Global Consensus Monte Carlo

Lewis J. Rendell*

Department of Statistics, University of Warwick

and

Adam M. Johansen

Department of Statistics, University of Warwick

and

Anthony Lee

School of Mathematics, University of Bristol

and

Nick Whiteley

School of Mathematics, University of Bristol

August 11, 2020

Abstract

To conduct Bayesian inference with large data sets, it is often convenient or necessary to distribute the data across multiple machines. We consider a likelihood function expressed as a product of terms, each associated with a subset of the data. Inspired by global variable consensus optimisation, we introduce an instrumental hierarchical model associating auxiliary statistical parameters with each term, which are conditionally independent given the top-level parameters. One of these top-level parameters controls the unconditional strength of association between the auxiliary parameters. This model leads to a distributed MCMC algorithm on an extended state space yielding approximations of posterior expectations. A trade-off between computational tractability and fidelity to the original model can be controlled by changing the association strength in the instrumental model. We further propose the use of a SMC sampler with a sequence of association strengths, allowing both the automatic determination of

appropriate strengths and for a bias correction technique to be applied. In contrast to similar distributed Monte Carlo algorithms, this approach requires few distributional assumptions. The performance of the algorithms is illustrated with a number of simulated examples.

1   Introduction

Large data sets arising in modern statistical applications present serious challenges for standard computational techniques for Bayesian inference, such as Markov chain Monte Carlo (MCMC) and other approaches requiring repeated evaluations of the likelihood function. We consider here the situation where the data are distributed across multiple computing nodes. This could be because the likelihood function cannot be computed on a single computing node in a reasonable amount of time, e.g. the data might not fit into main memory.

We assume that the likelihood function can be expressed as a product of terms so that the posterior density for the statistical parameter Z satisfies

$$\pi(z) \propto \mu(z) \prod_{j=1}^{b} f_j(z), \tag{1}$$

where Z takes values $z \in \mathsf{E} \subseteq \mathbb{R}^d$, and μ is a prior density. We assume that f_j is computable on computing node j and involves consideration of $\mathbf{y}_j$, the jth subset or 'block' of the full data set, which comprises b such blocks.

Many authors have considered 'embarrassingly parallel' MCMC algorithms approximating expectations with respect to (1), following the Consensus Monte Carlo approach of Scott et al. (2016). Such procedures require separate MCMC chains to be run on each computing node, each targeting a distribution dependent only on the associated likelihood contribution f_j. The samples from each of these chains are then combined in a final post-processing step to generate an approximation of the true posterior π. Such algorithms require communication between the nodes only at the very beginning and end of the procedure, falling into the MapReduce framework (Dean and Ghemawat, 2008); their use is therefore more advantageous when inter-node communication is costly, for example due to high latency. However, the effectiveness of such approaches at approximating the true posterior π depends heavily on the final combination step. Many proposed approaches are based on assumptions on the likelihood contributions f_j, or employ techniques that may be infeasible in high-dimensional settings. We later

review some of these approaches, and some issues surrounding their use, in Section 2.4.

Instead of aiming to minimise entirely communication between nodes, we propose an approach that avoids employing a final aggregation step, thereby avoiding distributional assumptions on π. This approach is motivated by global variable consensus optimisation (see, e.g., Boyd et al., 2011, Section 7; we give a summary in Section 2.1). Specifically we introduce an instrumental hierarchical model, associating an auxiliary parameter with each likelihood contribution (and therefore each computing node), which are conditionally independent given Z. An additional top-level parameter controlling their unconditional strength of association is also introduced. This allows the construction of an MCMC algorithm on an extended state space, yielding estimates of expectations with respect to π. By tuning the association strength through the top-level parameter, a trade-off between computational tractability and fidelity to the original model can be controlled.

As well as avoiding issues associated with embarrassingly parallel algorithms, our framework presents benefits compared to the simple approach of constructing an MCMC sampler to directly target (1). In settings where communication latency is non-negligible but the practitioner's time budget is limited, our approach allows a greater proportion of this time to be spent on computation rather than communication, allowing for faster exploration of the state space.

Our approach was initially presented in Rendell et al. (2018). A proposal to use essentially the same framework in a serial context has been independently and contemporaneously published by Vono et al. (2019a), who construct a Gibbs sampler via a 'variable splitting' approach. Rather than distributing the computation, the authors focus on the setting where $b = 1$ in order to obtain a relaxation of the original simulation problem. An implementation of this approach for problems in binary logistic regression has been proposed in Vono et al. (2018), with a number of non-asymptotic and convergence results presented more recently in Vono et al. (2019b). Our work focuses on distributed settings, providing a sequential Monte Carlo implementation of the framework that may be used to generate bias-corrected estimates.

4

We introduce the proposed framework and the resulting algorithmic structure in Section 2, including some discussion of issues in its implementation, and comparisons with related approaches in the literature. We then introduce a sequential Monte Carlo implementation of the framework in Section 3. Various simulation examples are presented in Section 4, before conclusions are provided in Section 5.

2 The instrumental model and MCMC

For simplicity, we shall occasionally abuse notation by using the same symbol for a probability measure on E, and for its density with respect to some dominating measure. For the numerical examples presented herein, $\mathsf{E} \subseteq \mathbb{R}^d$ and all densities are defined with respect to a suitable version of the Lebesgue measure. We use the notation $x_{m:n} :=$ $(x_m, \ldots, x_n)$ for arbitrary $x_m, \ldots, x_n$. For a probability density function ν and function φ we denote by $\nu(\varphi)$ the expectation of $\varphi(Z)$ when $Z \sim \nu$, i.e.

$$\nu(\varphi) := \int \varphi(z)\nu(z)\,\mathrm{d}z.$$

2.1 The instrumental model

The goal of the present paper is to approximate (1). We take an approach that has also been developed in contemporaneous work by Vono et al. (2019a), although their objectives were somewhat different. Alongside the variable of interest Z, we introduce a collection of b instrumental variables each also defined on E, denoted by $X_{1:b}$. On the extended state space $\mathsf{E} \times \mathsf{E}^b$, we define the probability density function $\tilde{\pi}_\lambda$ by

$$\tilde{\pi}_\lambda(z, x_{1:b}) \propto \mu(z) \prod_{j=1}^{b} K_j^{(\lambda)}(z, x_j) f_j(x_j), \tag{2}$$

where for each $j \in \{1, \ldots, b\}$, $\{K_j^{(\lambda)} : \lambda \in \mathbb{R}_+\}$ is a family of Markov transition densities on E. Defining

$$f_j^{(\lambda)}(z) := \int_{\mathsf{E}} K_j^{(\lambda)}(z, x) f_j(x)\,\mathrm{d}x, \tag{3}$$
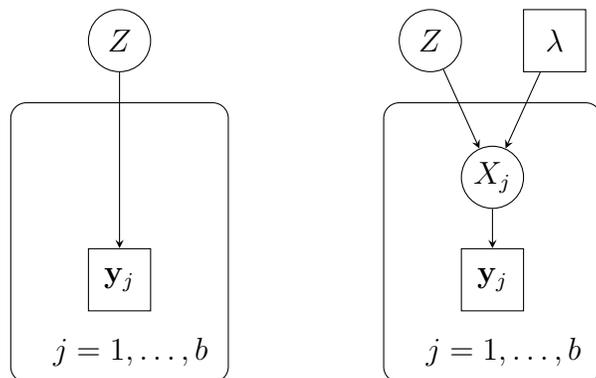
Figure 1: Directed acyclic graphs, representing the original statistical model (left) and the instrumental model we construct (right).

the density of the Z-marginal of $\tilde{\pi}_\lambda$ may be written as

$$\pi_\lambda(z) := \int_{\mathsf{E}^b} \tilde{\pi}_\lambda(z, x_{1:b}) \, \mathrm{d}x_{1:b} \propto \mu(z) \prod_{j=1}^{b} f_j^{(\lambda)}(z). \tag{4}$$

Here, we may view each $f_j^{(\lambda)}$ as a smoothed form of f_j, with π_λ being the corresponding smoothed form of the target density (1).

The role of λ is to control the fidelity of $f_j^{(\lambda)}$ to f_j, and so we assume the following in the sequel.

Assumption 1 *For all $\lambda > 0$, $f_j^{(\lambda)}$ is bounded for each $j \in \{1, \ldots, b\}$; and $f_j^{(\lambda)} \to f_j$ pointwise as $\lambda \to 0$ for each $j \in \{1, \ldots, b\}$.*

For example, Assumption 1 implies that π_λ converges in total variation to π by Scheffé's lemma (Scheffé, 1947), and therefore $\pi_\lambda(\varphi) \to \pi(\varphi)$ for bounded $\varphi : \mathsf{E} \to \mathbb{R}$. Assumption 1 is satisfied for essentially any kernel that may be used for kernel density estimation; here, λ takes a similar role to that of the smoothing bandwidth.

On a first reading one may assume that the $K_j^{(\lambda)}$ are chosen to be independent of j; for example, with $\mathsf{E} = \mathbb{R}$ one could take $K_j^{(\lambda)}(z, x) = \mathcal{N}(x; z, \lambda)$. We describe some considerations in choosing these transition kernels in Section S1.2 of the supplement, and describe settings in which choosing these to differ with j may be beneficial.

The instrumental hierarchical model is presented diagrammatically in Figure 1. The variables $X_{1:b}$ may be seen as 'proxies' for Z associated with each of the data subsets,

which are conditionally independent given Z and λ. Loosely speaking, λ represents the extent to which we allow the local variables $X_{1:b}$ to differ from the global variable Z. In terms of computation, it is the separation of Z from the subsets of the data $\mathbf{y}_{1:b}$, given $X_{1:b}$ introduced by the instrumental model, that can be exploited by distributed algorithms.

This approach to constructing an artificial joint target density is easily extended to accommodate random effects models, in which the original statistical model itself contains local variables associated with each data subset. These variables may be retained in the resulting instrumental model, alongside the local proxies $X_{1:b}$ for Z. A full description of the resulting model is presented in Rendell (2020).

The framework we describe is motivated by concepts in distributed optimisation, a connection that is also explored in the contemporaneous work of Vono et al. (2019a). The global consensus optimisation problem (see, e.g., Boyd et al., 2011, Section 7) is that of minimising a sum of functions on a common domain, under the constraint that their arguments are all equal to some global common value. If for each $j \in \{1, \ldots, b\}$ one uses the Gaussian kernel density $K_j^{(\lambda)}(z, x) = \mathcal{N}(x; z, \lambda)$, then taking the negative logarithm of (2) gives

$$-\log \tilde{\pi}_\lambda(z, x_{1:b}) = C - \log \mu(z) - \sum_{j=1}^{b} \log f_j(x_j) + \frac{1}{2\lambda} \sum_{j=1}^{b} (z - x_j)^2 \qquad (5)$$

where C is a normalising constant. Maximising $\pi(z)$ is equivalent to minimising this function under the constraint that $z = x_j$ for $j \in \{1, \ldots, b\}$, which may be achieved using the alternating direction method of multipliers (Bertsekas and Tsitsiklis, 1989). Specifically, (5) corresponds to the use of $1/\lambda$ as the penalty parameter in this procedure.

There are some similarities between this framework and Approximate Bayesian Computation (ABC; see Marin et al., 2012, for a review of such methods). In both cases one introduces a kernel that can be viewed as acting to smooth the likelihood. In the case of (2) the role of λ is to control the scale of smoothing that occurs in the parameter space; the tolerance parameter used in ABC, in contrast, controls the extent of a comparable form of smoothing in the observation (or summary statistic) space.

2.2 Distributed Metropolis-within-Gibbs

The instrumental model described forms the basis of our proposed global consensus framework; 'global consensus Monte Carlo' is correspondingly the application of Monte Carlo methods to form an approximation of π_λ. We focus here on the construction of a Metropolis-within-Gibbs Markov kernel that leaves $\tilde{\pi}_\lambda$ invariant. If λ is chosen to be sufficiently small, then the Z-marginal π_λ provides an approximation of π. Therefore given a chain with values denoted $(Z^i, X^i_{1:b})$ for $i \in \{1, \ldots, N\}$, an empirical approximation of π is given by

$$\pi_\lambda^N := \frac{1}{N} \sum_{i=1}^{N} \delta_{Z^i}, \tag{6}$$

where δ_z denotes the Dirac measure at z.

The Metropolis-within-Gibbs kernel we consider utilises the full conditional densities

$$\tilde{\pi}_\lambda(x_j \mid z) \propto K_j^{(\lambda)}(z, x_j) f_j(x_j) \tag{7}$$

for $j \in \{1, \ldots, b\}$, and

$$\tilde{\pi}_\lambda(z \mid x_{1:b}) \propto \mu(z) \prod_{j=1}^{b} K_j^{(\lambda)}(z, x_j), \tag{8}$$

where (7) follows from the mutual conditional independence of $X_{1:b}$ given Z. Here we observe that $K_j^{(\lambda)}(z, x_j)$ simultaneously provides a pseudo-prior for X_j and a pseudo-likelihood for Z.

