + m\delta}{m\delta}\exp\left\{-\frac{4\theta j}{\Delta}(2\theta j + m\delta)\right\}. \tag{45}$$

Proof. Begin by noting that the strong Markov property enables us to decompose our required probability as follows:

$$\mathbb{P}(W_{[s,\tau]} \in [W_s - \theta, W_s + \theta] | W_s, W_q, W_\tau)$$
$$= \underbrace{\mathbb{P}(W_{[s,q]} \in [W_s - \theta, W_s + \theta] | W_s, W_q)}_{p_1}\,\underbrace{\mathbb{P}(W_{[q,\tau]} \in [W_s - \theta, W_s + \theta] | W_q, W_\tau)}_{p_2}. \tag{46}$$

Relating the decomposition to the statement of theorem 4, p_1 follows directly from the parameteriza-tion given and the representation in Pollock (2013), theorem 6.1.2, of the result in Beskos *et al.* (2008), proposition 3. p_2 similarly follows from the representation found in Pollock *et al.* (2016), theorem 5.

Corollary 1. Letting $p := \mathbb{P}(W_{[s,\tau]} \in [W_s - \theta, W_s + \theta])$, monotonically convergent upper and lower bounding probabilities ($p_n^{\uparrow}$ and $p_n^{\downarrow}$ respectively) with the property that $\lim_{n\to\infty} p_n^{\uparrow} \to p$ and $\lim_{n\to\infty} p_n^{\downarrow} \to p$ can be found (where $n_0 := \lceil\sqrt{(\tau - q + 4\theta^2)}/4\theta\rceil$),

$$p_n^{\downarrow} := \frac{1-\sum_{j=1}^{n}\varsigma_{q-s}(j;W_s-W_q,\theta)+\sum_{j=1}^{n-1}\varphi_{q-s}(j;W_s-W_q,\theta)}{1-\exp[-2\theta\{m(W_s-W_q)+\theta\}/(q-s)]} \times\left\{1+\sum_{j=1}^{n_0+n}\psi_{\tau-q}(j;W_q-W_\tau,\theta,m)+\sum_{j=1}^{n_0+n-1}\chi_{\tau-q}(j;W_q-W_\tau,\theta,m)\right\}, \tag{47}$$

$$p_n^{\uparrow} := \frac{1-\sum_{j=1}^{n}\varsigma_{q-s}(j;W_s-W_q,\theta)+\sum_{j=1}^{n}\varphi_{q-s}(j;W_s-W_q,\theta)}{1-\exp[-2\theta\{m(W_s-W_q)+\theta\}/(q-s)]} \times\left\{1+\sum_{j=1}^{n_0+n}\psi_{\tau-q}(j;W_q-W_\tau,\theta,m)+\sum_{j=1}^{n_0+n}\chi_{\tau-q}(j;W_q-W_\tau,\theta,m)\right\}. \tag{48}$$

Furthermore we have

$$\frac{p_n^{\uparrow}-p_n^{\downarrow}}{p_{n-1}^{\uparrow}-p_{n-1}^{\downarrow}} =: r_n \leqslant r \in (0,1), \tag{49}$$

and so

$$\bar{K} := \sum_{i=1}^{\infty}|p_i^{\uparrow}-p_i^{\downarrow}| = |p_1^{\uparrow}-p_1^{\downarrow}| + \sum_{i=2}^{\infty}\prod_{j=2}^{i} r_j \leqslant \sum_{i=0}^{\infty} r^i = \frac{1}{1-r} < \infty. \tag{50}$$

Proof. The summations in the first term of the sequences (47) and (48) follow from theorem 4 and Beskos *et al*. (2008), proposition 3. The summations in the second term of the sequences (47) and (48), and the necessary condition on n_0, follow from Pollock *et al*. (2016), corollary 5. The validity of the product form of sequences (47) and (48) follows from Pollock *et al*. (2016), corollary 1. The bound on the ratio of subsequent bound ranges of p in expression (49) follows from the exponential decay in n of $\varsigma(n)$, $\varphi(n)$, $\psi(n)$ and $\chi(n)$ of theorem 4, and, as shown in the proof of Pollock (2013), theorem 6.1.1, and Pollock (2013), corollary 6.1.3. Expression (50) follows directly from expression (49). □

Having established theorem 4 and corollary 1 we we can now construct a (retrospective) rejection sampler in which we simulate W_q (as per the law of a Bessel bridge) and, by means of an algorithmic loop in which the bounding sequences of the acceptance probability are evaluated to sufficient precision, we make the determination of acceptance or rejection. This is summarized in algorithm 5 in Table 6 further noting that, although the embedded loop is of random length, by corollary 1 we know that it halts in finite expected time ($\bar{K}$ can be interpreted as the expected computational cost of the nested loop, noting that $\mathbb{E}[\text{iterations}] := \Sigma_{i=0}^{\infty} i\,\mathbb{P}(\text{halt at step } i) = \Sigma_{i=0}^{\infty}\mathbb{P}(\text{halt at step } i \text{ or later}) = \bar{K}$).

C.3. Simulation of a single trajectory of constrained Brownian motion

We now have the constituent elements for Section 3, in which we simulate multivariate Brownian motion at any desired time marginal, with d-dimensional hypercubes inscribing intervals of the state space in which the sample path almost surely lies (layers, more formally defined in Pollock *et al*. (2016)). Recall from Section 3 that the killing times are determined by a random variable whose distribution depends on the inscribed layers, and so the presentation of algorithm 6 in Table 7 necessitates a loop in which the determination of whether the stopping time occurs in the interval is required.

We require the user-specified vector θ to determine the default hypercube inscription size. In practice, as with other MCMC methods, we might often apply a preconditioning matrix to the state space before applying the algorithm.

Further note that, because of the strong Markov property, it is user preference whether this algorithm is run in its entirety for every required time marginal, or whether it resets layer information whenever any one component breaches its boundary and reinitializes from that time on according to algorithm 6, step 4(b).

