PARALLEL ADAPTIVE IMPORTANCE SAMPLING

COLIN COTTER*, SIMON COTTER†, AND PAUL RUSSELL‡

Abstract. Markov chain Monte Carlo methods are a powerful and commonly used family of numerical methods for sampling from complex probability distributions. As applications of these methods increase in size and complexity, the need for efficient methods which can exploit the parallel architectures which are prevalent in high performance computing increases. In this paper, we aim to develop a framework for scalable parallel sampling algorithms. At each iteration, an importance sampling proposal distribution is formed using the an ensemble of particles. A stratified sample is taken from this distribution and weighted under the posterior, a state-of-the-art resampling method is then used to create an evenly weighted sample ready for the next iteration. We demonstrate that this parallel adaptive importance sampling (PAIS) method outperforms naive parallelisation of serial MCMC methods for low dimensional problems, and in fact shows better than linear improvements in convergence rates with respect to the number of ensemble members. We also introduce a new resampling strategy, approximate multinomial resampling (AMR), which while not as accurate as other schemes is substantially less costly for large ensemble sizes, which can then be used in conjunction with PAIS for complex problems. In particular, we demonstrate this methodology's superior sampling for multimodal problems, such as those arising from inference for mixture models.

Key words. MCMC, parallel, importance sampling, Bayesian, inverse problems.

1. Introduction. Having first been developed in the early 1970s [18], Markov chain Monte Carlo (MCMC) methods have been of increasing importance and interest in the last 20 years or so. They allow us to sample from complex probability distributions which we would not be able to sample from directly. In particular, these methods have revolutionised the way in which inverse problems can be tackled, allowing full posterior sampling when using a Bayesian framework.

However, this often comes at a very high cost, with a very large number of iterations required in order for the empirical approximation of the posterior to be considered good enough. As the cost of computing likelihoods can be extremely large, this means that many problems of interest are simply computationally intractable.

This problem has been tackled in a variety of different ways. One approach is to construct increasingly complex MCMC methods which are able to use the structure of the posterior to make more intelligent proposals, leading to more thorough exploration of the posterior with fewer iterations. For example, the Hamiltonian or Hybrid Monte Carlo (HMC) algorithm uses gradient information and symplectic integrators in order to make very large moves in state with relatively high acceptance probability [36]. Non-reversible methods are also becoming quite popular as they can improve mixing [3]. Riemann manifold Monte Carlo methods exploit the Riemann geometry of the parameter space, and are able to take advantage of the local structure of the target density to produce more efficient MCMC proposals [16]. This methodology has been successfully applied to MALA-type proposals and methods which exploit even higher order gradient information [6].

Since the clock speed of an individual processor is no longer following Moore's law [26], improvements in computational power are largely coming from the parallelisation of multiple cores. As such, the area of parallel MCMC methods is becoming increasingly of interest. One class of parallel MCMC method uses multiple proposals, with only

*Department of Mathematics, Imperial College, London, UK

†School of Mathematics, University of Manchester, Manchester, UK. e: simon.cotter@manchester.ac.uk. SLC is grateful for EPSRC First grant award EP/L023393/1

‡School of Mathematics, University of Manchester, Manchester, UK

one of these proposals being accepted. Examples of this approach include multiple try MCMC [23] and ensemble MCMC [27]. In [7], a general construction for the parallelisation of MCMC methods was presented, which demonstrated speed ups of up to two orders of magnitude when compared with serial methods on a single core. Another approach involves pre-fetching, where the possible future acceptances/rejections are calculated in advance [2].

A variety of other methods have been designed with particular scenarios in mind. For instance, sampling from high/infinite-dimensional posterior distributions is of interest in many applications. The majority of Metropolis-Hastings algorithms suffer from the curse of dimensionality, requiring more samples for a given degree of accuracy as the parameter dimension is increased. However some dimension-independent methods have been developed, based on Crank-Nicolson discretisations of certain stochastic differential equations [14]. Other ideas such as chain adaptation [17] and early rejection of samples can also aid reduction of the computational workload [38].

However, high dimensionality is not the only challenge that we may face. Complex structure in low dimensions can cause significant issues. These issues may arise due to large correlations between parameters in the posterior, leading to long thin curved structures which many standard methods can struggle with. These features are common, for example, in inverse problems related to epidemiology and other biological applications [19]. Multimodality of the posterior can also lead to incredibly slow convergence in many methods. Many methods allow for good exploration of the current mode, but the waiting time to the next switch of the chain to another mode may be large. Since many switches are required in order for the correct weighting to be given to each mode, and for all of the modes to be explored fully, this presents a significant challenge.

One application where this is an ever-present problem is that of mixture models. Given a dataset, where we know that the data is from two or more different distributions, we wish to be able to identify the parameters, e.g. the mean and variance and relative weight, of each part of the mixture [24]. The resulting posterior distribution is invariably a multimodal distribution, since the likelihood is invariant to permutations. Metropolis-Hastings algorithms, for example, will often fail to converge in a reasonable time frame for problems such as this. Since the posterior may be multimodal, independent of this label switching, it is important to be able to efficiently sample from the whole posterior.

Importance samplers are another class of methods which allow us to sample from complex probability distributions. A related class of algorithms, adaptive importance sampling (AIS) [22] reviewed in [5], had received less attention until their practical applicability was demonstrated in the mid-2000s [4, 8, 10, 20]. AIS methods produce a sequence of approximating distributions, constructed from mixtures of standard distributions, from which samples can be easily drawn. At each iteration the samples are weighted, often using standard importance sampling methods. The weighted samples are used to train the adapting sequence of distributions so that samples are drawn more efficiently as the iterations progress. The weighted samples form a sample from the posterior distribution under some mild conditions [25, 30].

Ensemble importance sampling schemes also exist, e.g. population Monte Carlo (PMC) [9]. PMC uses an ensemble to build a mixture or kernel density estimate (KDE) of the posterior distribution. The efficiency of this optimisation is restricted by the component kernel(s) chosen, and the quality of the current sample from the posterior. Extensions, such as adaptive multiple importance sampling algorithm

(AMIS) [11], and adaptive population importance sampling (APIS) [25] have enabled these methods to be applied to various applications, including population genetics [37].
In this paper, we present a framework for parallelisation of importance sampling, which can be built around many of the current Metropolis-based methodologies in order to create an efficient target distribution from the current ensemble. The method makes use of a resampler based on optimal transport which has been used in the context of particle filters [29]. In particular we demonstrate the advantages of this method when attempting to sample from multimodal posterior distributions, such as those arising from inference for mixture models.
In Section 2 we introduce some mathematical preliminaries upon which we will later rely. In Section 3 we present the general framework of the PAIS algorithm. In Section 4 we consider adaptive versions of PAIS which automatically tune algorithmic parameters concerned with the proposal distributions. In Section 5 we introduce the approximate multinomial resampling (AMR) algorithm which is a less accurate but faster alternative to resamplers which solve the optimal transport problem exactly. In Section 6 we consider consistency of the PAIS algorithm. In Section 7 we present some numerical examples, before a brief conclusion and discussion in Section 8.

2. Preliminaries. In this Section we will introduce preliminary topics and algorithms that will be referred to throughout the paper.