We define $M_1^{(\lambda)}$ to be a $\tilde{\pi}_\lambda$-invariant Markov kernel that fixes z; we may write

$$M_1^{(\lambda)}((z, x_{1:b}); d(z', x'_{1:b})) = \delta_z(dz') \prod_{j=1}^{b} P_{j,z}^{(\lambda)}(x_j, dx'_j), \tag{9}$$

where for each j, $P_{j,z}^{(\lambda)}(x_j, \cdot)$ is a Markov kernel leaving (7) invariant. We similarly define $M_2^{(\lambda)}$ to be a $\tilde{\pi}_\lambda$-invariant Markov kernel that fixes $x_{1:b}$,

$$M_2^{(\lambda)}((z, x_{1:b}); d(z', x'_{1:b})) = \left[\prod_{j=1}^{b} \delta_{x_j}(dx'_j)\right] P_{x_{1:b}}^{(\lambda)}(z, dz'), \tag{10}$$

where $P_{x_{1:b}}^{(\lambda)}(z, \cdot)$ is a Markov kernel leaving (8) invariant.

Using these Markov kernels we construct an MCMC kernel that leaves $\tilde{\pi}_\lambda$ invariant; we present the resulting sampling procedure as Algorithm 1. In the special case in which

one may sample exactly from the conditional distributions (7)–(8), this algorithm takes the form of a Gibbs sampler.

Algorithm 1 Global consensus Monte Carlo: MCMC algorithm

Fix $\lambda > 0$. Set initial state $(Z^0, X_{1:b}^0)$; choose chain length N.

For $i = 1, \ldots, N$:

- For $j \in \{1, \ldots, b\}$, sample $X_j^i \sim P_{j,Z^{i-1}}^{(\lambda)}(X_j^{i-1}, \cdot)$.

- Sample $Z^i \sim P_{X_{1:b}^i}^{(\lambda)}(Z^{i-1}, \cdot)$.

Return $(Z^i, X_{1:b}^i)_{i=1}^N$.

The interest from a distributed perspective is that the full conditional density (7) of each X_j, for given values x_j and z, depends only on the jth block of data (through the partial likelihood f_j) and may be computed on the jth machine. Within Algorithm 1, the sampling of each X_j^i from $P_{j,Z^{i-1}}^{(\lambda)}(X_j^{i-1}, \cdot)$ may therefore occur on the jth machine; these $X_{1:b}^i$ may then be communicated to a central machine that draws Z^i.

Our approach has particular benefits when sampling exactly from (7) is not possible, in which case Algorithm 1 takes a Metropolis-within-Gibbs form. One may choose the Markov kernels $P_{j,z}^{(\lambda)}$ to comprise multiple iterations of an MCMC kernel leaving (7) invariant; indeed multiple computations of each f_j (and therefore multiple accept/reject steps) may be conducted on each of the b nodes, without requiring communication between machines. This stands in contrast to more straightforward MCMC approaches directly targeting π, in which such communication is required for each evaluation of (1), and therefore for every accept/reject step. Similar to such a 'direct' MCMC approach, each iteration of Algorithm 1 requires communication to and from each of the b machines on which the data are stored; but in cases where the communication latency is high, the resulting sampler will spend a greater proportion of time exploring the state space. This may in turn result in faster mixing (e.g. with respect to wall-clock time). We further discuss this comparison, and the role of communication latency, in Section S1.1

of the supplement.

The setting in which Algorithm 1 describes a Gibbs sampler, with all variables drawn exactly from their full conditional distributions, is particularly amenable to analysis. We provide such a study in Section S2 of the supplement. This analysis may also be informative about the more general Metropolis-within-Gibbs setting, when $P_{j,z}^{(\lambda)}$ comprises enough MCMC iterations to exhibit good mixing.

2.3 Choosing the regularisation parameter

For $\varphi : \mathsf{E} \to \mathbb{R}$ we may estimate $\pi(\varphi)$ using (6) as $\pi_\lambda^N(\varphi)$. The regularisation parameter λ here takes the role of a tuning parameter; we can view its effect on the mean squared error of such estimates using the bias–variance decomposition,

$$\mathbb{E}\left[\left(\pi_\lambda^N(\varphi) - \pi(\varphi)\right)^2\right] = [\pi_\lambda(\varphi) - \pi(\varphi)]^2 + \mathrm{var}[\pi_\lambda^N(\varphi)], \tag{11}$$

which holds exactly when $\mathbb{E}[\pi_\lambda^N(\varphi)] = \pi_\lambda(\varphi)$. In many practical cases this decomposition will provide a very accurate approximation for large N, as the squared bias of $\pi_\lambda^N(\varphi)$ is typically asymptotically negligible in comparison to its variance.

The decomposition (11) separates the contributions to the error from the bias introduced by the instrumental model and the variance associated with the MCMC approximation. If λ is too large, the squared bias term in (11) can dominate while if λ is too small, the Markov chain may exhibit poor mixing due to strong conditional dependencies between $X_{1:b}$ and Z, and so the variance term in (11) can dominate.

It follows that λ should ideally be chosen in order to balance these two considerations. We provide a theoretical analysis of the role of λ in Section S2 of the supplement, by considering a simple Gaussian setting. In particular we show that for a fixed number of data, one should scale λ with the number of samples N as $\mathcal{O}(N^{-1/3})$ in order to minimise the mean squared error. We also consider the consistency of the approximate posterior π_λ as the number of data n tends to infinity. Specifically, suppose these are split equally into b blocks; considering λ and b as functions of n, we find that π_λ exhibits

posterior consistency if λ/b decreases to 0 as $n \to \infty$, and that credible intervals have asymptotically correct coverage if λ/b decreases at a rate strictly faster than n^{-1}.

As an alternative to selecting a single value of λ, we propose in Section 3 a Sequential Monte Carlo sampler employing Markov kernels formed via Algorithm 1. In this manner a decreasing sequence of λ values may be considered, which may result in lower-variance estimates for small λ values; we also describe a possible bias correction technique.

2.4   Related approaches

As previously mentioned, a Gibbs sampler construction essentially corresponding to Algorithm 1 has independently been proposed by Vono et al. (2019a). Their main objective is to improve algorithmic performance when computation of the full posterior density is intensive by constructing full conditional distributions that are more computationally tractable, for which they primarily consider a setting in which $b = 1$. In contrast, we focus on the exploitation of this framework in distributed settings (i.e. with $b > 1$), in the manner described in Section 2.2.

The objectives of our algorithm are similar, but not identical, to those of the previously-introduced 'embarrassingly parallel' approaches proposed by many authors. These reduce the costs of communication latency to a near-minimum by simulating a separate MCMC chain on each machine; typically, the chain on the jth machine is invariant with respect to a 'subposterior' distribution with density proportional to $\mu(z)^{1/b} f_j(z)$. Communication is necessary only for the final aggregation step, in which an approximation of the full posterior is obtained using the samples from all b chains.

A well-studied approach within this framework is Consensus Monte Carlo (Scott et al., 2016), in which the post-processing step amounts to forming a 'consensus chain' by weighted averaging of the separate chains. In the case that each subposterior density is Gaussian this approach can be used to produce samples asymptotically distributed according to π, by weighting each chain using the precision matrices of the subposterior distributions. Motivated by Bayesian asymptotics, the authors suggest using

11

this approach more generally. In cases where the subposterior distributions exhibit near-Gaussianity this performs well, with the final 'consensus chain' providing a good approximation of posterior expectations. However, there are no theoretical guarantees associated with this approach in settings in which the subposterior densities are poorly approximated by Gaussians. In such cases consensus Monte Carlo sometimes performs poorly in forming an approximation of the posterior π (as in examples of Wang et al., 2015; Srivastava et al., 2015; Dai et al., 2019), and so the resulting estimates of integrals $\pi(\varphi)$ exhibit high bias.

Various authors (e.g. Minsker et al., 2014; Rabinovich et al., 2015; Srivastava et al., 2015; Wang et al., 2015) have therefore proposed alternative techniques for utilising the values from each of these chains in order to approximate posterior expectations, each of which presents benefits and drawbacks. For example, Neiswanger et al. (2014) propose a strategy based on kernel density estimation; based on this approach, Scott (2017) suggests a strategy based on finite mixture models, though notes that both methods may be impractical in high-dimensional settings.

An aggregation procedure proposed by Wang and Dunson (2013) bears some relation to our proposed framework, being based on the application of Weierstrass transforms to each subposterior density. The resulting smoothed densities are analogous to (3), which represents a smoothed form of the partial likelihood f_j. As well as proposing an aggregation method based on rejection sampling, the authors propose a technique for 'refining' an initial posterior approximation, which may be expressed in terms of a Gibbs kernel on an extended state space. Comparing with our framework, this is analogous to applying Algorithm 1 for one iteration with N different initial values.

A potential issue common to these approaches is the treatment of the prior density μ. Each subposterior density is typically assigned an equal share of the prior information in the form of a fractionated prior density $\mu(z)^{1/b}$, but it is not clear when this approach is satisfactory. For example, suppose μ belongs to an exponential family; any property that is not invariant to multiplying the canonical parameters by a constant will not be preserved in the fractionated prior. As such if $\mu(z)^{1/b}$ is proportional to

a valid probability density function of z, then the corresponding distribution may be qualitatively very different to the full prior. Although Scott et al. (2016) note that fractionated priors perform poorly on some examples (for which tailored solutions are provided), no other general way of assigning prior information to each block naturally presents itself. In contrast our approach avoids this problem entirely, with μ providing prior information for Z at the 'global' level.

Finally, we believe this work is complementary to other distributed algorithms lying outside of the 'embarrassingly parallel' framework (including Xu et al., 2014; Jordan et al., 2019), and to approaches that aim to reduce the amount of computation associated with each likelihood calculation on a single node, e.g. by using only a subsample or batch of the data (as in Korattikara et al., 2014; Bardenet et al., 2014; Huggins et al., 2016; Maclaurin and Adams, 2014).

3 Sequential Monte Carlo approach

As discussed in Section 2.3, as λ approaches 0 estimators $\pi_\lambda^N(\varphi)$ formed using (6) exhibit lower bias but higher variance, due to poorer mixing of the resulting Markov chain. In order to obtain lower-variance estimators for λ values close to 0, we consider the use of Sequential Monte Carlo (SMC) methodology to generate suitable estimates for a sequence of λ values.

SMC methodology employs sequential importance sampling and resampling; recent surveys include Doucet and Johansen (2011) and Doucet and Lee (2018). We consider here approximations of a sequence of distributions with densities $\tilde{\pi}_{\lambda_0}, \tilde{\pi}_{\lambda_1}, \ldots$, where $\lambda_0, \ldots, \lambda_n$ is a decreasing sequence. The procedure we propose, specified in Algorithm 2, is an SMC sampler within the framework of Del Moral et al. (2006). This algorithmic form was first proposed by Gilks and Berzuini (2001) and Chopin (2002) in different settings, building upon ideas in Crooks (1998), Neal (2001).

The algorithm presented involves simulating particles using $\tilde{\pi}_\lambda$-invariant Markov kernels, and has a genealogical structure imposed by the ancestor indices A_p^i for $p \in$

Algorithm 2 Global consensus Monte Carlo: SMC algorithm

Fix a decreasing sequence $(\lambda_0, \lambda_1, \ldots, \lambda_n)$. Set number of particles N.

Initialise:

- For $i \in \{1, \ldots, N\}$, sample $\zeta_0^i = (Z_0^i, X_{0,1:b}^i) \sim \tilde{\pi}_{\lambda_0}$ and set $W_0^i \leftarrow 1$.

For $p = 1, \ldots, n$:

1. For $i \in \{1, \ldots, N\}$, set $\tilde{W}_p^i \leftarrow W_{p-1}^i w_p(\zeta_{p-1}^i)$, where

$$w_p(z, x_{1:b}) := \frac{\tilde{\pi}_{\lambda_p}(z, x_{1:b})}{\tilde{\pi}_{\lambda_{p-1}}(z, x_{1:b})} = \prod_{j=1}^{b} \frac{K_j^{(\lambda_p)}(z, x_j)}{K_j^{(\lambda_{p-1})}(z, x_j)}.$$

2. Optionally, carry out a resampling step: for $i \in \{1, \ldots, N\}$,

    - Sample $A_{p-1}^i \sim \text{Categorical}(\tilde{W}_p^1, \ldots, \tilde{W}_p^N)$ independently.

    - Set $W_p^i \leftarrow 1$.

    Otherwise: for $i \in \{1, \ldots, N\}$ set $A_{p-1}^i \leftarrow i$, $W_p^i \leftarrow \tilde{W}_p^i$.

3. For $i \in \{1, \ldots, N\}$, sample $\zeta_p^i = (Z_p^i, X_{p,1:b}^i) \sim M_p(\zeta_{p-1}^{A_{p-1}^i}, \cdot)$, where M_p is a $\tilde{\pi}_{\lambda_p}$-invariant MCMC kernel constructed in the manner of Algorithm 1.

4. Optionally, store the particle approximation of $\tilde{\pi}_{\lambda_p}$,

$$\tilde{\pi}_{\lambda_p}^N := \frac{\sum_{i=1}^{N} W_p^i \delta_{\zeta_p^i}}{\sum_{i=1}^{N} W_p^i}.$$

$\{0, \dots, n-1\}$ and $i \in \{1, \dots, N\}$. The specific scheme for simulating the ancestor indices here is known as multinomial resampling; other schemes can be used (see Douc et al., 2005; Gerber et al., 2019, for a summary of some schemes and their properties). We use this simple scheme here as it validates the use of the variance estimators used in Section 3.1. This optional resampling step is used to prevent the degeneracy of the particle set; a common approach is to carry out this step whenever the particles' effective sample size (Liu and Chen, 1995) falls below a pre-determined threshold.