Appendix D: Path space rejection sampler for μ_T

A path space rejection sampler for μ_T can be constructed by drawing from Brownian motion measure, $\mathbf{X} \sim \mathbb{W}_T^{\mathbf{x}}$, accepting with probability $P(\mathbf{X})$ given by

Table 6. Algorithm 5: simulating $W_q \sim \mathbb{W}_{s,\tau}^{W_s,W_\tau}|(W_s, W_\tau, \theta)$, given $q \in [s,\tau]$, the end points (W_s and the extremal value W_τ), and constrained such that $\forall\, u \in [s,\tau], W_u \in [W_s - \theta, W_s + \theta]$

1 $\{b_q^{(i)}\}_{i=1}^3$: simulate
$$b_q^{(1)}, b_q^{(2)}, b_q^{(3)} \overset{\text{IID}}{\sim} N\left\{0, \frac{|\tau - q||q-s|}{(\tau-s)^2}\right\}$$
2 W_q: set
$$W_q := W_\tau + (-1)^{\mathbb{1}(W_\tau < W_s)} \sqrt{\left((\tau-s)\left[\left\{\frac{\theta(\tau-q)}{(\tau-s)^{3/2}} + b_q^{(1)}\right\}^2 + (b_q^{(2)})^2 + (b_q^{(3)})^2\right]\right)}$$
3 u: simulate $u \sim U[0,1]$ and set $n=1$
4 $p^{\downarrow}, p^{\uparrow}$: while $u \notin [p_n^{\downarrow}, p_n^{\uparrow}]$, set $n = n+1$
5 p: if $u \leqslant p_n^{\downarrow}$ accept; otherwise reject and return to step 1
6 Return (q, W_q)

Table 7. Algorithm 6: simulating constrained Brownian motion at a desired time marginal (t, W_t)

1 Input $\mathbf{W}_s$ and θ
2 τ: for $i \in \{1,\ldots,d\}$, simulate $(\tau^{(i)}, W_\tau^{(i)})$ as per algorithm 4
3 $\hat{\tau}$: set $\hat{\tau} := \inf_i\{\tau^{(i)}\}$, set $j := \{i \in \{1,\ldots,d\} : \tau^{(i)} = \hat{\tau}\}$; t: if required, simulate t as outlined in Section 3
4 t: if $t \notin [s, \hat{\tau}]$,

(a) $(\hat{\tau}, W_{\hat{\tau}}^{(\cdot)})$, for $i \in \{1,\ldots,d\} \setminus j$, simulate $(\hat{\tau}, W_{\hat{\tau}}^{(i)})$ as per algorithm 5;
(b) $(\tau^{(j)}, W_\tau^{(j)})$, simulate $(\tau^{(j)}, W_\tau^{(j)})$ as per algorithm 4;
(c) s, set $s := \hat{\tau}$, and return to step 3

5 $(t, W_t^{(\cdot)})$: for $i \in \{1,\ldots,d\}$, simulate $(t, W_t^{(i)})$ as per algorithm 5
6 Return (t, W_t)

$$P(\mathbf{X}) = \underbrace{\exp\left[\Phi T - \sum_{i=1}^{n_R} L_{\mathbf{X}}^{(i)}\{(\tau_i \wedge T) - \tau_{i-1}\}\right]}_{=:P^{(1)}(\mathbf{X}) \in [0,1]} \prod_{i=1}^{n_R} \underbrace{\exp\left[-\int_{\tau_{i-1}}^{\tau_i \wedge T} \{\phi(\mathbf{X}_s) - L_{\mathbf{X}}^{(i)}\}\, \mathrm{d}s\right]}_{=:P^{(2,i)}(\mathbf{X})} \tag{51}$$

$$= \prod_{i=1}^{n_R} \left(\underbrace{\exp[(\Phi - L_{\mathbf{X}}^{(i)})\{(\tau_i \wedge T) - \tau_{i-1}\}]}_{=:P^{(1,i)}(\mathbf{X}) \in [0,1]} \underbrace{\exp\left[-\int_{\tau_{i-1}}^{\tau_i \wedge T} \{\phi(\mathbf{X}_s) - L_{\mathbf{X}}^{(i)}\}\, \mathrm{d}s\right]}_{=:P^{(2,i)}(\mathbf{X})} \right). \tag{52}$$

The algorithmic pseudocode for this approach is thus presented in algorithm 7 in Table 8.

Crucially, determination of acceptance is made using only a path *skeleton* (as introduced in Pollock *et al.* (2016), a path skeleton is a finite dimensional realization of the sample path, including a *layer* constraining the sample path, sufficient to recover the sample path at any other finite collection of time points without error as desired). The path space rejection sampler for μ_T outputs the skeleton composed of all intermediate simulations:

$$\mathcal{S}_{\text{PsRS}}(\mathbf{X}) := \{\mathbf{X}_0, ((\xi_j^{(i)}, \mathbf{X}_{\xi_j^{(i)}})_{j=1}^{\kappa_i}, R_{\mathbf{X}}^{(i)})_{i=1}^{n_R}\}, \tag{53}$$

which is sufficient to simulate any finite dimensional subset of the remainder of the sample path (denoted by $\mathbf{X}^{\text{rem}}$) as desired without error (as outlined in Pollock *et al.* (2016), section 3.1 and Appendix C,

Table 8. Algorithm 7: path space rejection sampler for μ_T algorithm

1 Input: $\mathbf{X}_0$
2 R: simulate layer information $R \sim \mathcal{R}$ as per Appendix C
3 $P^{(1)}$: with probability $1 - \exp[\Phi T - \Sigma_{i=1}^{n_R} L_{\mathbf{X}}^{(i)}\{(\tau_i \wedge T) - \tau_{i-1}\}]$ reject and return to step 2
4 n_R: for i in $1 \to n_R$,
(a) $\mathbb{U}_R^{(i)}$, set $j=0$, $\kappa_i = 0$, $\xi_0^{(i)} := \tau_{i-1}$ and $E_1^{(i)} \sim \text{Exp}(U_{\mathbf{X}}^{(i)} - L_{\mathbf{X}}^{(i)})$, while $\Sigma_j E_j^{(i)} < (\tau_i \wedge T) - \tau_{i-1}$,
(i) $\xi_j^{(i)}$, set $j = j+1$ and $\xi_j^{(i)} = \xi_{j-1}^{(i)} + E_j^{(i)}$,
(ii) $\mathbf{X}_{\xi_j^{(i)}}$, simulate $\mathbf{X}_{\xi_j^{(i)}} \sim \text{MVN}\{\mathbf{X}_{\xi_{j-1}^{(i)}}, (\xi_j^{(i)} - \xi_{j-1}^{(i)})\} | R_{\mathbf{X}}^{(i)}$,
(iii) $P^{(2,i,j)}$, with probability $1 - \{U_{\mathbf{X}}^{(i)} - \phi(\mathbf{X}_{\xi_j^{(i)}})\}/(U_{\mathbf{X}}^{(i)} - L_{\mathbf{X}}^{(i)})$, reject path and return to step 2,
(iv) $E_{j+1}^{(i)}$, simulate $E_{j+1}^{(i)} \sim \text{Exp}(U_{\mathbf{X}}^{(i)} - L_{\mathbf{X}}^{(i)})$;
(b) $\mathbf{X}_{\tau_i \wedge T}$: simulate $\mathbf{X}_{\tau_i \wedge T} \sim \text{MVN}[\mathbf{X}_{\xi_j^{(i)}}, \{(\tau_i \wedge T) - \xi_j^{(i)}\}] | R_{\mathbf{X}}^{(i)}$