2.1. Bayesian inverse problems. In this paper, we focus on the use of MCMC methods for characterising posterior probability distributions arising from Bayesian inverse problems. We wish to learn about a particular unknown quantity x, of which we are able to make direct or indirect noisy observations. For now we say that x is a member of a Hilbert space X.
The quantity x is mapped on to observable space by the observation operator $\mathcal{G} : X \to \mathbb{R}^d$. We assume that the observations, D, are subject to Gaussian noise,

$$D = \mathcal{G}(x) + \varepsilon, \qquad \varepsilon \sim \mu_\varepsilon = \mathcal{N}(0, \Sigma). \tag{2.1}$$

For example, if x is an initial condition for the wave equation, $\mathcal{G}$ might calculate the height of a wave at a particular point and time.
These modelling assumptions allow us to construct the likelihood of observing the data D given the quantity $x = x^*$. Rearranging (2.1) and using the distribution of ε, we get:

$$\mathbb{P}(D|x = x^*) \propto \exp\left(-\frac{1}{2}\|\mathcal{G}(x^*) - D\|_\Sigma^2\right) = \exp\left(-\Phi(x^*)\right), \tag{2.2}$$

where $\|y_1 - y_2\|_\Sigma = (y_1 - y_2)^\top \Sigma^{-1}(y_1 - y_2)$ for $y_1, y_2 \in \mathbb{R}^d$.
As discussed in [13, 39], in order for this inverse problem to be well-posed in the Bayesian sense, we require the posterior distribution, μ_Y, to be absolutely continuous with respect to the prior, μ_0. A minimal regularity prior can be chosen informed by regularity results of the observational operator $\mathcal{G}$. Given such a prior, then the Radon-Nikodym derivative of the posterior measure, μ_Y, with respect to the prior measure, μ_0, is proportional to the likelihood:

$$\frac{d\mu_Y}{d\mu_0} \propto \exp\left(-\Phi(x^*)\right). \tag{2.3}$$

2.2. Particle filters and resamplers. In this subsection we briefly review particle filters, since the development of the resampler that we incorporate into the PAIS is motivated by this area.

Particle filters are a class of Monte Carlo algorithms designed to solve the filtering problem. That is, to find the best estimate of the true state of a system when given only noisy observations of the system. The solution of this problem has been of importance since the middle of the 20th century in fields such as molecular biology, computational physics and signal processing. In recent years the data assimilation community has contributed several efficient particle filters, including the ensemble Kalman filter (EnKF) [15] and the ensemble transform particle filter (ETPF) [29].

The ETPF defines a coupling T between two random variables Y and X, allowing us to use the induced map as a resampler. An *optimal coupling* T^* is one which maximises the correlation between X and Y [12]. This coupling is the solution to a linear programming problem in M^2 variables with $2M-1$ constraints, where M is the size of the sample. Maximising the correlation preserves the statistics of X in the new sample.

In this work we use these particle filters as resampling methods. We approximate the posterior distribution, μ_Y, with a small sample of weighted particles, $\{(w_i, y_i)\}_{i=1}^M$. In filtering problems the weights would be found by incorporating new observed data whereas here we simply use importance weights. This distribution, $\hat{\mu}_Y \approx \mu_Y$, can then be resampled, using the ETPF or otherwise, into a new distribution, $\hat{\eta} \approx \mu_Y$,

$$\hat{\mu}_Y = \sum_{i=1}^{M} w_i \delta_{y_i}(\cdot) \quad \xrightarrow{\text{ETPF}} \quad \sum_{i=1}^{M} n_i \delta_{x_i}(\cdot) = \hat{\eta}. \tag{2.4}$$

where $n_i \in \{0, 1, \ldots, M\}$ and $\sum_{i=1}^M n_i = M$. This new sample $\{x_i\}_{i=1}^M$ is equally weighted.

Particle filters such as the ETPF are well suited to this problem since it is easy to introduce conditions in the resampling to ensure you obtain the behaviour you require. One downside is that the required ensemble size increases quickly with dimension, making it difficult to use in high-dimensional problems.

2.3. Deficiencies of Metropolis-type MCMC schemes. All MCMC methods are naively parallelisable. One can take a method and simply implement it simultaneously over an ensemble of processors. All of the states of all of the ensemble member can be recorded, and in the time that it takes one MCMC chain to draw N samples, M ensemble members can draw NM samples.

However, we argue that this is not an optimal scenario. First of all, unless we have a lot of information about the posterior, we will initialise the algorithm's initial state in the tails of the distribution. The samples that are initially made as the algorithm finds its way to the mode(s) of the distribution cannot be considered to be samples from the target distribution, and must be thrown away. This process is known as the burn-in. In a naively parallelised scenario, each ensemble member must perform this process independently, and therefore mass parallelisation makes no inroads to cutting this cost.

Moreover, many MCMC algorithms suffer from poor mixing, especially in multimodal systems. The amount of samples that it takes for an MCMC trajectory to switch between modes can be large, and given that a large number of switches are required before we have a good idea of the relative probability densities of these different regions, it can be prohibitively expensive.

Another aspect of Metropolis-type samplers is that information computed about a proposed state is simply lost if we choose to reject that proposal in the Metropolis step. An advantage of importance samplers is that no evaluations of $\mathcal{G}$ are ever wasted since all samples are saved along with their relative weighting.
Moreover, a naively parallelised MCMC scheme is exactly that - naive. Intuition suggests that we can gain some speed up by sharing information across the ensemble members, and this is what we wish to demonstrate in this paper.
These deficiencies of the naive method of parallelising MCMC methods motivated the development of the Parallel Adaptive Importance Sampler (PAIS). In the next section we will introduce the method in its most general form.

3. The Parallel Adaptive Importance Sampler (PAIS). Importance sampling can be a very efficient method for sampling from a probability distribution. A proposal density is chosen, from which we can draw samples. Each sample is assigned a weight given by the ratio of the target density and the proposal density at that point. They are efficient when the proposal density is concentrated in similar areas to the target density, and incredibly inefficient when this is not the case. The aim of the PAIS is to use an ensemble of states to construct a proposal distribution which will be as close as possible to the target density. If this ensemble is large enough, the distribution of states will be approximately representative of the target density.
The proposal distribution could be constructed in many different ways, but we choose to use a mixture distribution, made up of a sum of MCMC proposal kernels. This corresponds to replacing the Dirac distribution in the right-hand side of Equation 2.4 with a chosen distribution e.g. Gaussian. Once the proposal is constructed, we can sample a new ensemble from the proposal distribution, and each is assigned a weight given by the ratio of the target density and the proposal mixture density. Assuming that our proposal distribution is a good one, then the variance of the weights will be small, and we will have many useful samples. Finally, we need to create a set of evenly weighted samples which best represent this set of weighted samples. This is achieved by implementing a resampling algorithm. These samples are not stored in order to characterise the posterior density, since the resampling process inevitably adds some error/bias. They are simply needed in order to inform the mixture distribution for the next iteration of the algorithm.
Initially we will use the ETPF algorithm [29], although we will suggest an alternative strategy in Section 5. The resampling algorithm gives us a set of evenly weighted samples which represents the target distribution well, which we can use to iterate the process again. The algorithm is summarised in Algorithm 1.
We wish to sample states $x \in X$ from a posterior probability distribution μ_D, where D represents our data which we wish to use to infer x. Since we have M ensemble members, we represent the current state of all of the Markov chains as a vector $\mathbf{X} = [x_1, x_2, \ldots, x_M]^\top$. We are also given a transition kernel $\nu(\cdot, \cdot)$, which might come from an MCMC method, for example the random walk Metropolis-Hastings proposal density $\nu(\cdot, x) \sim \mathcal{N}(x, \beta^2)$, where $\beta^2 \in \mathbb{R}$ defines the variance of the proposal.
Since the resampling does not give us a statistically identical sample to that which is inputted, we cannot assume that the samples $\mathbf{X}^{(i)}$ are samples from the posterior. Therefore, as with serial importance samplers, the weighted samples $(\mathbf{W}^{(i)}, \mathbf{Y}^{(i)})_{i=1}^N$ are the samples from the posterior that we will analyse.
The key is to choose a suitable transition kernel ν such that if $X^{(i)}$ is a representative sample of the posterior, then the mixture density $\chi(\cdot; \mathbf{X}^{(i)})$ is a good approximation of the posterior distribution. If this is the case, the newly proposed states $\mathbf{Y}^{(i)}$ will

Algorithm 1: The parallel adaptive importance sampler (PAIS).