Under weak conditions $\tilde{\pi}^N_{\lambda_p}(\varphi)$ converges almost surely to $\tilde{\pi}_{\lambda_p}(\varphi)$ as $N \to \infty$. One can also define the particle approximations of π_{λ_p} via

$$\pi^N_{\lambda_p} := \frac{\sum_{i=1}^{N} W^i_p \delta_{Z^i_p}}{\sum_{i=1}^{N} W^i_p}, \tag{12}$$

where Z^i_p is the first component of the particle ζ^i_p.

Although the algorithm is specified for simplicity in terms of a fixed sequence $\lambda_0, \dots, \lambda_n$, a primary motivation for the SMC approach is that the sequence used can be determined adaptively while running the algorithm. A number of such procedures have been proposed in the literature in the context of tempering, allowing each value λ_p to be selected based on the particle approximation of $\tilde{\pi}_{\lambda_{p-1}}$. For example, Jasra et al. (2011) propose a procedure that controls the decay of the particles' effective sample size. Within the examples in Section 4 we employ a procedure proposed by Zhou et al. (2016), which generalises this approach to settings in which resampling is not conducted in every iteration, aiming to control directly the dissimilarity between successive distributions. A possible approach to determining when to terminate the procedure, based on minimising the mean squared error (11), is detailed in Section S3.2 of the supplement.

With regard to initialisation, if it is not possible to sample from $\tilde{\pi}_{\lambda_0}$ one could instead use samples obtained by importance sampling, or one could initialise an SMC sampler with some tractable distribution and use tempering or similar techniques to reach $\tilde{\pi}_{\lambda_0}$. At the expense of the introduction of an additional approximation, an alternative would be to run a $\tilde{\pi}_{\lambda_0}$-invariant Markov chain, and obtain an initial collection of particles by

thinning the output (an approach that may be validated using results of Finke et al., 2020). Specifically, one could use Algorithm 1 to generate such samples for some large λ_0, benefiting from its good mixing and low autocorrelation when λ is sufficiently large. The effect of Algorithm 2 may then be seen as refining or improving the resulting estimators, by bringing the parameter λ closer to zero.

Other points in favour of this approach are that many of the particle approximations (12) can be used to form a final estimate of $\pi(\varphi)$ as explored in Section 3.1, and that SMC methods can be more robust to multimodality of π than simple Markov chain schemes. We finally note that in such an SMC sampler, a careful implementation of the MCMC kernels used may allow the inter-node communication to be interleaved with likelihood computations associated with the particles, thereby reducing the costs associated with communication latency.

3.1   Bias correction using local linear regression

We present an approach to use many of the particle approximations produced by Algorithm 2. A natural idea is to regress the values of $\pi_\lambda^N(\varphi)$ on λ, extrapolating to $\lambda = 0$ to obtain an estimate of $\pi(\varphi)$. A similar idea has been used for bias correction in the context of Approximate Bayesian Computation, albeit not in an SMC setting, regressing on the discrepancy between the observed data and simulated pseudo-observations (Beaumont et al., 2002; Blum and François, 2010).

Under very mild assumptions on the transition densities $K_j^{(\lambda)}$, $\pi_\lambda(\varphi)$ is smooth as a function of λ. Considering a first-order Taylor expansion of this function, a simple approach is to model the dependence of $\pi_\lambda(\varphi)$ on λ as linear, for λ sufficiently close to 0. Having determined a subset of the values of λ used for which a linear approximation is appropriate (we propose a heuristic approach in Section S3.1 of the supplement) one can use linear least squares to carry out the regression. To account for the SMC estimates $\pi_{\lambda_p}^N(\varphi)$ having different variances, we propose the use of weighted least squares, with the 'observations' $\pi_{\lambda_p}^N(\varphi)$ assigned weights inversely proportional to their estimated

16

variances; we describe methods for computing such variance estimates in Section 3.2. A bias-corrected estimate of $\pi(\varphi)$ is then obtained by extrapolating the resulting fit to $\lambda = 0$, which corresponds to taking the estimated intercept term.

To make this explicit, first consider the case in which $\varphi : \mathsf{E} \to \mathbb{R}$, so that the estimates $\pi_\lambda^N(\varphi)$ are univariate. For each value λ_p denote the corresponding SMC estimate by $\eta_p := \pi_{\lambda_p}^N(\varphi)$, and let v_p denote some proxy for the variance of this estimate. Then for some set of indices $S := \{p^*, \dots, n\}$ chosen such that the relationship between η_p and λ_p is approximately linear for $p \in S$, a bias-corrected estimate for $\pi(\varphi)$ may be computed as

$$\pi_S^{\mathrm{BC}}(\varphi) := \tilde{\eta}_S - \tilde{\lambda}_S \frac{\sum_{p \in S}(\lambda_p - \tilde{\lambda}_S)(\eta_p - \tilde{\eta}_S)/v_p}{\sum_{p \in S}(\lambda_p - \tilde{\lambda}_S)^2/v_p}, \tag{13}$$

where $\tilde{\lambda}_S$ and $\tilde{\eta}_S$ denote weighted means given by

$$\tilde{\lambda}_S := \frac{\sum_{p \in S} \lambda_p/v_p}{\sum_{p \in S} 1/v_p}, \qquad \tilde{\eta}_S := \frac{\sum_{p \in S} \eta_p/v_p}{\sum_{p \in S} 1/v_p}.$$

The formal justification of this estimate assumes that the observations are uncorrelated, which does not hold here. We demonstrate in Section 4 examples on which this simple approach is nevertheless effective, but in principle one could use generalised least squares combined with some approximation of the full covariance matrix of the SMC estimates.

In the more general case where $\varphi : \mathsf{E} \to \mathbb{R}^d$ for $d > 1$, we propose simply evaluating (13) for each component of this quantity separately, which corresponds to fitting an independent weighted least squares regression to each component. This facilitates the use of the variance estimators described in the following section, though in principle one could use multivariate weighted least squares or other approaches.

3.2 Variance estimation for weighted least squares

We propose the weighted form of least squares here since, as the values of λ used in the SMC procedure approach zero, the estimators generated may increase in variance: partly due to poorer mixing of the MCMC kernels as previously described, but also due to the gradual degeneracy of the particle set. In order to estimate the variances of estimates generated using SMC, several recent approaches have been proposed that allow

this estimation to be carried out online by considering the genealogy of the particles. Using any such procedure, one may estimate the variance of $\pi_\lambda^N(\varphi)$ for each λ value considered by Algorithm 2, with these values used for each v_p in (13).

Within our examples, we use the estimator proposed by Lee and Whiteley (2018), which for fixed N coincides with an earlier proposal of Chan and Lai (2013) (up to a multiplicative constant). Specifically, after each step of the SMC sampler we compute an estimate of the *asymptotic* variance of each estimate $\pi_{\lambda_p}^N(\varphi)$; that is, the limit of $N \operatorname{var}[\pi_{\lambda_p}^N(\varphi)]$ as $N \to \infty$. While this is not equivalent to computing the true variance of each estimate, for fixed large N the relative sizes of these asymptotic variance estimates should provide a useful indicator of the relative variances of each estimate $\pi_\lambda^N(\varphi)$. In Section S4.1 of the supplement we show empirically that inversely weighting the SMC estimates according to these estimated variances can result in more stable bias-corrected estimates as the particle set degenerates. We also explain in Section S3.2 how these estimated variances can be used within a rule to determine when to terminate the algorithm.

The asymptotic variance estimator described is consistent in N. However, if in practice resampling at the pth time step causes the particle set to degenerate to having a single common ancestor, then the estimator evaluates to zero, and so it is impossible to use this value as the inverse weight v_p in (13). Such an outcome may be interpreted as a warning that too few particles have been used for the resulting SMC estimates to be reliable, and that a greater number should be used when re-running the procedure.

4 Examples

4.1 Log-normal toy example

To compare the posterior approximations formed by our global consensus framework with those formed by some embarrassingly parallel approaches discussed in Section 2.4, we conduct a simulation study based on a simple model. Let $\mathcal{LN}(x; \mu, \sigma^2)$ denote the

density of a log-normal distribution with parameters (μ, σ^2); that is,

$$\mathcal{LN}(x; \mu, \sigma^2) = \frac{1}{x\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\log(x) - \mu)^2}{2\sigma^2}\right).$$

One may consider a model with prior density $\mu(z) = \mathcal{LN}(z; \mu_0, \sigma_0^2)$ and likelihood contributions $f_j(z) = \mathcal{LN}(\log(\mu_j); \log(z), \sigma_j^2)$ for $j \in \{1, \ldots, b\}$. This may be seen as a reparametrisation of the Gaussian model in Section S2 of the supplement, in which each likelihood contribution is that of a data subset with a Gaussian likelihood. This convenient setting allows for the target distribution π to be expressed analytically. For the implementation of the global consensus algorithm, we choose Markov transition kernels given by $K_j^{(\lambda)}(z, x) = \mathcal{LN}(x; \log(z), \lambda)$ for each $j \in \{1, \ldots, b\}$, which satisfy Assumption 1; this allows for exact sampling from all the full conditional distributions.

As a toy example to illustrate the effects of non-Gaussian partial likelihoods we consider a case in which $f_j(z) = \mathcal{LN}(\log(\mu_j); \log(z), 1)$ for each j, and $\mu(z) = \mathcal{LN}(z; 0, 25)$. Here we took $b = 32$, and selected the location parameters μ_j as i.i.d. samples from a standard normal distribution. We ran Global Consensus Monte Carlo (GCMC) using the Gibbs sampler form of Algorithm 1, for values of λ between 10^{-5} and 10. For comparison we also drew samples from each subposterior distribution as defined in Section 2.4, combining the samples using various approaches. These are the Consensus Monte Carlo (CMC) averaging of Scott et al. (2016); the nonparametric density product estimation (NDPE) approach of Neiswanger et al. (2014); and the Weierstrass rejection sampling (WRS) combination technique of Wang and Dunson (2013), using their R implementation (`https://github.com/wwrechard/weierstrass`). In each case we ran the algorithm 25 times, drawing $N = 10^5$ samples.

To demonstrate the role of λ in the bias–variance decomposition (11), Table 1 presents the means and standard deviations of estimates of $\pi(\varphi)$, for various test functions φ. In estimating the first moment of π, GCMC generates a low-bias estimator when λ is chosen to be sufficiently small; however, as expected, the variance of such estimators increases when very small values of λ are chosen. While the other methods produce estimators of reasonably low variance, these exhibit somewhat higher bias. For

19

		$\varphi(z) = z$	$\varphi(z) = z^5$	$\varphi(z) = \log(z)$
Algorithm		$(\pi(\varphi) = 1.141)$	$(\pi(\varphi) = 2.644)$	$(\pi(\varphi) = 0.1164)$
GCMC	$\lambda = 10^1$	1.329 ± 0.003	121.154 ± 10.487	0.1151 ± 0.0019
	$\lambda = 10^0$	1.159 ± 0.002	$3.901 \pm\ \ 0.037$	$\mathbf{0.1165 \pm 0.0014}$
	$\lambda = 10^{-1}$	$\mathbf{1.144 \pm 0.003}$	$\mathbf{2.763 \pm\ \ 0.044}$	0.1173 ± 0.0030
	$\lambda = 10^{-2}$	1.140 ± 0.011	$2.648 \pm\ \ 0.143$	0.1150 ± 0.0090
	$\lambda = 10^{-3}$	1.142 ± 0.022	$2.661 \pm\ \ 0.295$	0.1191 ± 0.0199
	$\lambda = 10^{-4}$	1.120 ± 0.077	$3.505 \pm\ \ 1.136$	0.1630 ± 0.0651
	$\lambda = 10^{-5}$	1.400 ± 0.110	$6.195 \pm\ \ 2.217$	0.3283 ± 0.0810
CMC		1.073 ± 0.010	$16.092 \pm\ \ 5.675$	0.0135 ± 0.0095
NDPE		1.148 ± 0.029	$2.800 \pm\ \ 0.385$	0.1231 ± 0.0246
WRS		1.111 ± 0.007	$2.444 \pm\ \ 0.086$	0.0862 ± 0.0063

Table 1: True values and estimates of $\pi(\varphi)$, for various test functions φ, for the log-normal toy model. Estimates obtained using Global Consensus Monte Carlo with various values of λ, and three embarrassingly parallel methods (see main text for abbreviations). For each method the mean estimate $\pm$ Monte Carlo standard error is presented, as computed over 25 replicates; the estimator corresponding to the lowest mean squared error is printed in bold.

CMC the bias is especially pronounced when estimating higher moments of the posterior distribution, as exemplified by the estimates of the fifth moment. Note however that high biases are also introduced when using Global Consensus Monte Carlo with large values of λ (as seen here with $\lambda = 10$), for which π_λ is a poor approximation of π.