Table 9. Algorithm 8: killed Brownian motion algorithm

1 Initialize: set $i=1$, $j=0$, $\tau_0 = 0$; input initial value $\mathbf{X}_0$
2 R: simulate layer information $R_{\mathbf{X}}^{(i)} \sim \mathcal{R}$ as per Appendix C, obtaining τ_i, $U_{\mathbf{X}}^{(i)}$
3 E: simulate $E \sim \text{Exp}(U_{\mathbf{X}}^{(i)} - \Phi)$
4 ξ_j: set $j = j+1$ and $\xi_j = (\xi_{j-1} + E) \wedge \tau_i$
5 $\mathbf{X}_{\xi_j}$: simulate $\mathbf{X}_{\xi_j} \sim \text{MVN}\{\mathbf{X}_{\xi_{j-1}}, (\xi_j - \xi_{j-1})\} | R_{\mathbf{X}}^{(i)}$
6 τ_i: if $\xi_j = \tau_i$, set $i = i+1$ and return to step 2
7 P: with probability $\{U_{\mathbf{X}}^{(i)} - \phi(\mathbf{X}_{\xi_j})\}/(U_{\mathbf{X}}^{(i)} - \Phi)$ return to step 3
8 $(\bar{\tau}, \mathbf{X}_{\bar{\tau}})$: return $(\bar{\tau}, \mathbf{X}_{\bar{\tau}}) = (\xi_j, \mathbf{X}_{\xi_j})$, $i_{\bar{\tau}} = i$, $j_{\bar{\tau}} = j$

$$\mathbf{X}_{(0,T)}^{\text{rem}} \sim \otimes_{i=1}^{n_R} \left(\otimes_{j=1}^{\kappa_i} \mathbb{W}_{\xi_{j-1}^{(i)}, \xi_j^{(i)}}^{\mathbf{X}[\xi_{j-1}^{(i)}, \xi_j^{(i)}]} \right) \Bigg| R_{\mathbf{X}}^{(i)}. \tag{54}$$

Appendix E: Killed Brownian motion

In algorithm 4 we detailed an approach to simulate the killing time and location, $(\bar{\tau}, \mathbf{X}_{\bar{\tau}})$, for killed Brownian motion. To avoid unnecessary algorithmic complexity, note that we can recover the pair $(\bar{\tau}, \mathbf{X}_{\bar{\tau}})$ by a simple modification of algorithm 7 in which we set $\forall i L_{\mathbf{X}}^{(i)} := \Phi$ and return the first rejection time. This is presented in algorithm 8 in Table 9. A variant in which $L_{\mathbf{X}}^{(i)}$ is incorporated would achieve greater efficiency, but has been omitted for notational clarity.

As in the path space rejection sampler for μ_T that was presented in Appendix D, in killed Brownian motion (algorithm 8) we can recover in the interval $[0, \bar{\tau})$ the remainder of the sample path as desired without error as follows (where for clarity we have suppressed the full notation, but can be conducted as described in Appendix C):

$$\begin{aligned} \mathcal{S}_{\text{KBM}}(\mathbf{X}) &:= \{\mathbf{X}_0, (\xi_j, \mathbf{X}_{\xi_j})_{j=1}^{j_{\bar{\tau}}}, (R_{\mathbf{X}}^{(i)})_{i=1}^{i_{\bar{\tau}}}\}, \\ \mathbf{X}_{(0,T)}^{\text{rem}} &\sim \mathbb{W} | \mathcal{S}_{\text{KBM}}. \end{aligned} \tag{55}$$

Appendix F: Rejection-sampling-based quasi-stationary Monte Carlo algorithm

In Section 3.3 we considered the embedding of the importance sampling killed Brownian motion of algorithm 1 within an SMC algorithm. A similar embedding for the rejection sampling variant (killed

Table 10. Algorithm 9 (continuous time) rejection QSMC algorithm R-QSMC

1 *Initialization step* ($m=0$):

(a) input, starting value $\hat{\mathbf{x}}$, number of particles, N;
(b) $\mathbf{X}_0^{(\cdot)}$, for k $1,\ldots,N$ set $\mathbf{X}_{t_0}^{(1:N)}=\hat{\mathbf{x}}$ and $w_{t_0}^{(1:N)}=1/N$;
(c) $\bar{\tau}_1^{(\cdot)}$, for k $1,\ldots,N$, simulate $(\bar{\tau}_1^{(k)},\mathbf{X}_{\bar{\tau}_1}^{(k)})|(t_0^{(k)},\mathbf{X}_{t_0}^{(k)})$ as per algorithm 8

2 *Iterative update steps* ($m=m+1$):

(a) $\bar{\bar{\tau}}_m$, set $\bar{\bar{\tau}}_m:=\inf\{\{\bar{\tau}_{m(k)}^{(k)}\}_{k=1}^N\}$, $\bar{k}:=\{k:\bar{\bar{\tau}}_m=\bar{\tau}_{m(k)}^{(k)}\}$;
(b) K, simulate $K\sim U\{\{1,\ldots,n\}\setminus\bar{k}\}$;
(c) $\mathbf{X}_{\bar{\bar{\tau}}_m}^{(\cdot)}$, simulate $\mathbf{X}_{\bar{\bar{\tau}}_m}^{(\bar{k})}\sim\mathbb{W}|\mathcal{S}_{\mathrm{KBM}}^{(K)}$ as given by expression (55) and as per algorithm 5;
(d) $\bar{\bar{\tau}}_{m+1}$, simulate $(\bar{\tau}_{m(\bar{k})+1}^{(\bar{k})},\mathbf{X}_{\bar{\tau}_{m(\bar{k})+1}}^{(\bar{k})})|(\bar{\bar{\tau}}_m,\mathbf{X}_{\bar{\bar{\tau}}_m}^{(\bar{k})})$ as per algorithm 8

Brownian motion) of Algorithm 8 is considered here as the probability of the killed Brownian motion trajectory of algorithm 8 remaining alive becomes arbitrarily small as the diffusion time increases. As such, if one wanted to approximate the law of the process conditioned to remain alive until large T it would have prohibitive computational cost.

Considering the killed Brownian motion algorithm that was presented in Appendix E, in which we simulate trajectories of killed Brownian motion, the most natural embedding of this within an SMC framework is to assign each particle constant unnormalized weight while alive, and zero weight when killed. Resampling in this framework simply consists of sampling killed particles uniformly at random from the remaining alive particle set. The manner in which we have constructed algorithm 8 enables us to conduct this resampling in continuous time, and so we avoid the possibility of at any time having an alive particle set of size zero. We term this the (continuous time) rejection QSMC approach, R-QSMC, and present it in algorithm 9 in Table 10. In algorithm 9 we denote by $m(k)$ as a count of the number of killing events of particle trajectory k in the time elapsed until the mth iteration of the algorithm.