Set $\mathbf{X}^{(0)} = \mathbf{X}_0 = [x_1^{(0)}, x_2^{(0)}, \ldots, x_M^{(0)}]^\top$.
for $i = 1, \ldots, N$ **do**
  Sample $\mathbf{Y}^{(i)} = [y_1^{(i)}, y_2^{(i)}, \ldots, y_M^{(i)}]^\top, \quad y_j^{(i)} \sim \nu(\cdot; x_j^{(i-1)})$.
  Calculate $\mathbf{W}^{(i)} = [w_1^{(i)}, w_2^{(i)}, \ldots, w_M^{(i)}]^\top, \quad w_j^{(i)} = \frac{\pi(y_j^{(i)})}{\chi(y_j^{(i)}; \mathbf{X}^{(i-1)})}$, where

$$\chi(\cdot; \mathbf{X}^{(i-1)}) = \frac{1}{M} \sum_{j=1}^{M} \nu(\cdot; x_j^{(i-1)}).$$

  Resample: $(\mathbf{W}^{(i)}, \mathbf{Y}^{(i)}) \to (\frac{1}{M}\mathbf{1}, \mathbf{X}^{(i)})$.
Output $(\mathbf{W}, \mathbf{Y})$.

also be a good sample of the posterior with low variance in the weights $\mathbf{W}^{(i)}$.
In Section 7, we will demonstrate how the algorithm performs, primarily using random walk (RW) proposals. We do not claim that this choice is optimal, but is simply chosen as an example to show that sharing information across ensemble members can improve on the original MH algorithm and lead to tolerances being achieved in fewer evaluations of $\mathcal{G}$. This is important since if the inverse problem being tackled involves computing the likelihood from a very large data set this could lead to a large saving of computational cost. We have observed that using more complex (and hence more expensive) kernels ν, does not significantly improve the speed of convergence of the algorithm for posteriors that we have considered [35].
Care needs to be taken when choosing the proposal distribution to ensure that the proposals are absolutely continuous with respect to the posterior distribution. If this is not the case, importance weights grow exponentially for samples in the tails. This will drastically decrease the efficiency of sampling.

4. Automated tuning of algorithm parameters. Efficient selection of scaling parameters in MCMC algorithms is critical to achieving optimal mixing rates and hence achieving fast convergence to the target density. It is well known that a scaling parameter which is either too large or too small results in a Markov chain with high autocorrelation. One aspect worthy of consideration with the PAIS, is finding an appropriate proposal kernel ν such that the mixture distribution χ is a close approximation to the posterior density π.
Most MCMC proposals have parametric dependence which allows the user to control their variance. For example, in the RW proposal $y = x + \beta\eta$, the parameter β is the standard deviation of the proposal distribution. Therefore the proposal distributions can be tuned such that they are are slightly over-dispersed. This tuning can take place during the burn-in phase of the algorithm. Algorithms which use this method to find optimal proposal distributions are known as adaptive MCMC algorithms, and have been shown to be convergent provided that they satisfy certain conditions [32, 33]. Alternatively, a costly trial and error scheme with short runs of the MCMC algorithm can be used to find an acceptable value of β.
Algorithms which use mixture proposals, e.g. PAIS, must tune the variance of the individual kernels within the proposal mixture. This adaptivity during early iterations

has some added benefits over and above finding an optimal parameter regime for the algorithm. If the initial value of the proposal variance is chosen to be very large, then the early mode-finding stages of the algorithm are expedited. Adaptively reducing the proposal variances to an optimal value then allows us to explore each region efficiently. Using an ensemble of chains allows quick and effective assessment of the value of the optimal scaling parameter.
In many MCMC algorithms such as the Random Walk Metropolis-Hastings (RWMH) algorithm, the optimal scaling parameter can be found by searching for the parameter value which gives an optimal acceptance rate, e.g. for near Gaussian targets the optimal rates are 23.4% for RWMH and 57.4% for MALA [31]. This method is not applicable to PAIS so we must use other statistics to optimise the scaling parameter. Section 4.1 gives some possible methods for tuning β.

4.1. Statistics for Determining the Optimal Scaling Parameter.

4.1.1. Determining optimal scaling parameter using error analysis. When available, an analytic form for the target distribution allows us to assess the convergence of sampling algorithms to the target distribution. Common metrics for this task include the relative error between the sample moments and the target's moments, or the relative L^2 error between the sample histogram and the target density, $\pi(x|D)$. The relative error in the n-th moment, $\hat{m}_n$, is given by:

$$e = \left| \frac{\hat{m}_n - \mathbb{E}[X^n]}{\mathbb{E}[X^n]} \right|, \quad \text{where} \quad \hat{m}_n = \frac{1}{N} \sum_{i=1}^{N} x_i^n, \tag{4.1}$$

and $\{x_i\}_{i=1}^N$ is a sample of size N.
The relative L^2 error, E, between a continuous density function to a piecewise constant approximation of that density, can be defined by considering the difference in mass between the self-normalised histogram of the samples and the posterior distribution over a set of disjoint sets or "bins":

$$E^2 = \sum_{i=1}^{n_b} \left[\int_{R_i} \pi(s|D)\,\mathrm{d}s - vB_i \right]^2 \Big/ \sum_{i=1}^{n_b} \left[\int_{R_i} \pi(s|D)\,\mathrm{d}s \right]^2, \tag{4.2}$$

where the regions $\{R_i\}_{i=1}^{n_b}$ are the d-dimensional histogram bins, so that $\bigcup_i R_i \subseteq X$ and $R_i \cap R_j = \emptyset$, n_b is the number of bins, v is the volume of each bin, and B_i is the value of the ith bin. This metric converges to the standard definition of the relative L^2 error as $v \to 0$.
These statistics cannot be used in general to find optimal values of β since they require the analytic solution, and are expensive to approximate. However they can be used to assess the ability of other indicators to find the optimal scaling parameters in a controlled setting.

4.1.2. The effective sample size. The effective sample size, n_{eff}, can be used to assess the efficiency of importance samplers. Ideally, in each iteration, we would like all M of our samples to provide us with new information about the posterior distribution. In practise, we cannot achieve a perfect effective sample size of M.
The effective sample size of a weighted sample is defined in the following way:

$$n_{\text{eff}} = \frac{\left(\sum_{i=1}^{M} w_i\right)^2}{\sum_{i=1}^{M} w_i^2} \approx \frac{M\mathbb{E}(w)^2}{\mathbb{E}(w^2)} = M\left(1 - \frac{\text{var}(w)}{\mathbb{E}(w^2)}\right).$$

The second two expressions are true when $M \to \infty$. From the last expression, if the variance of the weights is zero then $n_{\text{eff}} = M$; this is our ideal scenario. In the limit $M \to \infty$, maximising the effective sample size is equivalent to minimising the variance of the weights.