Also of note are estimates of $\int \log(z)\pi(z)\,\mathrm{d}z$, corresponding to the mean of the Gaussian model of which this a reparametrisation. While Global Consensus Monte Carlo performs well across a range of λ values, the other methods perform less favourably; Consensus Monte Carlo produces an estimate that is incorrect by an order of magnitude. While this could be solved by a simple reparametrisation of the problem in this case, in more general settings no such straightforward solution may exist.

In Section S4.2 of the supplement we present second example based on a log-normal model, demonstrating the robustness of these methods to permutation and repartitioning of the data.

4.2   Logistic regression

Binary logistic regression models are commonly used in settings related to marketing. In web design for example, A/B testing may be used to determine which content choices lead to maximised user interaction, such as the user clicking on a product for sale.

We assume that we have a data set of size n formed of responses $\eta_l \in \{-1, 1\}$, and vectors $\xi_l \in \{0, 1\}^d$ of binary covariates, where $l \in \{1, \ldots, n\}$. The likelihood contribution of each block of data then takes the form $f_j(z) = \prod_l S(\eta_l z^\mathsf{T} \xi_l)$, $z \in \mathbb{R}^d$, where the product is taken over those indices l included in the jth block of data, and S denotes the logistic function, $S(x) = (1 + \exp(x))^{-1}$.

For the prior μ, we use a product of independent zero-mean Gaussians, with standard deviation 20 for the parameter corresponding to the intercept term, and 5 for all other parameters. For the Markov transition densities in GCMC, we use multivariate Gaussian densities: $K_j^{(\lambda)}(z, x) = \mathcal{N}(x; z, \lambda I)$ for each $j \in \{1, \ldots b\}$.

We investigated several such simulated data sets and the efficacy of various ap-

proaches in approximating the true posterior π. To illustrate the bias–variance trade-off described in Section 2.3, in the presentation of these results we focus on the estimation of the posterior first moment; denoting the identity function on $\mathbb{R}^d$ by Id, we may write this as $\pi(\text{Id})$. While our global consensus approach was consistently successful in forming estimators with low mean squared error in each component, in low-dimensional settings the application of Consensus Monte Carlo often resulted in marginal improvements. However, in many higher-dimensional settings, the estimators resulting from CMC exhibited relatively large biases.

We present here an example in which the d predictors correspond to p binary input variables, their pairwise products, and an intercept term, so that $d = 1 + p + \binom{p}{2}$. In settings where the interaction effects corresponding to these pairwise products are of interest, the dimensionality d of the space can be very large compared to p. We used a simulated data set with $p = 20$ input variables, resulting in a parameter space of dimension $d = 211$. The data comprise $n = 80{,}000$ observations, split into $b = 8$ equally-sized blocks. Each observation of the 20 binary variables was generated from a Bernoulli distribution with parameter 0.1, and for each vector of covariates, the response was generated from the correct model, for a fixed underlying parameter vector z^*.

4.2.1 Metropolis-within-Gibbs

We applied GCMC for values of λ between 10^{-2} and 1. We used a Metropolis-within-Gibbs formulation of Algorithm 1, sampling directly from the Gaussian conditional distribution of Z given $X_{1:b}$. To sample approximately from the conditional distributions of each X_j given Z we used Markov kernels $P_{j,z}^{(\lambda)}$ comprising $k = 20$ iterations of a random walk Metropolis kernel.

As mentioned in Section 2.2, in settings of high communication latency our approach allows a greater proportion of wall-clock time to be spent on likelihood contributions, compared to an MCMC chain directly targeting the full posterior π. To compare across settings, we therefore consider an abstracted distributed setting as discussed in Section S1.1 of the supplement, here assuming that the latency is 10 times the time

taken to compute each partial likelihood f_j (in the notation of Section S1.1, $C = 10\ell$).

We also compare with the same embarrassingly parallel approaches as in Section 4.1 (CMC, NDPE, WRS), which are comparatively unaffected by communication latency. For these methods, we again used random walk Metropolis to draw samples from each subposterior distribution. To ease computation, we thinned these chains before applying the combination step; in practice, the estimators obtained using these thinned chains behaved very similarly to those obtained using all subposterior samples.

To provide a 'ground truth' against which to compare the results we ran a random walk Metropolis chain of length 500,000 targeting π. For all our random walk Metropolis samplers we used Gaussian proposal kernels. To determine the covariance matrices of these, we formed a Laplace approximation of the target density following the approach of Chopin and Ridgway (2017), scaling the resulting covariance matrix optimally according to results of Roberts and Rosenthal (2001).

For each algorithmic setting, we ran the corresponding sampler 25 times. To compare the resulting estimators of the posterior mean we computed the mean squared error of each of the d components of the posterior mean, summing these to obtain a 'mean sum of squared errors'.

Table 2 compares the values obtained by each algorithm after an approximate wall-clock time equal to 200,000 times the time taken to compute a single partial likelihood f_j. Accounting for latency in the abstracted distributed setting described above, the GCMC approach is able to generate 5000 approximate posterior samples during this time, spending 50% of time on likelihood computations. In contrast, a direct MCMC approach generates 9523 samples, but would only spend 4.8% of the time on likelihood computations, with the remainder lost due to latency.

The result is that the estimators generated by GCMC for appropriately-chosen λ exhibit lower mean sums of squared errors: we conduct many more accept/reject steps in each round of inter-node communication than if we were to directly target π, and so it becomes possible to achieve faster mixing of the Z-chain (and a better estimator) compared to such a direct approach. This may be seen when comparing the effective

Algorithm		Mean sum of squared errors
GCMC	$\lambda = 10^0$	0.1835
	$\lambda = 10^{-0.5}$	0.1379
	$\lambda = 10^{-1}$	0.0770
	$\lambda = 10^{-1.5}$	**0.0478**
	$\lambda = 10^{-2}$	0.0662
CMC		0.3710
NDPE		0.8476
WRS		0.6402
Direct MCMC		0.0884

Table 2: Mean sum of squared errors over all d components of estimates of the posterior mean for the logistic regression model, formed using various algorithmic approaches as described in the main text, during an approximate wall-clock time equal to 200,000 times that required to compute a single partial likelihood f_j. All values computed over 25 replicates, with the lowest value printed in bold.
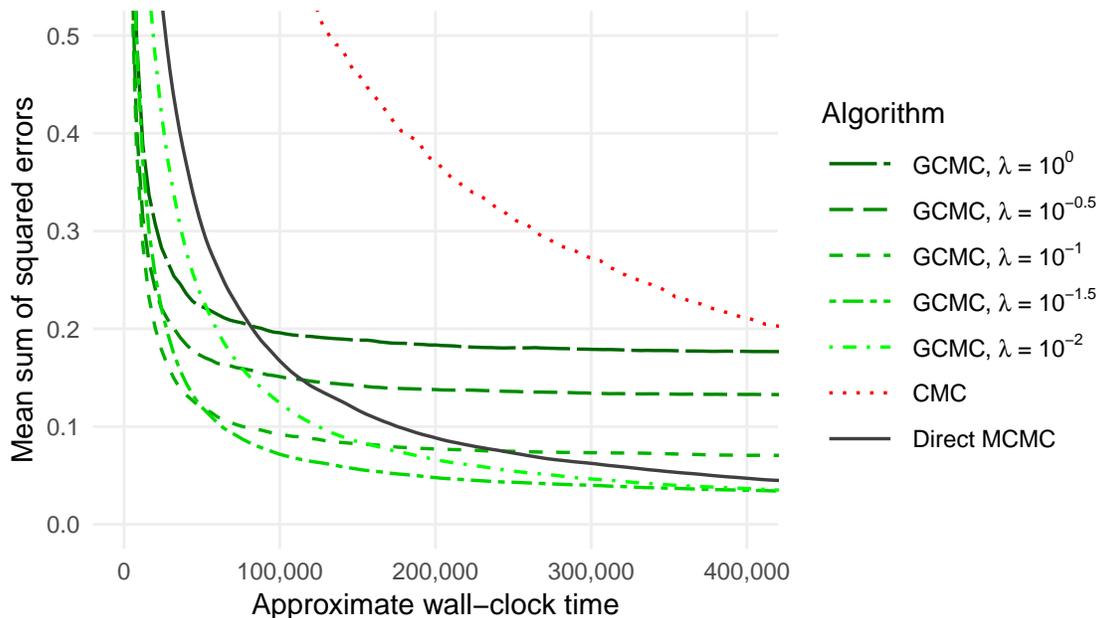
Figure 2: Mean sum of squared errors over all d components of estimates of the posterior mean for the logistic regression model, formed using various algorithmic approaches as described in the main text. Values plotted against the approximate wall-clock time, relative to the time taken to compute a single partial likelihood term. All values computed over 25 replicates.

sample size (ESS) of each chain, where we estimate this via the 'batch means' approach of Vats et al. (2019): we find that the average ESS of the direct MCMC chains is only 1111, while depending on the choice of λ, the shorter GCMC chains have average ESS values between 1327 and 4577.

Despite being unaffected by latency and therefore allowing many more samples to be drawn, the embarrassingly parallel approaches (CMC, NDPE, WRS) perform poorly compared to GCMC. This is particularly true of the nonparametric density product estimation (NDPE) method of Neiswanger et al. (2014): while asymptotically exact even in non-Gaussian settings, the resulting estimator is based on kernel density estimators and is not effective in this high-dimensional setting.

Figure 2 shows the mean sums of squared errors as a function of the approximate

wall-clock time (for simplicity we include only the best-performing of the three embarrassingly parallel methods, omitting the results for NDPE and WRS). We see that for large enough λ, the GCMC estimators $\pi_\lambda^N(\mathrm{Id})$ exhibit rather lower values than the corresponding CMC and 'direct' MCMC estimators. As the number of samples used grows, the squared bias of these estimators begins to dominate, and so smaller λ values result in lower mean squared errors. As λ becomes smaller the autocorrelation of the resulting Z-chain increases; indeed we found that for λ too small, the GCMC estimator $\pi_\lambda^N(\mathrm{Id})$ will always have a greater mean squared error than the 'direct' MCMC estimator, no matter how much time is used. Of course, since an MCMC estimator formed by directly targeting π is consistent in N, given sufficient time such an estimator will always outperform estimators formed using GCMC, which are biased for any λ. However, in many practical big data settings it may be infeasible to draw large numbers of samples using the available time budget.

4.2.2 Sequential Monte Carlo

We also applied the SMC procedure to this logistic regression model. While we found that the SMC approach was most effective in lower-dimensional settings (see Section S4.1 of the supplement for a simple example) in which it is less computationally expensive, the SMC procedure can be more widely useful as a means of 'refining' the estimator formed using a single λ value, as discussed in Section 3.

We used $N = 1250$ particles, initialising the particle set by thinning the chain generated by the Metropolis-within-Gibbs procedure with $\lambda = 10^{-1}$. To generate a sequence of subsequent λ values we used the adaptive procedure of Zhou et al. (2016), using tuning parameter $\mathrm{CESS}^\star = 0.98$. For the Markov kernels M_p we used Metropolis-within-Gibbs kernels as previously, with each update of X_j given Z comprising $k = 50$ iterations of a random walk Metropolis kernel.

The estimator $\pi_{\lambda_0}^N(\mathrm{Id})$ formed using the initial particle set was found to have a mean sum of squared errors of 0.0692. After a fixed number of iterations ($n = 100$) the resulting SMC estimate exhibited a mean sum of squared errors of 0.0418; this

represents a decrease of 40%, and has the benefit of avoiding the need to carefully specify a single value for λ.

Used alone, the bias correction procedure of Section 3.1 was found to perform best in lower-dimensional settings (as in Section S4.1 of the supplement); here, it resulted in a mean sum of squared errors of 0.0682 after 100 iterations. However, improved results were obtained using the stopping rule we propose in Section S3.2 of the supplement (with stopping parameter $\kappa = 15$), which is based on our proposed bias correction procedure. The estimator selected by this stopping rule, which automatically determines when to terminate the algorithm, obtained a mean sum of squared errors of 0.0367, a decrease of 47% from the estimator generated using the initial particle set.

5 Conclusion

We have presented a new framework for sampling in distributed settings, demonstrating its application on some illustrative examples in comparison to embarrassingly parallel approaches. Given that our proposed approach makes no additional assumptions on the form of the likelihood beyond its factorised form as in (1), we expect that our algorithm will be most effective in those big data settings for which approximate Gaussianity of the likelihood contributions may not hold. These may include high-dimensional settings, for which some subsets of the data may be relatively uninformative about the parameter. In such cases the likelihood contributions may be highly non-Gaussian, so that the consensus Monte Carlo approach of averaging across chains results in estimates of high bias; simultaneously, the high dimensionality may preclude the use of alternative combination techniques (e.g. the use of kernel density estimates).

This framework may be of use in serial settings. As previously noted, the contemporaneous work of Vono et al. (2019a) presents an example in which the use of an analogous framework (with $b = 1$) results in more efficient simulation than approaches that directly target the true posterior. Our proposed SMC sampler implementation and the associated bias correction technique may equally be applied to such settings,

reducing the need to specify a single value of the regularisation parameter λ.

There is potential for further improvements to be made to the procedures we present here. In the SMC case for example, while our proposed use of weighted least squares as a bias correction technique is simple, non-linear procedures (such those proposed in an ABC context by Blum and François, 2010) may provide a more robust alternative, with some theoretical guarantees. We also stress that the MCMC and SMC procedures presented here constitute only two possible approaches to inference within the instrumental hierarchical model that we propose, and there is considerable scope for alternative sampling algorithms to be employed within this global consensus framework.