Iterating R-QSMC beyond some time t^* at which point we believe that we have obtained convergence and, halting at time $T>t^*$, we can approximate the law of the killed process by the weighted occupation measures of the trajectories (where $\forall\, t w_t^{(\cdot)}=1/N$):

$$\pi(\mathrm{d}\mathbf{x})\approx\hat{\pi}(\mathrm{d}\mathbf{x}):=\frac{1}{T-t^*}\int_{t^*}^{\mathrm{T}}\sum_{k=1}^{N}w_t^{(k)}\delta_{\mathbf{X}_t^{(k)}}(\mathrm{d}\mathbf{x})\mathrm{d}t. \tag{56}$$

In some instances the tractable nature of Brownian motion will admit an explicit representation of expression (56). If not, one can simply sample the trajectories exactly at equally spaced points to find an unbiased approximation of expression (56), by means detailed in Appendix C.2 and algorithm 4. In particular, if we let $t_0:=0<t_1<\ldots<t_m:=T$ such that $t_i-t_{i-1}:=T/m$, then we can approximate the law of the killed process as we did in expression (7), where $w_{t^*:T}^{(1:N)}=1/N$.

Appendix G: Rejection sampling scalable Langevin exact algorithm (R-ScaLE)

In Section 4 we noted that the survival probability of a proposal Brownian motion sample path was related to the estimator $P(\mathbf{X})$ of Appendix D and in section 4.2 where we developed a replacement estimator. The construction of control variates in Section 4.2 enables us to construct the replacement estimator such that it has good scalability properties. In a similar fashion to the embedding of this estimator within a QSMC algorithm (algorithm 2) resulting in ScaLE (algorithm 3), we can embed this estimator with the rejection sampling variant R-QSMC (algorithm 9) resulting in the *rejection scalable Langevin exact algorithm R-ScaLE* which we present in algorithm 10 in Table 11.

As presented in algorithm 10 we may also be concerned with the absolute growth of $\tilde{\Phi}$ (relative to Φ) as a function of n to study its computational complexity. Note, however, as remarked on in Appendix E, if this growth is not favourable one can modify algorithm 8 to incorporate the additional path space bound $\tilde{L}_{\mathbf{X}}^{(i)}$ for each layer. Details of this modification have been omitted for notational clarity.

Table 11. Algorithm 10: R-ScaLE (as per algorithm 9 unless stated otherwise)

Initialize: choose $\hat{\mathbf{x}}$ and compute $\nabla \log\{\pi(\hat{\mathbf{x}})\}$, $\Delta \log\{\pi(\hat{\mathbf{x}})\}$, $\tilde{\Phi}$
1(c) On calling algorithm 8:

(i) replace Φ with $\tilde{\Phi}$;
(ii) replace $U_{\mathbf{X}}^{(i)}$ in step 2 with $\tilde{U}_{\mathbf{X}}^{(i)}$;
(iii) replace step 7 with simulate $I, J \sim^{\text{IID}} U\{0,\ldots,n\}$, and with probability $\{\tilde{U}_{\mathbf{X}}^{(i)} - \tilde{\phi}(\mathbf{X}_{\xi_j})\}/(\tilde{U}_{\mathbf{X}}^{(i)} - \Phi)$ return to step 3

2(d) As for step 1(c)

Appendix H: Discrete time sequential Monte Carlo construction

Consider the discrete time system with state space $E_k = (C(h(k-1), hk], \mathcal{Z}_k)$ at discrete time k, with the process denoted $\mathfrak{X}_k = (X_{(h(k-1),hk]}, \mathfrak{Z}_k)$ in which the auxiliary variables $\mathfrak{Z}_k$ take values in some space $\mathcal{Z}_k$.

ScaLE, with resampling conducted deterministically at times $h, 2h, \ldots$, coincides exactly with the mean field particle approximation of a discrete time Feynman–Kac flow, in the sense and notation of Del Moral (2004), with transition kernel

$$M_k(\mathfrak{X}_{k-1}, d\mathfrak{X}_k) = \mathbb{W}_{h(k-1),hk}^{X_{h(k-1)}}(\mathrm{d}X_{(h(k-1),hk]})\, Q_k(X_{(h(k-1),hk]}, d\mathfrak{Z}_k)$$

and a potential function $G_k(\mathfrak{X}_k)$, which is left intentionally unspecified to allow a broad range of variants of the algorithm to be included: the property which it must have to lead to a valid form of ScaLE is specified below. Allow

$$\bar{\mathbb{W}}_{0,hk}^{x}(\mathfrak{X}_{1:k}) = \mathbb{W}_{x}^{0,hk}(\mathrm{d}X_{0:hk}) \prod_{i=1}^{k} Q_i(X_{(h(i-1),hi]}, d\mathfrak{Z}_i)$$

and specify an extended version of the killed process via

$$\frac{\mathrm{d}\bar{\mathbb{K}}_{0,hk}^{x}}{\mathrm{d}\bar{\mathbb{W}}_{0,hk}^{x}}(\mathfrak{X}_{1:k}) \propto \prod_{i=1}^{k} G(\mathfrak{X}_i).$$

The validity of such a ScaLE algorithm depends on the following identity holding:

$$\frac{\mathrm{d}\mathbb{K}_{0,hk}^{x}}{\mathrm{d}\mathbb{W}_{0,hk}^{x}}(X_{0:hk}) \propto \mathbb{E}_{\mathbb{W}_{0,hk}^{x}}\left[\prod_{i=1}^{k} G_i(\mathfrak{X}_i) \middle| X_{0:hk}\right].$$

It is convenient to define some simplifying notation. We define the law of a discrete time process (in which is embedded a continuous time process taking values in $C[0,\infty)$):

$$\bar{\mathbb{W}}^{x}(\mathrm{d}\mathfrak{X}) = \bar{\mathbb{W}}_{0,h}^{x}(\mathrm{d}\mathfrak{X}_1) \prod_{k=1}^{\infty} \bar{\mathbb{W}}_{h(k-1),hk}^{X_{h(k-1)}}(\mathrm{d}\mathfrak{X}_k)$$

and of a family of processes indexed by k, $\bar{\mathbb{K}}_k^x$, again incorporating a continuous time process taking values in $C[0,\infty)$, via

$$\frac{\mathrm{d}\bar{\mathbb{K}}_{k}^{x}}{\mathrm{d}\bar{\mathbb{W}}_{x}}(\mathfrak{X}) \propto \prod_{i=1}^{k} G_i(\mathfrak{X}_i).$$