The statistic n_{eff} is easier to deal with than the variance of the weights, as it takes values on $[1, M]$ while the variance of the weights takes values on $\mathbb{R}_{\geq}$. Moreover the variance of the weights can vary over many orders of magnitude causing numerical issues, so that the effective sample size is more desirable as an indicator of optimality.

In this paper we tune our simulations using the effective sample size statistic. We calculate this statistic using a sample size of Mn_k, where $1 \leq n_k \leq N$ is sufficiently large enough to obtain a reasonable estimate of n_{eff} with scaling parameter δ_k. Calculating n_{eff} over these subsets of the simulations tends to underestimate the optimal value of the scaling parameter due to the possibility of missing unlikely proposals with extreme weights.

The effective sample size also has another useful property; if we consider the algorithm in the early stages, for example, we have all M particles in the tails of the target searching for a mode. The particle closest to the mode will have an exponentially higher weight and the effective sample size will be close to 1. In later stages the ensemble populates high density regions, and better represents the posterior distribution. This leads to smaller variation in the weights, and a higher effective sample size. By looking for approximations of the effective sample size which look like a stationary distribution, we can tell when the algorithm is working efficiently.

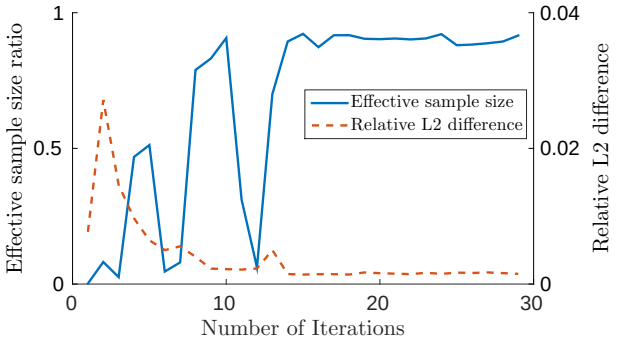
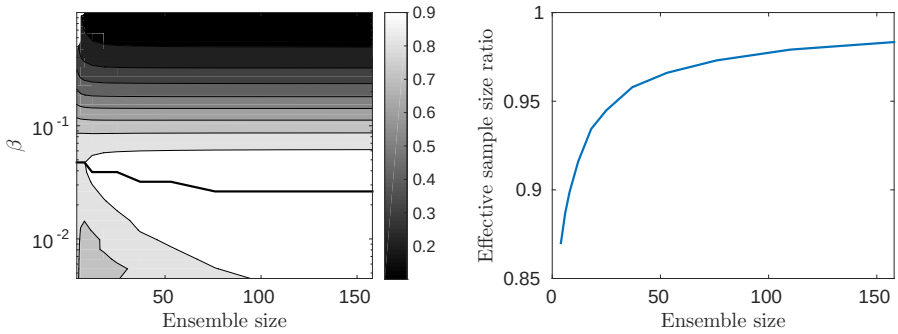


FIG. 4.1. *The effective sample size ratio and relative L^2 difference E during the first 30 iterations. These numerics are taken from the example in Section 7.4.3 using the PAIS-RW algorithm.*

Figure 4.1 demonstrates that the effective sample size flattens out as the relative L^2 difference between the posterior distribution and the proposal distribution stabilises close to its minimum.

Figure 4.2 shows how the effective sample size used in PAIS is affected by the ensemble size. We see from subfigure (a) that as the ensemble size increases, the optimal scaling parameter decreases. This is expected since the larger ensemble allows for finer resolution in the proposal distribution approximation of the posterior distribution. We also see that the algorithm becomes less sensitive to changes in the scaling parameter as the ensemble size increases. Subfigure (b) shows that as the ensemble size increases, the algorithm becomes more efficient.



(a) Contours showing optimal ranges of the scaling parameter.

(b) The highest effective sample size achieved for each ensemble size.

FIG. 4.2. *The behaviour of the effective sample size as the ensemble size increases, considering the example in Section 7.2 using the PAIS-RW algorithm.*

4.2. Adaptively Tuned PAIS. To adapt the scaling parameter β, we use a version of the gradient ascent method modified for a stochastic function. Some more sophisticated examples are described in [1, 21, 33].

Algorithm 2: Adaptively tuned PAIS algorithm.

1 Define update times $\{n_k\}_k$.
2 **for** $n = 1, \ldots, N$ **do**
3 Complete steps 3-5 of Algorithm 1.
4 **if** $n \in \{n_k\}$ **then**
5 Divide ensemble into two halves. Use these halves to estimate the gradient in n_{eff} at β_k.
6 Update β_k using gradient ascent,

$$\beta_{k+1} = \beta_k + \gamma \nabla n_{\text{eff}}.$$

Our adaptive algorithm is given in Algorithm 2. We choose update times which, as suggested in Section 4.1.2 allow for a reasonable estimate of the effective sample size, but do not waste too many iterations. In Step 6, the gradient ascent parameter γ may decrease over time, e.g. as a function of n_k.

5. Approximate Multinomial Resampling. Although the ETPF is optimal in terms of preserving statistics of the sample, it can also become quite costly as the number of ensemble members is increased. It is arguable that in the context of PAIS, we do not require this degree of accuracy, and that a faster more approximate method for resampling could be employed. One approach would be to use the bootstrap resampler, which simply takes the M ensemble members' weights and constructs a multinomial distribution, from which M samples are drawn. This is essentially the cheapest resampling algorithm that one could construct. However it too has some drawbacks. The algorithm is random, and as such it is possible for all of the ensemble members in a particular region not to be sampled. This could be particularly

problematic when attempting to sample from a multimodal distribution, where it might take a long time to find one of the modes again. The bootstrap filter is also not guaranteed to preserve the mean of the weighted sample, unlike the ETPF.
Ideally, we would like to use a resampling algorithm which is not prohibitively costly for moderately or large sized ensembles, which preserves the mean of the samples, and which makes it much harder for the new samples to forget a significant region in the density. This motivates the following algorithm, which we refer to as approximate multinomial resampling (AMR).
Instead of sampling M times from an M-dimensional multinomial distribution as is the case with the bootstrap algorithm, we sample once each from M different multinomials. Suppose that we have M samples y_n with weights w_n. The multinomial sampled from in the bootstrap filter has a vector of probabilities given by:

$$\frac{1}{\sum w_n}[w_1, w_2, \ldots, w_M] = \bar{\mathbf{w}},$$

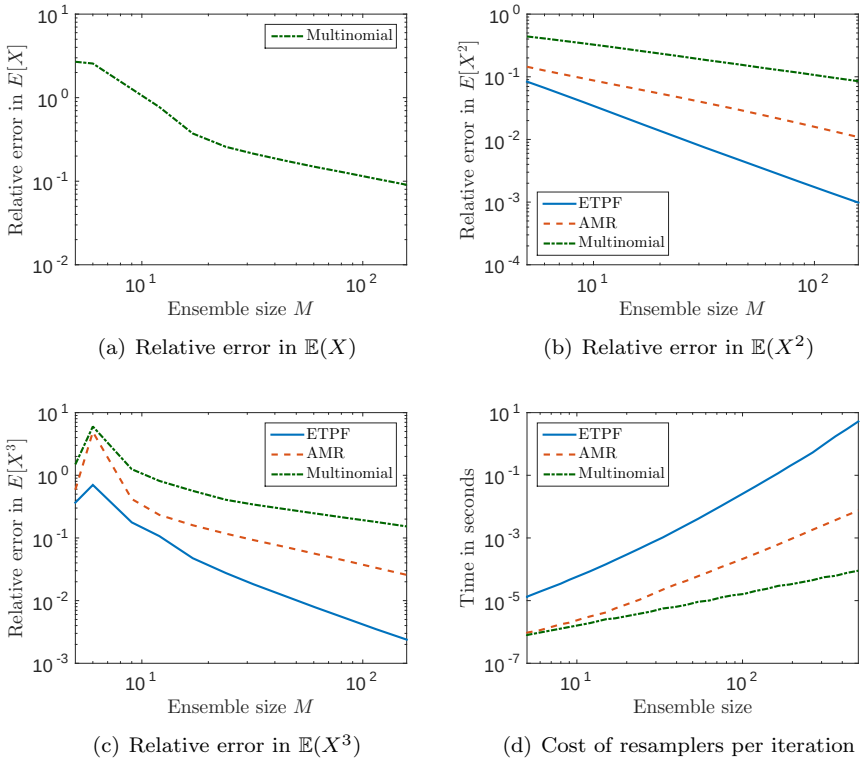
with associated states y_n. We wish to find M vectors $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M\} \subset \mathbb{R}^M_{\geq 0}$ such that $\frac{1}{M}\sum \mathbf{p}_i = \bar{\mathbf{w}}$. The AMR is then given by a sample from each of the multinomials defined by the vectors $\mathbf{p}_i = [p_{i,1}, p_{i,2}, \ldots, p_{i,M}]$ with associated states $\mathbf{y}_i$. Alternatively, as with the ETPF, a deterministic sample can be chosen by picking each sample to be equal to the mean value of each of these multinomial distributions, i.e. each new sample $\hat{x}_i$ is given by:

$$\hat{x}_i = \sum p_{i,j} x_j, \qquad i \in \{1, 2, \ldots, M\}. \tag{5.1}$$

The resulting sample has several properties which are advantageous in the context of being used with the PAIS algorithm. Firstly, we have effectively chopped up the multinomial distribution used in the bootstrap filter into M pieces, and we can guarantee that exactly one sample will be taken from each section. This leads to a much smaller chance of losing entire modes in the density, if each of the sub-multinomials is picked in an appropriate fashion. Secondly, if we do not make a random sample for each multinomial with probability vector $\mathbf{p_i}$ but instead take the mean of the multinomial to be the sample, this algorithm preserves the mean of the sample exactly. Lastly, as we will see shortly, this algorithm is significantly less computationally intensive than the ETPF.
There are of course infinitely many different ways that one could use to split the original multinomial up into M parts, some of which will be far from optimal. The method that we have chosen is loosely based on the idea of optimal transport. We search out states with the largest weights, and choose a cluster around these points based on the closest states geographically. This method is not optimal since once most of the clusters have been selected the remaining states may be spread across the parameter space.
Algorithm 3 describes the basis of the algorithm with deterministic resampling, using the means of each of the sub-multinomials as the new samples. This resampler was designed with the aims of being numerically cheaper than the ETPF, and more accurate than straight multinomial resampling. Therefore we now present numerical examples which demonstrate this.
To test the accuracy and speed of the three resamplers (ETPF, bootstrap and AMR), we drew a sample of size M from the proposal distribution $\mathcal{N}(1, 2)$. Importance weights were assigned, based on a target distribution of $\mathcal{N}(2, 3)$. The statistics of

Algorithm 3: The approximate multinomial resampler (AMR).

$\mathbf{z} = M\bar{\mathbf{w}}$.
for $i = 1, \ldots, M$ **do**
  $J = \arg\max_j z_j$.
  $p_{i,J} = \min\{1, z_J\}$.
  $z_J = z_J - p_{i,J}$.
  while $\sum_j p_{i,j} < 1$ **do**
    $K = \arg\min_{k \in \{k | z_k > 0\}} \|y_J - y_k\|$.
    $p_{i,K} = \min\{1 - \sum_j p_{i,j}, z_K\}$.
    $z_K = z_K - p_{i,K}$.
  $x_i = \sum_k p_{i,k} y_k$.



(a) Relative error in $\mathbb{E}(X)$ (b) Relative error in $\mathbb{E}(X^2)$

(c) Relative error in $\mathbb{E}(X^3)$ (d) Cost of resamplers per iteration

FIG. 5.1. *Comparison of the performance between different resampling schemes. The example in Section 7.2 is implemented for this demonstration.*

the resampled outputs were compared with the original weighted samples. Figure 5.1 (a)-(c) show how the relative errors in the first three moments of the samples changes with ensemble size M for the three different samplers. As expected, the AMR lies somewhere between the high accuracy of the ETPF and the less accurate bootstrap resampling. Note that only the error for the bootstrap multinomial sampler is presented for the first moment since both the ETPF and the AMR preserve the

mean of the original weighted samples up to machine precision. Figure 5.1 (d) shows how the computational cost, measured in seconds, scales with the ensemble size for the three different methods. These results demonstrate that the AMR behaves how we wish, and importantly ensures that exactly one sample of the output will lie in each region with weights up to $\frac{1}{M}$ of the total.
We will use the AMR in the numerics in Section 7.4 where we have chosen to use a larger ensemble size. We do not claim that the AMR is the optimal choice within PAIS, but it does have favourable features, and demonstrates how different choices of resampler can affect the speed and accuracy of the PAIS algorithm.

6. Consistency of PAIS. As outlined in [25] consistency of population AIS algorithms can be considered in two different cases. In the first case we fix the number of iterations $N < \infty$, but allow the population size to grow to infinity $M \to \infty$. In the second case we hold the population size fixed $M < \infty$, and allow infinitely many iterations $N \to \infty$.
In case one, from standard importance sampling results we know that for an iteration n, as $M \to \infty$, we obtain a consistent estimator for any statistic of interest,

$$\hat{\phi}(\Theta) \approx \frac{1}{\hat{Z}} \sum_{i=1}^{M} \frac{1}{M} w_i \phi(\theta_i) \to \phi(\Theta),$$

where the normalisation constant, $\hat{Z} = \frac{1}{M} \sum_{i=1}^{M} w_i$, also converges to the true normalisation constant Z [30].
Case two is slightly more involved. Estimation of the normalisation constant Z is biased, and so estimates of statistics are sums of independent but biased estimators. Since the estimators are independent, the proof of consistency of PAIS in this second limit can be approached in the same way as the PMC algorithm, where it has been shown that $\hat{Z} \to Z$ [30]. Since the normalisation constant is consistent, sums of the independent estimators are also consistent.

7. Numerical Examples. In this section we demonstrate convergence of the PAIS algorithm. We also investigate the convergence properties of PAIS compared with naively parallelised Metropolis-Hastings samplers. We assess the performance of the PAIS algorithm using the relative L^2 error defined in Equation (4.2), as well as the relative error in the first moment (Equation (4.1)).