SUPPLEMENTARY MATERIAL

Supplement to 'Global Consensus Monte Carlo': "GCMCsupplement.pdf" includes supplementary material covering implementation considerations, theoretical analysis for a simple model, and heuristic procedures for the proposed SMC sampler (with numerical demonstrations).

R code for examples: "GCMCexamples.zip" contains R scripts used to generate the numerical results and figures presented here and in the supplement.

References

Bardenet, R., Doucet, A., and Holmes, C. (2014). Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 405–413.

Beaumont, M. A., Zhang, W., and Balding, D. J. (2002). Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035.

Bertsekas, D. P. and Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall.

Blum, M. G. and François, O. (2010). Non-linear regression models for approximate Bayesian computation. *Statistics and Computing*, 20(1):63–73.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Chan, H. P. and Lai, T. L. (2013). A general theory of particle filters in hidden Markov models and some applications. *Annals of Statistics*, 41(6):2877–2904.

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika*, 89(3):539–552.

Chopin, N. and Ridgway, J. (2017). Leave Pima Indians alone: binary regression as a benchmark for Bayesian computation. *Statistical Science*, 32(1):64–87.

Crooks, G. E. (1998). Nonequilibrium measurements of free energy differences for microscopically reversible Markovian systems. *Journal of Statistical Physics*, 90(5–6):1481–1487.

Dai, H., Pollock, M., and Roberts, G. (2019). Monte Carlo fusion. *Journal of Applied Probability*, 56(1):174–191.

Dean, J. and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436.

Douc, R., Cappé, O., and Moulines, E. (2005). Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pages 64–69. IEEE.

Doucet, A. and Johansen, A. M. (2011). A tutorial on particle filtering and smoothing: Fifteen years later. In Crisan, D. and Rozovskii, B., editors, *Handbook of Nonlinear Filtering*, pages 656–704. Oxford University Press.

Doucet, A. and Lee, A. (2018). Sequential Monte Carlo methods. In Maathuis, M., Drton, M., Lauritzen, S. L., and Wainwright, M., editors, *Handbook of Graphical Models*, pages 165–189. CRC Press.

Finke, A., Doucet, A., and Johansen, A. M. (2020). Limit theorems for sequential MCMC methods. *Advances in Applied Probability*, 52(2):377–403.

Gerber, M., Chopin, N., and Whiteley, N. (2019). Negative association, ordering and convergence of resampling methods. *Annals of Statistics*, 47:2236–2260.

Gilks, W. R. and Berzuini, C. (2001). Following a moving target — Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146.

Huggins, J., Campbell, T., and Broderick, T. (2016). Coresets for scalable Bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pages 4080–4088.

Jasra, A., Stephens, D. A., Doucet, A., and Tsagaris, T. (2011). Inference for Lévy-driven stochastic volatility models via adaptive sequential Monte Carlo. *Scandinavian Journal of Statistics*, 38(1):1–22.

Jordan, M. I., Lee, J. D., and Yang, Y. (2019). Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, 114(526):668–681.

Korattikara, A., Chen, Y., and Welling, M. (2014). Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*.

Lee, A. and Whiteley, N. (2018). Variance estimation in the particle filter. *Biometrika*, 105(3):609–625.

Liu, J. S. and Chen, R. (1995). Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, 90(430):567–576.

Maclaurin, D. and Adams, R. P. (2014). Firefly Monte Carlo: Exact MCMC with subsets of data. In *Proceedings of the 30th International Conference on Uncertainty in Artificial Intelligence (UAI-14)*, pages 543–552.

Marin, J. M., Pudlo, P., Robert, C. P., and Ryder, R. J. (2012). Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180.

Minsker, S., Srivastava, S., Lin, L., and Dunson, D. (2014). Scalable and robust Bayesian inference via the median posterior. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1656–1664.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing*, 11(2):125–139.

Neiswanger, W., Wang, C., and Xing, E. (2014). Asymptotically exact, embarrassingly parallel MCMC. In *Proceedings of the 30th International Conference on Uncertainty in Artificial Intelligence (UAI-14)*, pages 623–632.

Rabinovich, M., Angelino, E., and Jordan, M. I. (2015). Variational consensus Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 1207–1215.

Rendell, L. J. (2020). *Sequential Monte Carlo variance estimators and global consensus.* PhD thesis, University of Warwick.

Rendell, L. J., Johansen, A. M., Lee, A., and Whiteley, N. (2018). Global consensus Monte Carlo. *arXiv preprint arXiv:1807.09288*.

Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various Metropolis–Hastings algorithms. *Statistical Science*, 16(4):351–367.

Scheffé, H. (1947). A useful convergence theorem for probability distributions. *Annals of Mathematical Statistics*, 18(3):434–438.

Scott, S. L. (2017). Comparing consensus Monte Carlo strategies for distributed Bayesian computation. *Brazilian Journal of Probability and Statistics*, 31(4):668–685.

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. *International Journal of Management Science and Engineering Management*, 11(2):78–88.

Srivastava, S., Cevher, V., Dinh, Q., and Dunson, D. (2015). WASP: Scalable Bayes via barycenters of subset posteriors. In *Artificial Intelligence and Statistics*, pages 912–920.

Vats, D., Flegal, J. M., and Jones, G. L. (2019). Multivariate output analysis for Markov chain Monte Carlo. *Biometrika*, 106(2):321–337.

Vono, M., Dobigeon, N., and Chainais, P. (2018). Sparse Bayesian binary logistic regression using the split-and-augmented Gibbs sampler. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing.*

Vono, M., Dobigeon, N., and Chainais, P. (2019a). Split-and-augmented Gibbs sampler – application to large-scale inference problems. *IEEE Transactions on Signal Processing*, 67(6):1648–1661.

Vono, M., Paulin, D., and Doucet, A. (2019b). Efficient MCMC sampling with dimension-free convergence rate using ADMM-type splitting. *arXiv preprint arXiv:1905.11937.*

Wang, X. and Dunson, D. B. (2013). Parallelizing MCMC via Weierstrass sampler. *arXiv preprint arXiv:1312.4605.*

Wang, X., Guo, F., Heller, K. A., and Dunson, D. B. (2015). Parallelizing MCMC with random partition trees. In *Advances in Neural Information Processing Systems*, pages 451–459.

Xu, M., Lakshminarayanan, B., Teh, Y. W., Zhu, J., and Zhang, B. (2014). Distributed Bayesian posterior sampling via moment sharing. In *Advances in Neural Information Processing Systems*, pages 3356–3364.

Zhou, Y., Johansen, A. M., and Aston, J. A. (2016). Towards automatic model comparison: an adaptive sequential Monte Carlo approach. *Journal of Computational and Graphical Statistics*, 25(3):701–726.

Supplement to 'Global Consensus Monte Carlo'

Lewis J. Rendell*
Department of Statistics, University of Warwick
and
Adam M. Johansen
Department of Statistics, University of Warwick
and
Anthony Lee
School of Mathematics, University of Bristol
and
Nick Whiteley
School of Mathematics, University of Bristol

August 11, 2020

S1 Considerations when implementing the Metropolis-within-Gibbs sampler

S1.1 Repeated MCMC kernel iterations

To analyse the effects of communication latency on our approach, we consider an abstracted distributed setting:

- Let ℓ represent the approximate wall-clock time required to compute each $f_j(z)$ for a given $z \in \mathsf{E}$, which we here assume is independent of j for simplicity.

- Let the communication latency be C; for the purposes of this analysis we shall consider the additional time taken due to bandwidth restrictions to be negligible.

- Assume also that the time taken to compute the prior μ, and the global consensus transition densities $K_j^{(\lambda)}$, is negligible.

In an MCMC approach directly targeting the full posterior, each accept/reject step requires communication to and from each node in order to compute the posterior density $\pi(z)$ at the proposed $z \in \mathsf{E}$. Assuming that the likelihood contributions of each block may be computed synchronously, the time taken by each iteration of such an algorithm is therefore approximately $\ell + 2C$.

Within our proposed global consensus approach computations of the full conditional densities of each X_j, and of Z, may each occur on a single node. Suppose that the Markov kernels $P_{j,z}^{(\lambda)}$ comprise k iterations of an MCMC kernel leaving $\pi_\lambda(x_j \mid z)$ invariant. Then, under the same assumptions of synchronous computation, a single iteration of our Metropolis-within-Gibbs algorithm (generating one new value of the Z-chain) requires a time of approximately $k\ell + 2C$.

The consequence is that, while our proposed approach would generally generate fewer samples per unit of wall-clock time, the proportion of time spent on likelihood computation (rather than communication) may be made far greater: $k\ell/(k\ell + 2C)$ for global consensus Monte Carlo, versus $\ell/(\ell + 2C)$ for the 'direct' MCMC approach. This

may be especially important when the latency C is large compared to the likelihood computation time ℓ; by choosing the number of MCMC kernel applications k to be sufficiently large, the resulting sampler will spend a greater proportion of time exploring the state space, and may therefore exhibit faster mixing (with respect to wall-clock time). This approach may be particularly useful for high-dimensional settings, in which constructing a well-mixing MCMC kernel can be difficult.

The 'local' application of MCMC kernels in our framework may also allow a wider range of such kernels to be computationally feasible than in a direct approach. For example, in the case where the state space E is multi-dimensional, one may wish to use 'componentwise' proposals, in which new values are proposed for each individual component (or collection thereof), with all others held fixed. Updating each component in turn may be infeasible in a direct MCMC approach, due to the communication latency involved in computing the acceptance probability for each proposed value. Similarly, our proposed framework may also be beneficial when using adaptive MCMC algorithms in distributed settings. In a direct MCMC approach, adapting the choice of proposal distribution to the target π may be slow (in the sense of wall-clock time) due to the communication required in each accept/reject step. Within the global consensus framework, for which the acceptance probabilities required by the MCMC kernels $P_{j,z}^{(\lambda)}$ may be computed locally, adaptation may occur faster, which may contribute to better mixing.

Our analysis of the Gibbs sampler setting in Section S2 may be informative about the more general Metropolis-within-Gibbs setting, in cases where each $P_{j,z}^{(\lambda)}$ comprises enough MCMC iterations k to exhibit good mixing.

S1.2 Choosing the Markov transition densities

Depending on μ and $f_{1:b}$, appropriate choices of $K_j^{(\lambda)}$ may enable direct sampling from some of the full conditional distributions of Z and $X_{1:b}$. For example within (8), each $K_j^{(\lambda)}(\cdot, x_j)$ is a pseudo-likelihood for $Z \sim \mu$, and so if μ is conjugate to $K_j^{(\lambda)}(\cdot, x_j)$ for each $j \in \{1, \dots, b\}$, then the conditional distribution of Z given $X_{1:b}$ will be from the

same family as μ. Similarly, one might choose for each $K_j^{(\lambda)}(z, \cdot)$ a conjugate prior for the partial likelihood terms f_j, so that the conditional distribution (7) of each X_j given $\mathbf{y}_j$ and Z is from the same family as $K_j^{(\lambda)}(z, \cdot)$.

It may also be appropriate to choose the Markov transition densities to have relative scales comparable to those of the corresponding partial likelihood terms. To motivate this consider a univariate setting in which the partial likelihood terms are Gaussian, so that we may write $f_j(z) = \mathcal{N}(\mu_j; z, \sigma_j^2)$ for each $j \in \{1, \ldots, b\}$. Suppose one uses Gaussian transition densities $K_j^{(\lambda)}(z, x) = \mathcal{N}(x; z, c_j \lambda)$, where $c_1, \ldots, c_b$ are positive values controlling the relative strengths of association between Z and the local variables $X_{1:b}$. As seen in (4), in the approximating density π_λ the partial likelihood terms f_j are replaced by smoothed terms (3), in this case given by

$$f_j^{(\lambda)}(z) \propto \mathcal{N}(\mu_j; z, \sigma_j^2 + c_j \lambda). \tag{S1}$$

The resulting smoothed posterior density is presented as (S2) in Section S2, where this setting is further explored.

In this case, the role of λ may be seen as 'diluting' or downweighting the contribution of each partial likelihood to the posterior distribution π_λ. A natural choice is to take $c_j \propto \sigma_j^2$, so that the dilution of each f_j is in proportion to the strength of its contribution to π. In this case (S1) becomes

$$f_j^{(\lambda)}(z) \propto \mathcal{N}(\mu_j; z, (1 + c\lambda)\sigma_j^2)$$

for some constant c. The relative strengths of contribution of the $f_{1:b}$ are thereby preserved in the posterior density π_λ.

A particular case of interest in that in which the blocks of data $\mathbf{y}_j$ differ in size. If each observation y_l has a likelihood contribution of the form $\mathcal{N}(y_l; z, \sigma^2)$, then the jth partial likelihood may be expressed as $f_j(z) \propto \mathcal{N}(\bar{y}_j; z, \sigma^2/n_j)$, where n_j is the number of data in the jth block and $\bar{y}_j$ is their mean. Taking $c_j \propto 1/n_j$, the smoothed partial likelihood (S1) becomes

$$f_j^{(\lambda)}(z) \propto \mathcal{N}(\bar{y}_j; z, (\sigma^2 + c\lambda)/n_j)$$

4

for some c, so that the information from each observation is diluted in a consistent way. Motivated by Bayesian asymptotic arguments, we suggest that this scaling of the regularisation parameter in inverse proportion to the relative block sizes may be beneficial in more general settings.