With a slight abuse of notation, we use the same symbol to refer to the associated finite dimensional distributions, with the intended distribution being indicated by the argument. We also define the *marginal* laws $\mathbb{W}^x$ and $\mathbb{K}_k^x$ via

$$\mathbb{W}^{x}(\mathrm{d}X) = \bar{\mathbb{W}}^{x}\{\mathrm{d}X \times (\otimes_{p=1}^{\infty} \mathcal{Z}_p)\},$$
$$\mathbb{K}_{k}^{x}(\mathrm{d}X) = \bar{\mathbb{K}}_{k}^{x}\{\mathrm{d}X \times (\otimes_{p=1}^{\infty} \mathcal{Z}_p)\}.$$

Proposition 3. Under mild regularity conditions (see Del Moral (2004) and Chopin (2004)), for any $\varphi : \mathbb{R}^d \to \mathbb{R}$, any algorithm within the framework described admits a central limit in that

$$\lim_{N\to\infty} \sqrt{N}\left[\frac{1}{N}\sum_{i=1}^{N}\varphi(X_{hk}^i) - \mathbb{K}_k^x\{\varphi(X_{hk}^i)\}\right] \Rightarrow \sigma_{k,G}(\varphi)Z$$

where Z is a standard normal random variable, '$\Rightarrow$' denotes convergence in distribution and

$$\begin{aligned}
\sigma_k^2(\varphi) = \mathbb{E}_{\bar{\mathbb{W}}}\Bigg[\Bigg[\frac{G_1(\mathfrak{X}_1)\mathbb{E}_{\bar{\mathbb{W}}^x}\left[\prod_{i=2}^{k} G(\mathfrak{X}_i)|\mathfrak{X}_1\right]}{\bar{\mathbb{W}}^x\left\{\prod_{i=1}^{k} G(\mathfrak{X}_i)\right\}}\Bigg]^2 \mathbb{E}_{\mathbb{K}_k^x}[[\varphi(X_{hk}) - \mathbb{K}_k^x\{\varphi(X_{hk})\}]^2|X_h]\Bigg] \\
+ \sum_{p=2}^{k-1}\mathbb{E}_{\bar{\mathbb{K}}_{p-1}^x}\Bigg[\Bigg[\frac{\bar{\mathbb{W}}^x\left\{\prod_{i=0}^{p-1} G(\mathfrak{X}_i)\right\}}{\bar{\mathbb{W}}^x\left\{\prod_{i=0}^{k} G(\mathfrak{X}_i)\right\}} G(\mathfrak{X}_p)\mathbb{E}_{\mathbb{W}^x}\left[\prod_{i=p+1}^{k} G(\mathfrak{X}_i)\middle| X_{hp}\right]\Bigg]^2 \\
\times \mathbb{E}_{\mathbb{K}_k^x}[[\varphi(X_{hk}) - \mathbb{K}_k^x\{\varphi(X_k)\}]^2|X_{hp}]\Bigg] \\
+ \mathbb{E}_{\bar{\mathbb{K}}_{k-1}^x}\Bigg[\Bigg[\frac{\bar{\mathbb{W}}^x\left\{\prod_{i=0}^{k-1} G(\mathfrak{X}_i)\right\}}{\bar{\mathbb{W}}^x\left\{\prod_{i=0}^{k} G(\mathfrak{X}_i)\right\}} G(\mathfrak{X}_k)\Bigg]^2 [\varphi(X_{hk}) - \bar{\mathbb{K}}_k^x\{\varphi(X_{hk})\}]^2\Bigg].
\end{aligned}$$

H.1. Proof outline

It follows by a direct application of the argument underlying the proposition of Johansen and Doucet (2008) (which itself follows from simple but lengthy algebraic manipulations from the results of Del Moral (2004) and Chopin (2004)) that, for any test function $\varphi : \mathbb{R}^d \to \mathbb{R}$ satisfying mild regularity conditions (see Del Moral (2004) and Chopin (2004)),

$$\lim_{N\to\infty} \sqrt{N}\left[\frac{1}{N}\sum_{i=1}^{N}\varphi(X_{hk}^i) - \mathbb{K}_k^x\{\varphi(X_{hk}^i)\}\right] \Rightarrow \sigma_{k,G}(\varphi)Z$$

where, Z is a standard normal random variable, '$\Rightarrow$' denotes convergence in distribution and

$$\begin{aligned}
\sigma_{k,G}^2(\varphi) = \mathbb{E}_{\bar{\mathbb{W}}}\left[\left\{\frac{\mathrm{d}\bar{\mathbb{K}}_k^x}{\mathrm{d}\bar{\mathbb{W}}^x}(X_{(0,h]}, \mathfrak{Z}_1)\right\}^2 \mathbb{E}_{\bar{\mathbb{K}}_k^x}[[\varphi(X_{hk}) - \bar{\mathbb{K}}_k^x\{\varphi(X_{hk})\}]^2|\bar{\mathcal{F}}_1]\right] \\
+ \sum_{p=2}^{k-1}\mathbb{E}_{\bar{\mathbb{K}}_{p-1}}\left[\left\{\frac{\mathrm{d}\bar{\mathbb{K}}_k^x}{\mathrm{d}\bar{\mathbb{K}}_{p-1}^x}(X_{(0,hp]}, \mathfrak{Z}_{1:p})\right\}^2 \mathbb{E}_{\bar{\mathbb{K}}_k^x}[[\varphi(X_{hk}) - \bar{\mathbb{K}}_k^x\{\varphi(X_k)\}]^2|\bar{\mathcal{F}}_p]\right] \\
+ \mathbb{E}_{\bar{\mathbb{K}}_{k-1}}\left[\left\{\frac{\mathrm{d}\bar{\mathbb{K}}_k^x}{\mathrm{d}\bar{\mathbb{K}}_{k-1}}(X_{(0,hk]}, \mathfrak{Z}_{1:k})\right\}^2 [\varphi(X_{hk}) - \bar{\mathbb{K}}_k^x\{\varphi(X_{hk})\}]^2\right]
\end{aligned}$$

with $\{\bar{\mathcal{F}}_p\}_{p\geqslant 0}$ being the natural filtration that is associated with $\bar{\mathbb{W}}^x$.