7.1. Numerical implementation. For each example, we perform the following three tasks.

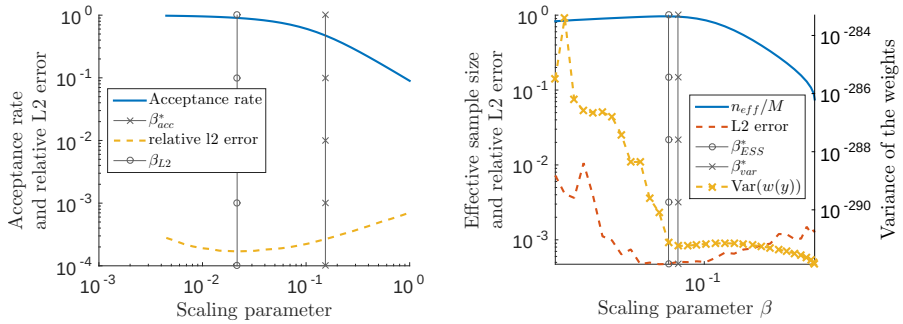
1. **Finding the optimal scaling parameters**: We chose 32 values of β evenly spaced on a log scale in the interval $[10^{-5}, 2]$. We ran the PAIS and MH algorithms for one million iterations, each with an ensemble size of $M = 50$. We took 32 repeats and used the geometric means of the sample statistics in Section 4.1 to find the optimal scaling parameter.
2. **Measuring convergence of nonadaptive algorithms**: We ran the PAIS and MH algorithms, using the scaling parameters found in Step 1, for one million iterations, again with $M = 50$. The simulations are repeated 32 times. The performance is evaluated using the relative L^2 error defined in Equation (4.2).
3. **Measuring convergence of adaptive algorithms**: We run the adaptive algorithms under the same conditions as the nonadaptive algorithms, and again use the relative L^2 error to compare efficiency. The adaptive algorithms are initialised with $\beta_1 = 1$.

7.2. Sampling from a one dimensional Gaussian distribution. This first example shows how PAIS handles searching for the mode of a Gaussian distribution when the initial state is a long way out in the tails. We optimise the naively parallelised RWMH algorithm using the optimal acceptance rate $\hat{\alpha} = 0.5$. This value differs from the theoretical asymptotic value of 0.234 which applies in higher dimensions. This higher acceptance rate is commonly used for one dimensional Gaussian posteriors [34]. To tune the PAIS algorithm we maximise the effective sample size as discussed in Section 4.1.2.

7.2.1. Target distribution. Consider the case of a linear observation operator which maps a one dimensional parameter onto the observable space, $\mathcal{G}\colon x \mapsto x$. We assign centred priors to the parameter x, as well as to the observational noise. We choose the true value of the parameter to be $x_{\text{ref}} = 4$, and choose $\sigma^2 = \tau^2 = 0.1$. Then following Equations (2.1) and (2.2),

$$\text{law}(\mu_D) = \pi(x|D) \propto \exp\left(-\frac{1}{2}\|x - D\|^2_{\sigma^2} - \frac{1}{2}\|x\|^2_{\tau^2}\right). \tag{7.1}$$

This set-up results in a posterior density in which the vast majority of the density is out in the tails of the prior distribution. The Kullback-Leibler (KL) divergence, which gives us a measure of how different the prior and posterior are, is $D_{KL}(\mu_D||\mu_0) = 4.67$ for this problem. A KL divergence of zero indicates that two distributions are identical almost everywhere.



(a) RWMH algorithm.

(b) PAIS-RW algorithm.

FIG. 7.1. *Finding optimal values of β for the example in Section 7.2. The setup is as in Section 7.1. Resampling is performed using the ETPF.*

7.2.2. Optimising the scaling parameter. Figure 7.1 (a) shows the two values of β which are optimal according to the acceptance rate and relative L^2 error criteria for the RWMH algorithm. The smaller estimate comes from the relative L^2 error, and the larger from the acceptance rate. The results in Figure 7.1 are summarised in Table 7.1. Since in general we cannot calculate the relative L^2 error, we must optimise the algorithm using the acceptance rate. From the relative L^2 error curve we can see that the minimum is very wide and despite the optimal values being very different there is not a large difference in the convergence rate.
Figure 7.1 (b) shows the effective sample size ratio compared to the error analysis and the variance of the weights. The relative L^2 error graph is noisy, but it is clear that the maximum in the effective sample size and the minimum in the variance of

Statistic	RWMH
β^*_{L2}	2.1e-2
$\beta^*_{\%}$	1.5e-1
Acceptance Rate (β^*_{L2})	9.0e-1
Acceptance Rate ($\beta^*_{\%}$)	5.0e-1

Statistic	PAIS-RW
β^*_{eff}	4.7e-2
$\beta^*_{\text{var}(w(y))}$	5.8e-2
β^*_{L2}	3.9e-2

TABLE 7.1

Optimal values of β summarised from Figure 7.1. Statistics calculated as described in Section 4.1. The values β^*_{L2} and $\beta^*_{\%}$ are the optimal scaling parameters found by optimising the relative L^2 errors and acceptance rate respectively. Similarly β^*_{eff} and $\beta^*_{var(w(y))}$ optimise the effective sample size and variance of the weights statistics.

the weights are both close to the minimum in the relative L^2 error. Due to this we say that the estimate of the effective sample size found by averaging the statistic over each iteration is a good indicator for the optimal scaling parameter.

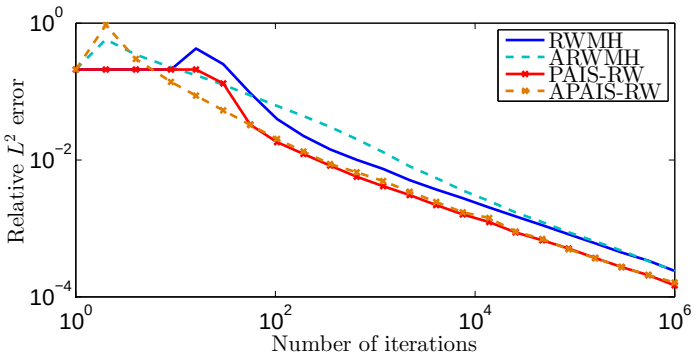


FIG. 7.2. *Convergence of the (A)RWMH and (A)PAIS-RW algorithms. The setup is as in Section 7.1 (2, 3). Resampling is performed using the ETPF.*

7.2.3. Convergence of RWMH vs PAIS-RW. Figure 7.2 shows that the PAIS-RW algorithm converges to the posterior distribution significantly faster than the RWMH algorithm, in both L^2 error and relative error in the moments. A description of the speed up attained by this algorithm is given in Section 7.3.4. The adaptive algorithms are also shown in Figure 7.2. We see that both adaptive algorithms converge to the posterior at a similar speed to the respective optimised algorithm. This shows that, particularly for PAIS, we can optimise simulations efficiently on the fly.

7.2.4. Scaling of the PAIS algorithm with ensemble size. In this and the next example we use an ensemble size $M = 50$, however it is interesting to see how the PAIS algorithm scales when we increase the ensemble size, and if there is a minimum required ensemble size. We implement the problem in Section 7.2, using the RWMH and PAIS-RW algorithms with ensemble sizes in the interval $M \in [1, 160]$.
Figure 7.3 was produced using the method of finding optimal β described in Section 7.1 (1), then running 32 repeats at each ensemble size
We repeat the process described in Section 7.1 for each of the ensemble sizes we are interested in. The convergence rates are then found by regressing through the data. The graph is still very noisy but demonstrates that increasing the ensemble

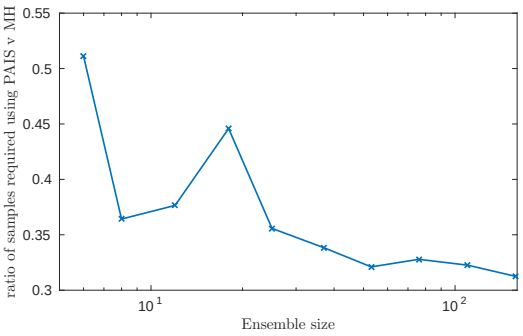


FIG. 7.3. *Ratio of PAIS-RW samples required to reach the same tolerance as the RWMH algorithm.*

size continues to reduce the number of iterations required in comparison with naively parallelised MH. The decreasing trend shows superlinear improvement of PAIS with respect to ensemble size, in terms of the number of iterations required, which is a demonstration of our belief that parallelism of MCMC should give us added value over and above that provided by naive parallelism. This decrease is due to the increasing effective sample size shown in Figure 4.2 (b).