The effect of such choices on the MCMC algorithm is most readily seen by considering the improper uniform prior $\mu(z) \propto 1$ (a Gaussian prior is considered in Section S2). Taking $K_j^{(\lambda)}(z, x) = \mathcal{N}(x; z, c_j \lambda)$, the conditional density of Z given $X_{1:b}$ is

$$\tilde{\pi}_\lambda(z \mid x_{1:b}) = \mathcal{N}\left(z; \frac{\sum_{j=1}^{b} x_j/c_j}{\sum_{j=1}^{b} 1/c_j}, \frac{\lambda}{\sum_{j=1}^{b} 1/c_j}\right).$$

Therefore, when updating Z given the local variables' current values $x_{1:b}$, the choice of $c_{1:b}$ dictates the relative influence of each such value. For example, we might expect the local variables corresponding to larger blocks to be more informative about the distribution of Z, which further justifies choosing $c_{1:b}$ to be inversely proportional to the block sizes.

In a multidimensional setting, one could control the covariance structure of each X_j given Z by using transition densities of the form $\mathcal{N}(x; z, \lambda \Psi_j)$, where $\Psi_{1:b}$ are positive semi-definite matrices. By a similar Gaussian analysis, one could preserve the relative strengths of contribution of the partial likelihood terms by choosing for each Ψ_j an approximation of the covariance matrix of f_j.

S2   Theoretical analysis for a simple model

S2.1   Inferring the mean of a normal distribution

To study the theoretical properties of our algorithm, we here consider a simple model where the goal is to infer the mean of a normal distribution. While our approach does not require the distribution π to be approximately Gaussian, the behaviour of our algorithm in this simple setting is particularly amenable to analysis. The results here may also be indicative of performance for regular models with abundant data due to the Bernstein–von Mises theorem (see, e.g., van der Vaart, 2000, Chapter 10).

Let $\mu(z) = \mathcal{N}(z; \mu_0, \sigma_0^2)$, and for each $j \in \{1, \ldots, b\}$ let $f_j(z) = \mathcal{N}(\mu_j; z, \sigma_j^2)$ and $K_j^{(\lambda)}(z, x) = \mathcal{N}(x; z, c_j\lambda)$, following Section S1.2. We obtain

$$\pi_\lambda(z) = \mathcal{N}\left(z; \delta_\lambda^2\left[\frac{\mu_0}{\sigma_0^2} + \sum_{j=1}^b \frac{\mu_j}{\sigma_j^2 + c_j\lambda}\right], \delta_\lambda^2\right), \quad \delta_\lambda^2 = \left(\frac{1}{\sigma_0^2} + \sum_{j=1}^b \frac{1}{\sigma_j^2 + c_j\lambda}\right)^{-1}, \quad \text{(S2)}$$

and $\pi(z)$ can be recovered by taking $\lambda = 0$ in (S2). The corresponding full conditional densities for (S2) are

$$\tilde{\pi}_\lambda(x_j \mid z) = \mathcal{N}\left(x_j; \frac{\sigma_j^2 z + c_j\lambda\mu_j}{\sigma_j^2 + c_j\lambda}, \frac{c_j\lambda\sigma_j^2}{\sigma_j^2 + c_j\lambda}\right)$$

and

$$\tilde{\pi}_\lambda(z \mid x_{1:b}) = \mathcal{N}\left(z; \tilde{\delta}_\lambda^2\left[\frac{\mu_0}{\sigma_0^2} + \sum_{j=1}^b \frac{x_j}{c_j\lambda}\right], \tilde{\delta}_\lambda^2\right), \quad \tilde{\delta}_\lambda^2 = \left(\frac{1}{\sigma_0^2} + \sum_{j=1}^b \frac{1}{c_j\lambda}\right)^{-1},$$

and we consider the case where $M_1^{(\lambda)}$ and $M_2^{(\lambda)}$ as defined in (9)–(10) are Gibbs kernels. That is, we consider the case in which we draw samples exactly from the full conditional distributions (7)–(8). We choose this setting to facilitate analysis of the resulting Markov chain, but these results may be informative about more general Metropolis-within-Gibbs settings in which well-mixing Markov kernels are used (e.g. those comprising multiple MCMC kernel iterations, as described in Section S1.1).

Since $X_{1:b}$ are conditionally independent given Z and can therefore be updated simultaneously given Z, Algorithm 1 may be viewed analytically as a Gibbs sampler on two variables: Z and $X_{1:b}$. For any two-variable Gibbs Markov chain, each of the two 'marginal' chains (the sequences of states for each of the two variables) is also a Markov chain. In this setting we may therefore consider the Z-chain with transition kernel

$$M_{12}^{(\lambda)}(z, dz') = \left[\int_{\mathsf{E}^b} \tilde{\pi}_\lambda(z' \mid x_{1:b}) \prod_{j=1}^b \tilde{\pi}_\lambda(x_j \mid z) \, dx_{1:b}\right] dz'. \quad \text{(S3)}$$

Observing that $\tilde{\pi}_\lambda(z' \mid x_{1:b})$ depends on $x_{1:b}$ only through the sum $\sum_{j=1}^b x_j/c_j$, one can thereby show that the Z-chain defined by (S3) is an AR(1) process. Specifically, we have

$$Z^i = C + \alpha Z^{i-1} + \epsilon_i, \qquad k > 0,$$

6

where

$$\alpha := \tilde{\delta}_\lambda^2 \sum_{j=1}^b \frac{\sigma_j^2}{c_j \lambda (\sigma_j^2 + c_j \lambda)}, \qquad C := \tilde{\delta}_\lambda^2 \left(\frac{\mu_0}{\sigma_0^2} + \sum_{j=1}^b \frac{\mu_j}{\sigma_j^2 + c_j \lambda} \right),$$

and the ϵ_i are i.i.d. zero-mean normal random variables, with variance

$$\tilde{\delta}_\lambda^2 \left(1 + \tilde{\delta}_\lambda^2 \sum_{j=1}^b \frac{\sigma_j^2}{c_j \lambda (\sigma_j^2 + c_j \lambda)} \right).$$

It follows that the autocorrelation of lag k is given by α^k for $k \geq 0$, and that $\alpha \to 1$ as $\lambda \to 0$.

S2.2   Asymptotic bias and variance with n observations

We now consider the setting of Section S2.1, making the number of observations n explicit. In particular, for some $z^* \in \mathbb{R}$ consider realisations $y_{1:n}$ of i.i.d. $\mathcal{N}(z^*, \sigma^2)$ random variables, grouped into b blocks. For simplicity, assume that b divides n, that each block contains n/b observations, and that the observations are allocated to the blocks sequentially, so that the jth block comprises those y_l where $l \in B_j :=$ $\{(j-1)n/b + 1, \ldots, jn/b\}$. Then

$$f_j(z) = \prod_{l \in B_j} \mathcal{N}(y_l; z, \sigma^2) \propto \mathcal{N}\left(\frac{b}{n} \sum_{l \in B_j} y_l; z, \frac{b}{n} \sigma^2 \right).$$

Since the blocks are of equal size in this case, so that each partial likelihood is of the same scale, we consider using $K_j^{(\lambda)}(z, x) = \mathcal{N}(x; z, \lambda)$ for each j. From (S2), we obtain

$$\pi_\lambda(z) = \mathcal{N}\left(z; \delta_\lambda^2 \left[\frac{\mu_0}{\sigma_0^2} + \frac{n\bar{y}}{\sigma^2 + n\lambda/b} \right], \delta_\lambda^2 \right), \quad \delta_\lambda^2 = \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2 + n\lambda/b} \right)^{-1}. \qquad (S4)$$

Denoting the identity function on $\mathbb{R}$ by Id, we consider an estimator $\pi_\lambda^N(\mathrm{Id})$ of the posterior first moment $\pi(\mathrm{Id})$, as formed according to (6). We analyse its mean squared error using the bias–variance decomposition (11). The bias is

$$\pi_\lambda(\mathrm{Id}) - \pi(\mathrm{Id}) = \frac{n^2 (\lambda/b) \sigma_0^2 (\mu_0 - \bar{y})}{(\sigma^2 + n\sigma_0^2)(\sigma^2 + n\sigma_0^2 + n\lambda/b)}. \qquad (S5)$$

To assess the variance of $\pi_\lambda^N(\mathrm{Id})$, we consider the associated *asymptotic* variance,

$$\lim_{N \to \infty} N \operatorname{var}(\pi_\lambda^N(\varphi)) = \operatorname{var}(\varphi(Z_0)) \left[1 + 2 \sum_{k=1}^\infty \operatorname{corr}(\varphi(Z_0), \varphi(Z_k)) \right], \qquad Z_0 \sim \pi_\lambda, \quad (S6)$$

for φ square-integrable w.r.t. π_λ. As discussed earlier the Z-chain is an AR(1) process, and the autocorrelations are entirely determined by the autoregressive parameter

$$\alpha = \frac{n\sigma^2\sigma_0^2}{(\sigma^2 + n\lambda/b)(n\sigma_0^2 + n\lambda/b)},$$

from which one can find that the asymptotic variance for $\varphi = \mathrm{Id}$ is

$$\frac{\sigma_0^2(\sigma^2 + n\lambda/b)\left[(n\lambda/b)^2 + (\sigma^2 + n\sigma_0^2)(n\lambda/b) + 2n\sigma^2\sigma_0^2\right]}{(n\lambda/b)(\sigma^2 + n\sigma_0^2 + n\lambda/b)^2}. \tag{S7}$$

Following the definition (S6) of this asymptotic variance, dividing this expression by N gives an approximation of the variance term in (11) for large N.

As a caveat to this and the following analysis, estimation of the mean in Gaussian settings may not accurately reflect what happens in more complex settings. For example, if one uses an improper uniform prior then $\pi_\lambda(\mathrm{Id})$ is equal to $\pi(\mathrm{Id})$ for all λ, as seen in (S5) with $\sigma_0^2 \to \infty$; this will not be true in general.

One may also note that in this Gaussian setting, the variance of π_λ will always exceed the variance of the true target π, since the variance expression in (S4) is an increasing function of λ. The effect is that estimation of the posterior variance in Gaussian settings is likely to result in positive bias, and confidence intervals for $\pi(\mathrm{Id})$ may be conservative. This, of course, simply reflects the fact that marginally the instrumental model can be viewed as replacing the original likelihood with a smoothed version as shown in (4).

S2.3 Asymptotic optimisation of λ for large N

For fixed n, we consider the problem of choosing λ as a function of chain length N so as to minimise the mean squared error of the posterior mean estimator. This involves considering the contributions of the bias and variance to the mean squared error (11) in light of (S5) and (S7). Intuitively, with larger values of N, smaller values of λ can be used to reduce the bias while keeping the variance small. Defining $B(\lambda)$ to be the bias as given in (S5), we see that as $\lambda \to 0$,

$$\frac{B(\lambda)}{\lambda} \to \frac{n^2\sigma_0^2(\mu_0 - \bar{y})}{b(\sigma^2 + n\sigma_0^2)^2} =: B_\star.$$

Similarly, denoting by $V(\lambda)$ the asymptotic variance (S7), we see that

$$\lambda V(\lambda) \to \frac{2b\sigma^4\sigma_0^4}{(\sigma^2 + n\sigma_0^2)^2} =: V_\star.$$

For small λ, the mean squared error of the estimate is given approximately by

$$\mathbb{E}\left[\left(\pi_\lambda^N(\mathrm{Id}) - \pi(\mathrm{Id})\right)^2\right] \approx (\lambda B_\star)^2 + \frac{1}{N}\frac{V_\star}{\lambda},$$

which may be shown to be minimised when

$$\lambda^3 = \frac{V_\star}{2B_\star^2 N} = \frac{b^3\sigma^4\left(\sigma^2 + n\sigma_0^2\right)^2}{n^4 N\left(\mu_0 - \bar{y}\right)^2}.$$

We see that, for a fixed number of data n, we should scale λ with the number of samples N as $\mathcal{O}(N^{-1/3})$. The corresponding minimal mean squared error, in which the contribution of the variance is twice that of the squared bias, may be shown to behave as $\mathcal{O}(N^{-2/3})$.

Note that in this example, all dependence on λ and b in the smoothed likelihood (S4) is through their ratio λ/b. The result is that splitting the data into more blocks has the same effect as reducing λ, and so these results may be adapted to consider optimisation of the ratio λ/b. This relationship may not be representative of these variables' behaviour in other models; but in cases where Bernstein–von Mises arguments hold, such results may be useful in settings where the number of blocks b may be chosen by the practitioner.