This can be straightforwardly simplified to

$$\sigma_k^2(\varphi) = \mathbb{E}_{\bar{\mathbb{W}}}\left[\left[\frac{G_1(\mathfrak{X}_1)\mathbb{E}_{\bar{\mathbb{W}}^x}\left[\prod_{i=2}^{k} G(\mathfrak{X}_i)|\mathfrak{X}_1\right]}{\bar{\mathbb{W}}^x\left\{\prod_{i=1}^{k} G(\mathfrak{X}_i)\right\}}\right]^2 \mathbb{E}_{\mathbb{K}_k^x}[[\varphi(X_{hk}) - \mathbb{K}_k^x\{\varphi(X_{hk})\}]^2|X_h]\right]$$

$$+\sum_{p=2}^{k-1}\mathbb{E}_{\bar{\mathbb{K}}_{p-1}^x}\left[\left[\frac{\bar{\mathbb{W}}^x\left\{\prod_{i=0}^{p-1} G(\mathfrak{X}_i)\right\}}{\bar{\mathbb{W}}^x\left\{\prod_{i=0}^{k} G(\mathfrak{X}_i)\right\}} G(\mathfrak{X}_p)\mathbb{E}_{\mathbb{W}^x}\left[\prod_{i=p+1}^{k} G(\mathfrak{X}_i)\middle| X_{hp}\right]\right]^2\right.$$

$$\left.\times\, \mathbb{E}_{\mathbb{K}_k^x}[[\varphi(X_{hk}) - \mathbb{K}_k^x\{\varphi(X_k)\}]^2|X_{hp}]\right]$$

$$+\mathbb{E}_{\bar{\mathbb{K}}_{k-1}^x}\left[\left[\frac{\bar{\mathbb{W}}^x\left\{\prod_{i=0}^{k-1} G(\mathfrak{X}_i)\right\}}{\bar{\mathbb{W}}^x\left\{\prod_{i=0}^{k} G(\mathfrak{X}_i)\right\}} G(\mathfrak{X}_k)\right]^2 [\varphi(X_{hk}) - \bar{\mathbb{K}}_k^x\{\varphi(X_{hk})\}]^2\right].$$

We conclude with the following corollary, showing that the particular combination of subsampling scheme and path space sampler fits into this framework and providing its particular asymptotic variance expression.

Corollary 2. Such a central limit theorem is satisfied in particular

(a) if no subsampling is used and one evaluates the *exact* (intractable) killing rate (as described in algorithm 2), and
(b) if subsampling is employed within the construct of the layered path space rejection sampler (as described in algorithm 3).

Proof. Both claims follow directly by the above argument with the appropriate identifications:1pt

(a) is established by setting

$$\begin{aligned}\mathcal{Z}_k &= \emptyset,\\ G_k(\mathfrak{X}_k) &= G(X_{[h(k-1),hk]})\\ &\propto \frac{\mathrm{d}\mathbb{K}_{h(k-1),hk}^{X_{h(k-1)}}}{\mathrm{d}\mathbb{W}_{h(k-1),hk}^{X_{h(k-1)}}}(X_{(h(k-1):hk]});\end{aligned}$$

(b) is established by setting (where we denote by c the number of pairs of data points employed by the subsampling mechanism; $c = 1$ for the examples in this paper)

$$\mathcal{Z}_k = \bigcup_{m_k=1}^{\infty}\bigotimes_{p=1}^{m_k} R(\tau_{k,p-1}, \tau_{k,p}),$$

$$R(s,t) = \bigcup_{\kappa=0}^{\infty}\{\kappa\}\times(s,t]^{\kappa}\times\{1,\ldots,n\}^{2c\kappa},$$

$$\mathfrak{Z}_k = (r_{k,1},\ldots,r_{k,m_k}),$$

$$r_{k,p} = (\kappa_{k,p}, \xi_{k,p,1},\ldots,\xi_{k,p,\kappa_{k,p}}, s_{k,j,1,1:2c},\ldots,s_{k,p,\kappa_{k,p},1:2c}),$$

$$G_k(\mathfrak{X}_k) = \exp\left\{-\sum_{p=1}^{m_k} L_\theta(X_{\tau_{k,p-1}})(\tau_{k,p}-\tau_{k,p-1})\right\}\times\prod_{p=1}^{m_k}\prod_{j=1}^{\kappa_{k,p}}\left\{\frac{U_\theta(X_{\tau_{k,p-1}}) - \tilde{\phi}(X_{\xi_{k,p,j}}, s_{k,p,j,1:2c})}{U_\theta(X_{\tau_{k,p-1}}) - L_\theta(X_{\tau_{k,p-1}})}\right\},$$

$$Q_k(X_{(h(k-1),hk]}, \mathrm{d}\mathfrak{Z}_k) = \prod_{p=1}^{m_k} \left[\mathrm{PP}(\mathrm{d}\xi_{k,p,1:\kappa_{k,p}}; \{U_\theta(X_{\tau_{k,p-1}}) - L_\theta(X_{\tau_{k,p-1}})\}, [\tau_{k,p-1}, \tau_{k,p}]) \right.$$
$$\left. \times \prod_{j=1}^{\kappa_{k,p}} \frac{1}{n^{2c}} \prod_{l=1}^{2c} \delta_{\{1,\ldots,n\}}(\mathrm{d}s_{k,j,l}) \right]$$

where $\mathrm{PP}(\cdot; \lambda, [a,b])$ denotes the law of a homogeneous Poisson process of rate λ over interval $[a,b]$, $\delta_{\{1,\ldots,n\}}$ denotes the counting measure over the first n natural numbers and a number of variables which correspond to deterministic transformations of the X-process have been defined to lighten the notation:

$$\tau_{k,p} = \begin{cases} (k-1)h & p = 0, \\ \inf\{t : |X_t - X_{\tau_{k,p-1}}| \geqslant \theta\} & p = 1, \ldots, m_k - 1, \\ kh & p = m_k \end{cases}$$

and m_k is the number of distinct layer pairs that are employed in interval k of the discrete time embedding of the algorithm (i.e. it is the number of first-passage times simulated within the continuous time algorithm after time $(k-1)h$ until one of them exceeds kh, as detailed in Appendices C.1 and C.2).

Appendix I: Estimation of effective sample size

Assume that the QSMC algorithm (or ScaLE) has been run for an execution (diffusion) time of length T, and that the weighted particle set (of size N) is to be used at the following auxiliary mesh times $t^*, \ldots, t_m := T$ (recalling from Section 3.3 that $t^* \in (t_0, \ldots, t_m)$ is a user-selected quasi-stationary burn-in time) for computation of the Monte Carlo estimators (7) and (8).