7.3. Sampling from Bimodal Distributions. In this second example we investigate the behaviour of the PAIS algorithm when applied to bimodal problems. MH methods can struggle with multimodal problems, particularly where switches between the modes are rare, resulting in incorrectly proportioned modes in the histograms. This example demonstrates that the PAIS algorithm redistributes particles to new modes as they are found. This means that we expect the number of particles in a mode to be approximately proportional to the probability density in that mode, resulting in faster global convergence.

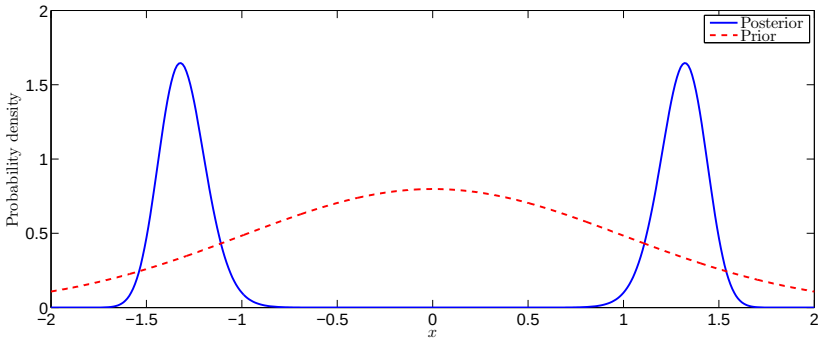


FIG. 7.4. *Posterior distribution for the bimodal example in Section 7.3.1.*

7.3.1. Target distribution. In this example we look at a bimodal posterior distribution which has a high energy barrier between the two modes. In the MH algorithm, the expected number of iterations for a switch between modes is a large proportion of the total number of iterations. This posterior is presented in Figure 7.4.

We consider a non-linear observation operator $\mathcal{G}\colon x \mapsto x^2$, and assign the prior $\mu_0 = \mathcal{N}(0, \tau^2 = 0.25)$. Again, we assume the true parameter $x_{\text{ref}} = 2$ is observed according to Equation (2.1) with observational noise $\sigma^2 = 0.1$, and the posterior is

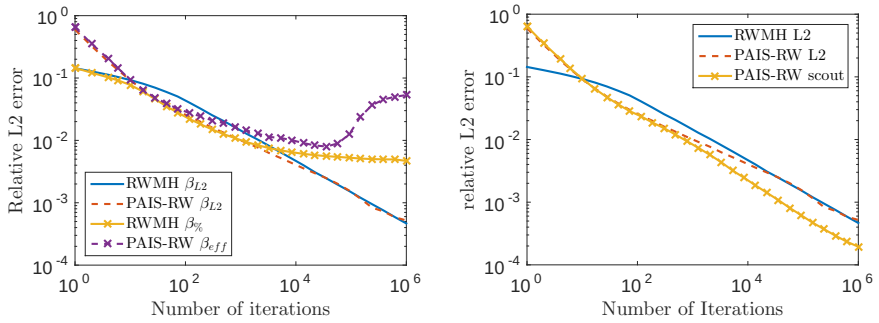
$$\pi(x|D) \propto \exp\left(-\frac{1}{2\sigma^2}\|x^2 - D\|^2 - \frac{1}{2\tau^2}\|x\|^2\right).$$

7.3.2. Calculating values of the optimal scaling parameter. Calculating the optimal values of the scaling parameters for this example is similar to the previous example. These values are given in Table 7.2. The subscript on β refers to the criterion which has been optimised.

Algorithm	β^*_{acc}	β^*_{eff}	β^*_{L2}
RWMH	2.3e-1	-	9.3e-1
PAIS-RW	-	5.1e-2	1.3e-1

TABLE 7.2
Optimal scaling parameters for the example in Section 7.3.

Since transitions between the modes are extremely unlikely, we consider the convergence on two levels; convergence to equally proportioned modes, and convergence to smooth histograms. To get correctly proportioned modes with the RWMH algorithm it is important that the chains can transition between the modes frequently, which requires a large proposal variance. However, this leads to a lower acceptance rate, and so we sacrifice convergence locally. For this reason, the RWMH algorithm is very slow to converge for problems of this type.
We can achieve these two regimes in RWMH by tuning β using different statistics; the acceptance rate for local convergence, and the L^2 error for global convergence. Similarly in PAIS-RW we can use the effective sample size for local convergence, and the L^2 error for global convergence.



(a) Relative L^2 error using the optimal scaling parameters.

(b) Convergence of locally optimised PAIS-RW with scout chain, and globally optimised PAIS-RW and RWMH.

FIG. 7.5. *Convergence of the PAIS-RW and RWMH algorithms for the bimodal example discussed in Section 7.3. Set up described in Section 7.1. Resampling is performed using the ETPF.*

7.3.3. Convergence of RWMH vs PAIS-RW. The MH and PAIS algorithms are run with both the globally and locally optimal scaling parameters. Figure 7.5 (a)

shows that the globally optimal β^* leads to normal convergence rates, $\mathcal{O}(n^{-1/2})$, whereas algorithms using locally optimal scaling parameters initially converge faster but do not sample proportionally for the whole simulation.

Using the PAIS algorithm we can ensure that we do not lose modes as simulation progresses. Since we have an ensemble, we can sacrifice locally optimal sampling for one or two ensemble members in favour of a larger proposal variance. These chains with larger scaling parameters act both as 'scouts' for new modes, and act to aid in the over-dispersal of the proposal distribution. Figure 7.5 (b) shows the results of using 49 chains with the local optimal scaling parameter, and one chain with ten times the local optimal scaling parameter. We see that modes are not forgotten and the algorithm converges with the improvement we see from PAIS-RW in the Gaussian example.

The adaptive algorithm can just as easily be applied to the PAIS algorithm with the 'scout' chains described in the previous paragraph. Since we need two equally sized sub-ensembles, we will use two groups of 25 ensemble members. Each group has 24 ensemble members tuned for locally optimal convergence and one 'scout' chain with a scaling parameter ten times that of the rest of the group.

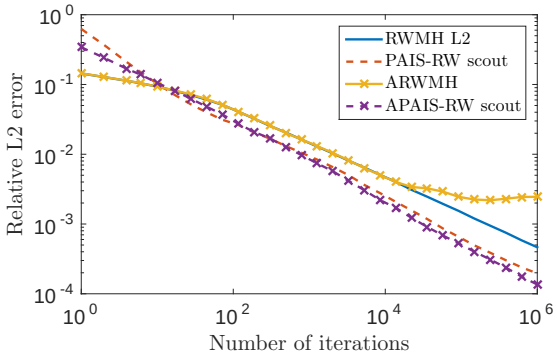


FIG. 7.6. *Convergence of modified PAIS and RWMH algorithms, comparing the globally optimised nonadaptive algorithms with the locally optimised adaptive algorithms. The setup is as described in Section 7.1 (3). Resampling is performed using the ETPF.*

Comparing the convergence of the adaptive algorithms against the nonadaptive algorithms in Figure 7.6 shows that the algorithms behave as expected. The adaptive RWMH algorithm tuned using the acceptance rate converges at the same rate as the L^2 optimised algorithm until the scaling parameter gets too small, and therefore switches between the modes are rare, and the relative heights of the modes are decided by the arbitrary proportion of chains which are in each mode at this point. The adaptive PAIS-RW with scout chains tuned to the effective sample size converges at about the same rate as the locally optimised nonadaptive algorithm also with scouts.