S2.4 Posterior consistency and coverage of credible intervals as $n \to \infty$

We now consider the behaviour of the algorithm as the number of data n tends to infinity. Recalling that we assume the true parameter value to be $z^* \in \mathbb{R}$, we may consider the consistency of the posterior distribution (S4) by treating the data $Y_{1:n}$ as random. We denote their mean by $\bar{Y}_n$, which is normally distributed with mean z^* and variance σ^2/n. We shall also consider allowing λ and b to vary with n; making this

explicit in the notation, (S4) becomes

$$\pi_{\lambda_n}(z) = \mathcal{N}\left(z; \mu_{(n)}, \delta^2_{(n)}\right) = \mathcal{N}\left(z; \delta^2_{(n)}\xi_{(n)}, \delta^2_{(n)}\right),$$

where $\mu_{(n)} = \delta^2_{(n)}\xi_{(n)}$, and

$$\xi_{(n)} = \frac{\mu_0}{\sigma_0^2} + \frac{n\bar{Y}_n}{\sigma^2 + n\lambda_n/b_n}, \qquad \delta^2_{(n)} = \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2 + n\lambda_n/b_n}\right)^{-1}.$$

We consider $\lambda_n/b_n = cn^{-\gamma}$ and using the fact that $\bar{Y}_n \overset{\text{a.s.}}{\to} z^*$, we obtain the following convergence results for different values of γ:

$\gamma < 0$	$\delta^2_{(n)} \to \sigma_0^2$	$\xi_{(n)} \overset{\text{a.s.}}{\to} \frac{\mu_0}{\sigma_0^2}$		$\mu_{(n)} \overset{\text{a.s.}}{\to} \mu_0$
$\gamma = 0$	$\delta^2_{(n)} \to \left(\frac{1}{\sigma_0^2} + \frac{1}{c}\right)^{-1}$	$\xi_{(n)} \overset{\text{a.s.}}{\to} \frac{\mu_0}{\sigma_0^2} + \frac{z^*}{c}$	$\mu_{(n)} \overset{\text{a.s.}}{\to} \left(\frac{1}{\sigma_0^2} + \frac{1}{c}\right)^{-1}\left(\frac{\mu_0}{\sigma_0^2} + \frac{z^*}{c}\right)$	
$\gamma \in (0,1)$	$n^\gamma\delta^2_{(n)} \to c$	$n^{-\gamma}\xi_{(n)} \overset{\text{a.s.}}{\to} \frac{z^*}{c}$		$\mu_{(n)} \overset{\text{a.s.}}{\to} z^*$
$\gamma = 1$	$n\delta^2_{(n)} \to \sigma^2 + c$	$n^{-1}\xi_{(n)} \overset{\text{a.s.}}{\to} \frac{z^*}{\sigma^2+c}$		$\mu_{(n)} \overset{\text{a.s.}}{\to} z^*$
$\gamma > 1$	$n\delta^2_{(n)} \to \sigma^2$	$n^{-1}\xi_{(n)} \overset{\text{a.s.}}{\to} \frac{z^*}{\sigma^2}$		$\mu_{(n)} \overset{\text{a.s.}}{\to} z^*$

Hence, it can be seen that the posterior is consistent (see, e.g., Ghosh and Ramamoorthi, 2003, Chapter 1) if $\gamma > 0$. Moreover, if $\gamma > 1$ then $1 - \alpha$ credible intervals will have asymptotically a coverage probability of exactly $1-\alpha$ due to the convergence $n\delta^2_{(n)} \to \sigma^2$.

If $\gamma \in (0,1)$ then the rate of approximate posterior contraction is too conservative, while if $\gamma = 1$ the corresponding credible intervals will be too wide by a constant factor depending on c. From a practical perspective, one can consider the case where n/b corresponds to the maximum number of data points that can be processed on an individual computing node. In such a setting, letting $b_n \propto n$ is reasonable and we require in addition that λ_n is decreasing to obtain credible intervals with asymptotically exact coverage.

S3 Heuristic procedures for the sequential Monte Carlo implementation

S3.1 Determining a subset of estimates to use for linear regression

If the local linear regression approach for bias correction is used, then the practitioner must determine a value of λ below which the dependence of $\pi_\lambda(\varphi)$ on λ is approximately linear. For this purpose, we propose a heuristic based on the coefficient of determination, commonly denoted R^2; here, this may be thought of as the proportion of the variance of the observed values of $\pi_\lambda^N(\varphi)$ that is explained by an assumed linear dependence on λ. To define this explicitly, consider the weighted least squares fit for which (13) is the resulting bias-corrected estimate. Extending the notation used therein, let $\hat{\eta}_p^S$ denote the predicted value of η_p under the model. Then the coefficient of determination R_S^2 for this weighted least squares model fit may be computed as the ratio of the weighted sum of squared errors and the weighted total sum of squares. That is,

$$R_S^2 := \frac{\sum_{p \in S}(\hat{\eta}_p^S - \tilde{\eta}_S)^2 / v_p}{\sum_{p \in S}(\eta_p - \tilde{\eta}_S)^2 / v_p} = 1 - \frac{\sum_{p \in S}(\eta_p - \hat{\eta}_p^S)^2 / v_p}{\sum_{p \in S}(\eta_p - \tilde{\eta}_S)^2 / v_p}. \tag{S8}$$

The heuristic procedure for determining such a subset of the estimates is presented in Algorithm S1. After completion of the SMC sampler described in Algorithm 2, one conducts weighted least squares in the manner described in Section 3.1, including all values of λ_p and the corresponding SMC estimates $\pi_{\lambda_p}^N(\varphi)$, and computing the R^2 value for the resulting fit. One then re-conducts the regression, without the observation in the subset corresponding to the largest λ value. If this results in a greater R^2 value, this observation should henceforth be excluded from the least squares regression. One continues to apply this procedure, each time repeating the regression without the observation corresponding to the highest remaining value of λ, until doing so no longer results in a model with a greater R^2 value than the current fit. The regression fit at this point may then be used to compute the bias-corrected estimate for $\pi(\varphi)$, so that

Algorithm S1 Linear regression inclusion procedure for SMC bias-correction

Fix a decreasing sequence $(\lambda_0, \lambda_1, \ldots, \lambda_n)$ and a test function φ.

Set number of particles N.

1. Complete Algorithm 2, generating and storing estimates $\eta_p := \pi_{\lambda_p}^N(\varphi)$ using the particle approximations (12), and estimates v_p of their variances, for $p \in \{0, \ldots, n\}$.

2. Initialise set of indices of estimates to be used in regression as $S \leftarrow \{0, \ldots, n\}$.

3. Regress η_p against λ_p using weighted least squares, with weights $1/v_p$, for $p \in S$. Compute the coefficient of determination R_S^2 according to (S8).

   (a) If $|S| \leq 3$, proceed to Step 4. Otherwise, set $S' \leftarrow S \setminus \{\min(S)\}$, and regress η_p against λ_p using weighted least squares, with weights $1/v_p$, for $p \in S'$. Compute $R_{S'}^2$ according to (S8).

   (b) If $R_{S'}^2 > R_S^2$, set $S \leftarrow S'$, and return to Step 3a. Otherwise, proceed to Step 4.

4. Return the bias-corrected estimate $\pi_S^{\mathrm{BC}}(\varphi)$, computed according to (13).

in (13), S corresponds to the set of indices of the remaining λ values.

The motivation for this approach is that if this largest λ value is not sufficiently close to zero for $\pi_\lambda(\varphi)$ to be approximately linear in λ, then retaining the corresponding SMC estimate in the regression may result in a large proportion of the variance in the data being unexplained by a linear dependence. By excluding the corresponding SMC estimate, one would expect the linear fit applied to the remaining estimates to better describe their variance, and therefore to have a greater R^2 value.

This heuristic approach has a natural online implementation, allowing a bias-corrected estimate to be computed after each step of the algorithm; we use this form within our proposed stopping rule in Section S3.2, for which it forms Step 2 in each iteration of Algorithm S2. Specifically, we maintain a set of the SMC estimates to be used in the regression (and the corresponding values of λ), initialising this to be empty. After the pth step of the SMC sampler, the newly-generated SMC estimate $\pi_{\lambda_p}^N(\varphi)$ is added to this set (with the corresponding λ_p). One conducts weighted least squares on this set of estimates and then, as long as the set contains more than 3 estimates, one proceeds in the manner described above, re-conducting the regression without the observation in the set corresponding to the highest value of λ. If this results in a fit with a higher R^2 value, then the omitted SMC estimate is henceforth excluded from the set used for regression, and this step is repeated. If not, then one terminates this procedure and proceeds to the next step of the SMC sampler.

S3.2   Stopping rule

As λ approaches zero we expect the bias resulting from estimating $\pi(\varphi)$ by $\pi_\lambda^N(\varphi)$ to decrease, while the variance of the resulting estimators may increase due to poorer mixing of the associated Markov kernels. Based on the bias–variance decomposition (11) of the mean squared error, we here propose a procedure for determining when to terminate the SMC sampler, in order to achieve such a bias–variance trade-off.

At each stage, having computed an updated bias-corrected estimate via the online

procedure described in Section S3.1, one may subtract this value from each of the SMC estimates generated so far in order to produce an estimate of the bias in each case. As discussed in Section 3.1, we also have an estimate of the variance of each SMC estimate, as used in the weighted linear regression procedure. As such, at each stage we are able to estimate the mean squared error of each SMC estimate so far generated, by squaring each estimate of the bias and adding the appropriate estimate of the variance.

The formation of these mean squared error estimates is based on, but does not exactly correspond to, the bias–variance decomposition (11). For example, the particle-based SMC estimates $\pi_\lambda^N(\varphi)$ are not unbiased as estimators of $\pi(\varphi)$, although they are consistent in the number of particles N (Beskos et al., 2016, Section 3). Furthermore, the bias-corrected estimate itself is not unbiased, since it is formed based on approximate local linearity rather than a true linear dependency of $\pi_\lambda^N(\varphi)$ on λ. Nonetheless, the use of this heuristic approach in our proposed stopping rule has been found to work well in practice, resulting in estimates of low mean squared error; we discuss one such example in Section S4.1.

Note that, since the bias-corrected estimate of $\pi(\varphi)$ is updated after each step (to take into account the most recent estimate), the estimated mean squared errors of all previous estimates may also all be updated after each step. After each SMC estimate is generated we may therefore determine which SMC estimate, of all those generated so far, has the lowest estimated mean squared error. We propose that, for some κ, the SMC sampler should be terminated after the same previous estimate is found to have the lowest mean squared error of all those generated so far, for κ consecutive iterations. Following the termination of the algorithm via the stopping rule this SMC estimate, which has been repeatedly found to have the lowest estimated MSE, may be returned as the final estimate for $\pi(\varphi)$.

This approach is described in Algorithm S2. In our simulation studies, we found that taking $\kappa = 15$ worked well in balancing robustness with the computational complexity of the resulting algorithm.

As an alternative, one may choose to return the final bias-corrected estimate, for

Algorithm S2 Global consensus Monte Carlo: SMC algorithm with stopping rule

Fix a decreasing sequence $(\lambda_0, \lambda_1, \ldots, \lambda_n)$ and a test function φ.

Set number of particles N and stopping rule parameter κ.

Initialise set of indices of estimates to be used in regression as $S \leftarrow \varnothing$.

For $p = 0, 1, \ldots$ until termination:

1. Complete the pth step of Algorithm 2, generating and storing an estimate $\eta_p :=$ $\pi_{\lambda_p}^N(\varphi)$ using the particle approximation (12), and an estimate v_p of its variance.

2. Set $S \leftarrow S \cup \{p\}$. If $|S| > 1$:

   (a) Regress η_q against λ_q using weighted least squares, with weights $1/v_q$, for $q \in S$. Compute the coefficient of determination R_S^2 according to (S8).

   (b) If $|S| \leq 3$, proceed to Step 3. Otherwise, set $S' \leftarrow S \setminus \{\min(S)\}$, and regress η_q against λ_q using weighted least squares, with weights $1/v_q$, for $q \in S'$. Compute $R_{S'}^2$ according to (S8).

   (c) If $R_{S'}^2 > R_S^2$, set $S \leftarrow S'$, and return to Step 2b. Otherwise, proceed to Step 3.

3. Set $m_p \leftarrow \pi_S^{\mathrm{BC}}(\varphi)$, a bias-corrected estimate computed according to (13).

4. Set
$$i_p \leftarrow \arg\min_{0 \leq q \leq p} \left[(\eta_q - m_p)^2 + v_q \right],$$
which corresponds to taking the index of the SMC estimate with the lowest estimated mean squared error.

5. If $p \geq \kappa - 1$, and $(i_{p-\kappa+1}, \ldots, i_p)$ are all equal, terminate the algorithm, returning the estimate η_{i_p} of lowest estimated MSE (and/or m_p, the final bias-corrected estimate).

which this approach also provides a justifiable stopping rule: repeatedly finding that the same previous estimate has the lowest MSE suggests stability in our estimates of the MSEs of each previous $\pi_\lambda^N(\varphi)$, and therefore in the bias-corrected estimate. Furthermore, since we expect the biases of the estimates $\pi_\lambda^N(\varphi)$ to decrease as λ approaches zero, repeatedly finding that a previous SMC estimate has the lowest MSE suggests that more recent estimates are of higher variances. Again, this is also indicative of a stabilisation of the bias-corrected estimate, since new observations are included in the regression-based bias correction procedure with weights inversely proportional to these variances.