The posterior mean for the parameters at time $t_i \in [t^*, T]$ is simply estimated by using the particle set by $\hat{\mathbf{X}}_{t_i} = \Sigma_{k=1}^N w_{t_i}^{(k)} \mathbf{X}_{t_i}^{(k)}$. An overall estimate of the posterior mean and variance can be computed as follows:

$$\bar{\mathbf{X}} = \frac{1}{m(T-t^*)/T} \sum_{i=m(T-t^*)/T}^{m} \hat{\mathbf{X}}_{t_i}, \tag{57}$$

$$\hat{\sigma}^2_{\mathbf{X}} = \frac{1}{m(T-t^*)/T} \sum_{i=m(T-t^*)/T}^{m} \sum_{k=1}^{N} w_{t_i}^{(k)} (\mathbf{X}_{t_i}^{(k)} - \bar{\mathbf{X}})^2. \tag{58}$$

The marginal ESS for particles at a single time point can be estimated as the ratio of the variance of $\hat{\mathbf{X}}_t$ to the estimate of the posterior variance:

$$\mathrm{ESS}_M = \hat{\sigma}^2_{\mathbf{X}} \left\{ \frac{1}{m(T-t^*)/T} \sum_{t=m(T-t^*)/T}^{m} (\hat{\mathbf{X}}_{t_i} - \bar{\mathbf{X}})^2 \right\}^{-1}. \tag{59}$$

Although in total we have $m(T-t^*)/T$ sets of particles (after burn-in), these will be correlated. This is accounted for by using the lag 1 auto-correlation of $\hat{\mathbf{X}}_{t^*}, \ldots, \hat{\mathbf{X}}_T$, which we denote $\hat{\rho}$. Our overall estimated ESS is

$$\mathrm{ESS} := \frac{m(T-t^*)}{T} \frac{1-\hat{\rho}}{1+\hat{\rho}} \mathrm{ESS}_M. \tag{60}$$

References

Ahn, S., Balan, A. K. and Welling, M. (2012) Bayesian posterior sampling via stochastic gradient Fisher scoring. In *Proc. 29th Int. Conf. Machine Learning*, pp. 1771–1778.

Andrieu, C. and Roberts, G. O. (2009) The pseudo-marginal approach for efficient Monte Carlo computations. *Ann. Statist.*, **37**, 697–725.

Asmussen, S., Glynn, P. and Pitman, J. (1995) Discretization error in simulation of one-dimensional reflecting Brownian motion. *Ann. Appl. Probab.*, **5**, 875–896.

Baker, J., Fearnhead, P., Fox, E. B. and Nemeth, C. (2019) Control variates for stochastic gradient MCMC. *Statist. Comput.*, **29**, 599–615.

Bardenet, R., Doucet, A. and Holmes, C. C. (2014) Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *Proc. 31st Int. Conf. Machine Learning*, pp. 405–413.

Bardenet, R., Doucet, A. and Holmes, C. C. (2017) On Markov chain Monte Carlo methods for tall data. *J. Mach. Learn. Res.*, **18**, 1–43.

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

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

Beskos, A. and Roberts, G. O. (2005) An exact simulation of diffusions. *Ann. Appl. Probab.*, **15**, 2422–2444.

Bierkens, J., Fearnhead, P. and Roberts, G. O. (2019) The zig-zag process and super-efficient sampling for Bayesian analysis of big data. *Ann. Statist.*, **47**, 1288–1320.

Blanchet, J., Glynn, P. and Zheng, S. (2016) Analysis of a stochastic approximation algorithm for computing quasi-stationary distributions. *Adv. Appl. Probab.*, **48**, 792–811.

Bottou, L. (2010) Large-scale machine learning with stochastic gradient descent. In *Proc. COMPSTAT'2010*, pp. 177–186. New York: Springer.

Brosse, N., Durmus, A. and Moulines, E. (2018) The promises and pitfalls of stochastic gradient Langevin dynamics. In *Adv. Neurl Inform. Process Systems*, pp. 8268–8278.

Burq, Z. A. and Jones, O. (2008) Simulation of Brownian motion at first passage times. *Math. Comput. Simuln*, **77**, 64–71.

Carpenter, J., Clifford, P. and Fearnhead, P. (1999) Improved particle filter for nonlinear problems. *IEE Proc. Radar Sonar Navign* **146**, 2–7.

Chen, C., Ding, N. and Carin, L. (2015) On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*, pp. 2278–2286.

Chen, T., Fox, E. B. and Guestrin, C. (2014) Stochastic gradient Hamiltonian Monte Carlo. In *Proc. 31st Int. Conf. Machine Learning*, pp. 1683–1691.

Chopin, N. (2004) Central limit theorem for sequential Monte Carlo methods and its applications to Bayesian inference. *Ann. Statist.*, **32**, 2385–2411.

Ciesielski, Z. and Taylor, S. J. (1962) First passage times and sojourn times for Brownian motion in space and the exact Hausdorff measure of the sample path. *Ann. Math. Statist.*, **103**, 1434–1450.

Collet, P., Martinez, S. and San Martin, S. (2013) *Quasi-Stationary Distributions: Markov Chains, Diffusions and Dynamical Systems*. Berlin: Springer.

Dalalyan, A. S. and Karagulyan, A. G. (2019) User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient. *Stoch. Processes Appl.*, **129**, 5278–5311.

Del Moral, P. (2004) *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*, 1st edn. New York: Springer.

Del Moral, P., Doucet, A. and Jasra, A. (2012) On adaptive resampling procedures for sequential Monte Carlo methods. *Bernoulli*, **18**, 252–278.

Del Moral, P. and Miclo, L. (2000) Branching and interacting particle systems and approximations of Feynman-Kac formulæ with applications to non-linear filtering. In *Séminaire de Probabilités XXXIV* (eds J. Azéma, M. Emery, M. Ledoux and M. Yor) pp. 1–145. Berlin: Springer.

Del Moral, P. and Miclo, L. (2003) Particle approximations of Lyapunov exponents connected to Schrödinger operators and Feynman–Kac semigroups. *ESAIM Probab. Statist.*, **7**, 171–208.

Del Moral, P., Moulines, E., Olsson, J. and Vergé, C. (2016) Convergence properties of weighted particle islands with application to the double bootstrap algorithm. *Stoch. Sys.*, **6**, 367–419.

Devroye, L. (1986) *Non-uniform Random Variate Generation*, 1st edn. New York: Springer.

Devroye, L. (2009) On exact simulation algorithms for some distributions related to Jacobi theta functions. *Statist. Probab. Lett.*, **79**, 2251–2259.

Dubey, K. A., Reddi, S. J., Williamson, S. A., Poczos, B., Smola, A. J. and Xing, E. P. (2016) Variance reduction in stochastic gradient Langevin dynamics. In *Advances in Neural Information Processing Systems*, pp. 1154–1162.

Fort, G. and Roberts, G. O. (2005) Subgeometric ergodicity of strong Markov processes. *Ann. Appl. Probab.*, **15**, 1565–1589.