S4 Additional examples

S4.1 SMC bias correction: Gaussian example

To demonstrate the SMC bias correction technique described in Section 3.1, we consider a univariate Gaussian model of the form described in Section S2, with the aim of estimating the posterior first moment $\pi(\mathrm{Id})$. We consider a case with $b = 32$, taking $f_j(z) = \mathcal{N}(\mu_j; z, 1)$ for $j \in \{1, \ldots, b\}$, with the values μ_j drawn independently from a normal distribution with mean 4 and variance 1. For the Markov transition kernels we use $K_j^{(\lambda)}(z, x) = \mathcal{N}(x; z, \lambda)$. For the purposes of illustrating the local linear regression approach to bias correction we consider the (quite concentrated) prior density $\mu(z) = \mathcal{N}(z; 4, 1)$. In this case, we see that the dependence of $\pi_\lambda(\mathrm{Id})$ on λ is highly non-linear on the range $0 < \lambda < 1000$ (see Figure S1a).

We constructed an SMC sampler using $N = 2500$ particles; we used sequences of λ values beginning with $\lambda_0 = 1000$, with subsequent values determined adaptively according to the procedure proposed by Zhou et al. (2016), for which we used parameter $\mathrm{CESS}^\star = 0.95$. For the purposes of illustrating the bias correction technique we here consider sequences of λ values of fixed length $n = 200$; we will describe the use of the proposed stopping rule subsequently.
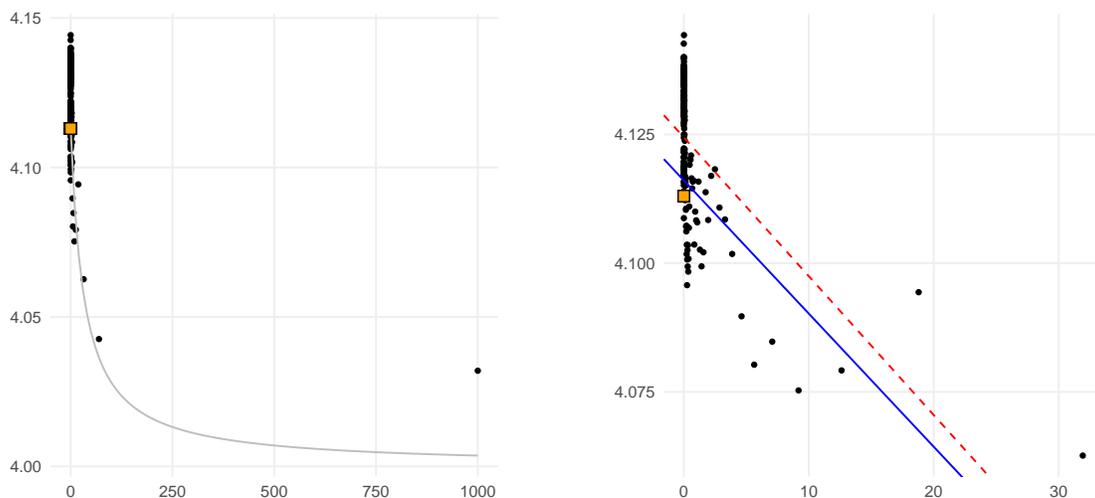
(a) All estimates $\pi_\lambda^N(\text{Id})$, and the true value of $\pi_\lambda(\text{Id})$ as a function of λ (solid grey line).

(b) Subset of estimates used for local linear regression, and linear fits using least squares: weighted (solid blue line) and unweighted (dashed red line).

Figure S1: Estimates $\pi_\lambda^N(\text{Id})$ plotted against λ, as obtained at each step of a single run of the SMC sampler for the Gaussian toy example. The orange square indicates the true value of $\pi(\text{Id}) \approx 4.113$.

To construct Markov kernels invariant with respect to each distribution $\tilde{\pi}_\lambda$, we used Gibbs kernels constructed in the manner of Algorithm 1. That is to say that in each time step of the SMC sampler (i.e. for each value of λ) and for each particle, each of $X_{1:b}$ was updated by drawing exactly from its conditional distribution, after which Z was updated similarly.

Figure S1a shows the SMC estimate $\pi_\lambda^N(\text{Id})$ obtained for each λ, in a single run of this algorithm. To determine a subset of these estimates to be used for local linear regression, we used the approach described in Section S3.1; this subset is displayed in Figure S1b. In this case, we see that for the smallest values of λ considered, the estimates exhibit increased variance, due to the poorer mixing of the Markov kernels, and the degeneracy of the particle set.
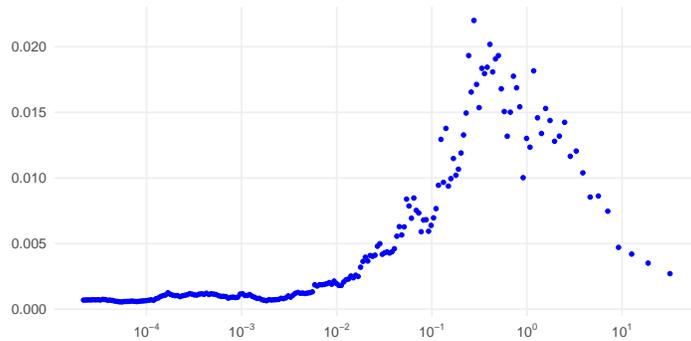
Figure S2: For the estimates $\pi_\lambda^N(\mathrm{Id})$ plotted in Figure S1b, the estimates' relative weights as used in the weighted least squares bias correction technique, plotted against λ on a logarithmic scale.

As described in Section 3.2, when conducting local least squared regression we weight each estimate in inverse proportion to its estimated (asymptotic) variance. For the estimates plotted in Figure S1b, these relative weights are presented in Figure S2, with λ on a log scale for clarity. The resulting weighted least squares fit is overplotted in Figure S1b, together with the corresponding *unweighted* (ordinary) least squares fit. We see that for these results, the weighted least squares fit better reflects the local linear dependence on λ, being less influenced by the high-variance estimates near 0, which correspondingly carry less weight in the regression.

As discussed in Section 3, we may view the SMC sampler as a method to improve or 'refine' the estimator that would be formed using the initial set of particles, i.e. $\pi_{\lambda_0}^N(\mathrm{Id})$, where $\lambda_0 = 1000$. A straightforward choice of such a refined estimator would therefore be the SMC estimate $\pi_{\lambda_n}^N(\mathrm{Id})$ corresponding to λ_n, the final (smallest) λ value considered. We ran the SMC sampler 25 times; the value of λ_n varied between runs due to the adaptive specification of the sequence of distributions, but each time was approximately 2.2×10^{-5}. The mean squared error of the resulting estimate, computed over the 25 replicates, was 1.13×10^{-3}. This is rather lower than the MSE of the estimator $\pi_{\lambda_0}^N(\mathrm{Id})$ corresponding to the initial value $\lambda_0 = 1000$, which was 1.32×10^{-2}.

For each run of the SMC sampler we also computed a bias-corrected estimate (13) of

18

$\pi(\mathrm{Id})$ using weighted least squares as described above; that is, the intercept of the local least squares linear fit. The mean squared error of this bias-corrected estimator was 3.60×10^{-5}, rather lower MSE than the simpler approach of considering solely the final λ value. Additionally, for purposes of comparison we also computed a bias-corrected estimate using ordinary (unweighted) least squares. This resulted in a mean squared error of 2.57×10^{-4}, between the values from the other approaches.

We subsequently considered the effects of using the stopping rule proposed in Section S3.2, retrospectively applying the procedure of Algorithm S2 with $\kappa = 15$ to each of the 25 simulations. On average, termination occurred after 53.0 SMC iterations (i.e. values of λ following the initial λ_0). The SMC estimate chosen by the stopping rule, estimated to have the lowest mean squared error, was found to have an MSE of 1.11×10^{-5}, lower than the values for which $n = 200$ iterations were used. The final bias-corrected estimate at the time of termination was found to have a comparably low MSE of 9.23×10^{-6}.

We see therefore that using the stopping rule to choose the SMC estimate of lowest estimated MSE here results in superior estimator to that obtained by simply taking the final SMC estimate after a fixed number of iterations, representing a significant refinement of the estimator formed using the initial particle set. We also find that the bias-corrected estimate here performs slightly better than the corresponding estimate obtained after using 200 iterations: in that case, the SMC estimates corresponding to the very smallest values of λ are of high variance and can distort the regression, despite being appropriately weighted.

S4.2 Log-normal example

We present an additional example based on the log-normal model presented in Section 4.1 of the main manuscript. Again, we compare Global Consensus Monte Carlo (GCMC), using the Gibbs sampler form of Algorithm 1, with three embarrassingly parallel methods based on subposterior sampling: the Consensus Monte Carlo (CMC)

method of Scott et al. (2016), the nonparametric density product estimation (NDPE) approach of Neiswanger et al. (2014), and the Weierstrass rejection sampling (WRS) combination technique of Wang and Dunson (2013).

We generated a data set comprising $b = 32$ blocks, each containing 10^4 data. Within the jth block, the data were generated as i.i.d. observations of a log-normal random variable with parameters $(\mu_j, 1)$; the parameters μ_j were drawn independently from a normal distribution with mean 0 and variance 10^{-2}. We took $f_j(z) = \mathcal{LN}(\bar{y}_j; \log(z), 10^{-4})$, with each $\bar{y}_j$ being the geometric mean of the observations in the jth block; we used the same prior $\mu(z) = \mathcal{LN}(z; 0, 25)$ as previously. While this represents a misspecified model, it is useful in exemplifying the behaviour of global consensus Monte Carlo in cases where there are differences between the blocks of data.

Table S1 shows the estimates of $\int z\pi(z)\,\mathrm{d}z$ and $\int \log(z)\pi(z)\,\mathrm{d}z$, from 25 runs in each algorithmic setting. Global consensus Monte Carlo produces low-bias estimates for a range of λ values. In contrast, the embarrassingly parallel methods result in somewhat larger biases; this is particularly the case for the expected value of the logarithm in the cases of CMC and WRS, which behave similarly on this example. The NDPE method, which is based on kernel density estimation, works reasonably well for this univariate model.

When the data are first randomly permuted and repartitioned into 32 new blocks, the performances of the embarrassingly parallel methods are improved, though we still find that for appropriately-chosen λ, GCMC estimators attain a lower mean squared error. Furthermore, for large distributed data sets permutation of the data in this manner may not be feasible, for example if security restrictions prevent the transfer of data between machines.

References

Beskos, A., Jasra, A., Kantas, N., and Thiery, A. (2016). On the convergence of adaptive sequential Monte Carlo methods. *Annals of Applied Probability*, 26(2):1111–1146.

| | | $\varphi(z) = z$ | $\varphi(z) = \log(z)$ |
Algorithm		$(\pi(\varphi) = 1.01139)$	$(\pi(\varphi) = 0.01132)$
GCMC	$\lambda = 10^1$	1.18022 ± 0.002097	0.01121 ± 0.001537
	$\lambda = 10^0$	1.02740 ± 0.000609	0.01142 ± 0.000623
	$\lambda = 10^{-1}$	1.01305 ± 0.000156	0.01140 ± 0.000154
	$\lambda = 10^{-2}$	1.01155 ± 0.000067	0.01133 ± 0.000066
	$\lambda = 10^{-3}$	1.01140 ± 0.000020	0.01132 ± 0.000020
	$\lambda = 10^{-4}$	$\mathbf{1.01139 \pm 0.000013}$	$\mathbf{0.01133 \pm 0.000013}$
	$\lambda = 10^{-5}$	1.01139 ± 0.000023	0.01132 ± 0.000023
CMC	data not permuted	0.99828 ± 0.000081	-0.00172 ± 0.000081
	data permuted	1.01141 ± 0.000007	0.01135 ± 0.000007
NDPE	data not permuted	1.01556 ± 0.000077	0.01518 ± 0.000076
	data permuted	1.01155 ± 0.000077	0.01123 ± 0.000077
WRS	data not permuted	0.99867 ± 0.000420	-0.00133 ± 0.000420
	data permuted	1.01135 ± 0.000039	0.01129 ± 0.000039

Table S1: True values and estimates of $\pi(\varphi)$, for various test functions φ, for the log-normal toy model described in this supplement. For the three embarrassingly parallel approaches (CMC, NDPE, WRS) we present results obtained both without and with first permuting and repartitioning the data into new blocks. For each method the mean estimate $\pm$ Monte Carlo standard error is presented, as computed over 25 replicates; the estimator corresponding to the lowest mean squared error is printed in bold.

Ghosh, J. and Ramamoorthi, R. (2003). *Bayesian Nonparametrics*. Springer Science & Business Media.

Neiswanger, W., Wang, C., and Xing, E. (2014). Asymptotically exact, embarrassingly parallel MCMC. In *Proceedings of the 30th International Conference on Uncertainty in Artificial Intelligence (UAI-14)*, pages 623–632.

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. *International Journal of Management Science and Engineering Management*, 11(2):78–88.

van der Vaart, A. W. (2000). *Asymptotic Statistics*, volume 3 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press.

Wang, X. and Dunson, D. B. (2013). Parallelizing MCMC via Weierstrass sampler. *arXiv preprint arXiv:1312.4605*.

Zhou, Y., Johansen, A. M., and Aston, J. A. (2016). Towards automatic model comparison: an adaptive sequential Monte Carlo approach. *Journal of Computational and Graphical Statistics*, 25(3):701–726.