Giardina, C., Kurchan, J., Lecomte, V. and Tailleur, J. (2011) Simulating rare events in dynamical processes. *J. Statist. Phy.*, **145**, 787–811.

Groisman, P. and Jonckheere, M. (2013) Simulation of quasi-stationary distributions on countable spaces. *Markov Process. Reltd Flds*, **19**, 521–542.

Huggins, J. H. and Zou, J. (2017) Quantifying the accuracy of approximate diffusions and Markov chains. In *Proc. 19th Int. Conf. Artificial Intelligence and Statistics*, pp. 382–391.

Jacob, P. E. and Thiéry, A. H. (2015) On non-negative unbiased estimators. *Ann. Statist.*, **43**, 769–784.

Jin, C., Netrapalli, P. and Jordan, M. I. (2018) Accelerated gradient descent escapes saddle points faster than gradient descent. *Proc. Mach. Learn. Res.*, **75**, 1042–1085.

Johansen, A. M. and Doucet, A. (2008) A note on the auxiliary particle filter. *Statist. Probab. Lett.*, **78**, 1498–1504.

Johnson, R. A. (1970) Asymptotic expansions associated with posterior distributions. *Ann. Math. Statist.*, **41**, 851–864.

Jordan, M. I. (2013) On statistics, computation and scalability. *Bernoulli*, **19**, 1378–1390.

Karatzas, I. and Shreve, S. (1991) *Brownian Motion and Stochastic Calculus*, 2nd edn. New York: Springer.

Kingman, J. F. C. (1992) *Poisson Processes*, 1st edn. Oxford: Clarendon.
Kong, A., Liu, J. S. and Wong, W. H. (1994) Sequential imputations and Bayesian missing data problems. *J. Am. Statist. Ass.*, **89**, 278–288.
Korattikara, A., Chen, Y. and Welling, M. (2014) Austerity in MCMC land: cutting the Metropolis-Hastings budget. In *Proc. 31st Int. Conf. Machine Learning*, pp. 181–189.
Li, C., Srivastava, S. and Dunson, D. B. (2017) Simple, scalable and accurate posterior interval estimation. *Biometrika*, **104**, 665–680.
Ma, Y., Chen, T. and Fox, E. (2015) A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, pp. 2917–2925.
Maclaurin, D. and Adams, R. P. (2015) Firefly Monte Carlo: exact MCMC with subsets of data. In *Proc. 24th Int. Jt Conf. Artificial Intelligence*, pp. 4289–4295.
Martin, A. D., Quinn, K. M. and Park, J. H. (2011) MCMCpack: Markov Chain Monte Carlo in R. *J. Statist. Softwr.*, **42**, 22.
Minsker, S., Srivastava, S., Lin, L. and Dunson, D. B. (2014) Scalable and robust Bayesian inference via the median posterior. In *Proc. 31st Int. Conf. Machine Learning*, pp. 1656–1664.
Nagapetyan, T., Duncan, A. B., Hasenclever, L., Vollmer, S. J., Szpruch, L. and Zygalakis, K. (2017) The true cost of stochastic gradient Langevin dynamics. *Preprint arXiv:1706.02692*.
Neiswanger, W., Wang, C. and Xing, E. (2014) Asymptotically exact, embarrassingly parallel MCMC. In *Proc. 30th Conf. Uncertainty in Artificial Intelligence*, pp. 623–632.
Nesterov, Y. (2013) *Introductory Lectures on Convex Optimization: a Basic Course*, vol. **87**. New York: Springer Science and Business Media.
Nicholls, G. K., Fox, C. and Watt, A. M. (2012) Coupled MCMC with a randomized acceptance probability. *Preprint arXiv:1205.6857*.
de Oliveira, M. M. and Dickman, R. (2005) How to simulate the quasistationary state. *Phys. Rev.* E, **71**, 61–129.
Pollock, M. (2013) Some Monte Carlo methods for jump diffusions. *PhD Thesis*. Department of Statistics, University of Warwick, Coventry.
Pollock, M. (2015) On the exact simulation of (jump) diffusion bridges. In *Proc. Winter Simulation Conf.*, pp. 348–359.
Pollock, M., Johansen, A. M. and Roberts, G. O. (2016) On the exact and ϵ-strong simulation of (jump) diffusions. *Bernoulli*, **22**, 794–856.
Quiroz, M., Minh-Ngoc, T., Villani, M. and Kohn, R. (2016) Exact subsampling MCMC. *Preprint arXiv: 1603.08232*.
Revuz, D. and Yor, M. (2013) *Continuous Martingales and Brownian Motion*. New York: Springer Science and Business Media.
Robert, C. and Casella, G. (2004) *Monte Carlo Statistical Methods*, 2nd edn. New York: Springer.
Rousset, M. (2006) On the control of an interacting particle estimation of Schrödinger ground states. *SIAM J. Math. Anal.*, **38**, 824–844.
Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I. and McCulloch, R. E. (2016) Bayes and big data: the consensus Monte Carlo algorithm. *Int. J. Mangment Sci. Engng Mangmnt*, **11**, 78–88.
Srivastava, S., Cevher, V., Tan-Dinh, Q. and Dunson, D. B. (2016) WASP: scalable Bayes via barycenters of subset posteriors. In *Proc. 18th Int. Conf. Artificial Intelligence and Statistics*, pp. 912–920.
Steinsaltz, D. and Evans, S. (2007) Quasistationary distributions for one-dimensional diffusions with killing. *Trans. Am. Math. Soc.*, **359**, 1285–1324.
Tanner, M. A. and Wong, W. H. (1987) The calculation of posterior distributions by data augmentation. *J. Am. Statist. Ass.*, **82**, 528–540.
Teh, Y. W., Thiéry, A. H. and Vollmer, S. J. (2016) Consistency and fluctuations for stochastic gradient Langevin dynamics. *J. Mach. Learn. Res.*, **17**, 193–225.
Vollmer, S. J., Zygalakis, K. C. and Teh, Y. W. (2016) Exploration of the (non-)asymptotic bias and variance of stochastic gradient Langevin dynamics. *J. Mach. Learn. Res.*, **17**, 1–48.
Wang, X. and Dunson, D. B. (2013) Parallelizing MCMC via Weierstrass sampler. *Preprint arXiv:1312.4605*.
Welling, M. and Teh, Y. W. (2011) Bayesian learning via stochastic gradient Langevin dynamics. In *Proc. 28th Int. Conf. Machine Learning*, pp. 681–688.
Whiteley, N. and Kantas, N. (2017) Calculating principal eigen-functions of non-negative integral kernels: particle approximations and applications. *Math. Oper. Res.*, **42**, 1007–1034.