7.3.4. Calculating the speed up in convergence. The graphs in the previous section clearly show that the PAIS-RW algorithm converges faster than the RWMH algorithm when both are parallelised with the same ensemble size. We can calculate the number of iterations required to achieve a particular tolerance level in our solution for each algorithm and compare these to calculate a percentage saving. In Figure 7.7 we demonstrate our calculation of the savings. The constants c_1 and c_2 are found by regressing through the data with a fixed exponent of $-1/2$ excluding the initial data

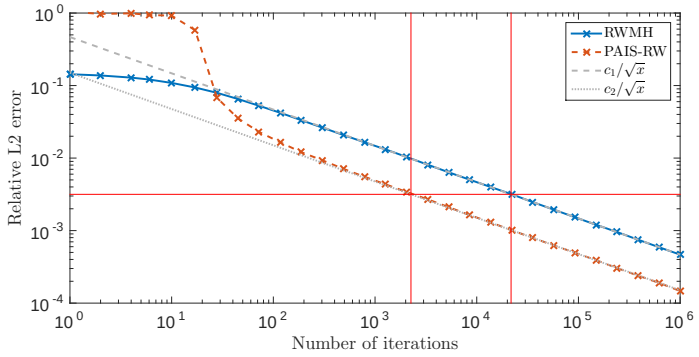


FIG. 7.7. *Illustration of calculating the number of PAIS-RW iterations required to reach a tolerance of $10^{-2.8}$ as a percentage of MH iterations. The relative L^2 error graphs are from the Gaussian problem in Section 7.2.*

points where the algorithm has not finished burning in.

	Gaussian	Bimodal
RWMH	10%	12.6% (scout)
MALA	42%	40%

TABLE 7.3

Iterations for the PAIS algorithms required to achieve a desired tolerance as a percentage of the number of iterations required by the respective MH algorithms.

A summary of the percentage of iterations required using the PAIS algorithm compared with the respective Metropolis-Hastings algorithms is given in Table 7.3.

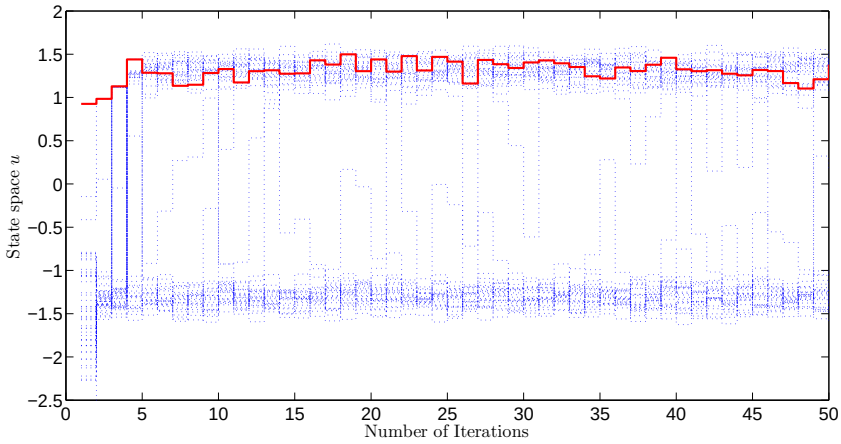


FIG. 7.8. *This figure demonstrates the redistribution property of the PAIS algorithm. Initially there is one chain in the positive mode, and 49 chains in the negative mode.*

7.3.5. A useful property of the PAIS algorithm for multimodal distributions. The biggest issue for the Metropolis-Hastings algorithms when sampling

from bimodal posterior distributions is that it is unlikely that the correct ratio of chains will be maintained in each of the modes, and since there is no interaction between the chains, there is no way to remedy this problem. The PAIS algorithm tackles this problem with its resampling step. The algorithm uses its dynamic kernel to build up an approximation of the posterior at each iteration, and then compares this to the posterior distribution via the weights function. Any large discrepancy in the approximation will result in a large or small weight being assigned to the relevant chain, meaning the chain will either pull other chains towards it or be sucked towards a chain with a larger weight. In this way, the algorithm allows chains to 'teleport' to regions of the posterior which are in need of more exploration. Figure 7.8 shows the trace of a simulation of the PAIS-RW algorithm with initially 1 chain in the positive mode, and 49 chains in the negative mode. It takes only a handful of iterations for the algorithm to balance out the chains into 25 chains in each mode. The chains switch modes without having to climb the energy gradient in the middle.

7.4. A Mixture Model. The technique of mixture modelling employs well known parametric families to construct an approximating nonparametric distribution which may have a complex structure. Most commonly, Gaussian kernels are used since underlying properties in the data can often be assumed to follow a Gaussian distribution. An example would be if a practitioner were to measure the heights of one hundred adults, but failed to record their gender. The data could be considered as one population with two sub populations, male and female. The problem then might be to find the average height of adult males from the data. In this case, since height is often considered to follow a Gaussian distribution, it makes sense to model the population as a mixture of two univariate Gaussian distributions.

A well known problem in the Bayesian treatment of mixture modelling is that of identification, sometimes referred to as the label-switching phenomenon. The likelihood distribution for mixture models is invariant under permutations of the mixture labels. If a mixture has n means and the point $(\mu_1, \ldots, \mu_n)$ maximises the likelihood, then the likelihood will also be maximised by $(\mu_{\varphi(1)}, \ldots, \mu_{\varphi(n)})$ for all permutations $\varphi(\cdot)$. This means that the number of modes in the posterior distribution is of order $\mathcal{O}(n!)$. As we have seen it can be hard for standard MH algorithms to obtain reliable inference for posterior distributions with a large number of modes, or even a small number of modes which are separated by a large distance.

7.4.1. Target Distribution. In particular we look at a data set where we assume that there are two subpopulations within the overall population. Since both subpopulations will be approximated by Gaussian distributions we have five parameters which we need to be estimated, two means $\{\mu_1, \mu_2\}$, two variances $\{\sigma_1^2, \sigma_2^2\}$, and the probability, p, that an individual observation belongs to the first subpopulation. We have 100 data points, D_i, which we assume to be distributed according to

$$D_i \sim p\mathcal{N}(\mu_1, \sigma_1^2) + (1-p)\mathcal{N}(\mu_2, \sigma_2^2), \quad i = 1, \ldots, 100,$$

where $p \in [0,1]$, $\mu_1, \mu_2 \in \mathbb{R}$ and $\sigma_1^2, \sigma_2^2 \in \mathbb{R}^+$. Due to the domains of these parameters and also some prior knowledge, we assign the priors

$$p \sim \text{Beta}(1,1), \quad \mu_{1,2} \sim \mathcal{N}(0,4) \quad \text{and} \quad \sigma_{1,2}^2 \sim \text{Gamma}(\alpha = 2, \beta = 1).$$

If we collect these parameters in the vector $\theta = (p, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)^\top$, the resulting posterior distribution is

$$\pi(\theta|D) \propto \prod_{i=1}^{100} (p\mathcal{N}(D_i; \mu_1, \sigma_1^2) + (1-p)\mathcal{N}(D_i; \mu_2, \sigma_2^2)) \prod_{i=1}^{5} \pi_0^i(\theta_i), \tag{7.2}$$

where $\pi_0^i(\cdot)$ is the prior density function corresponding to θ_i.

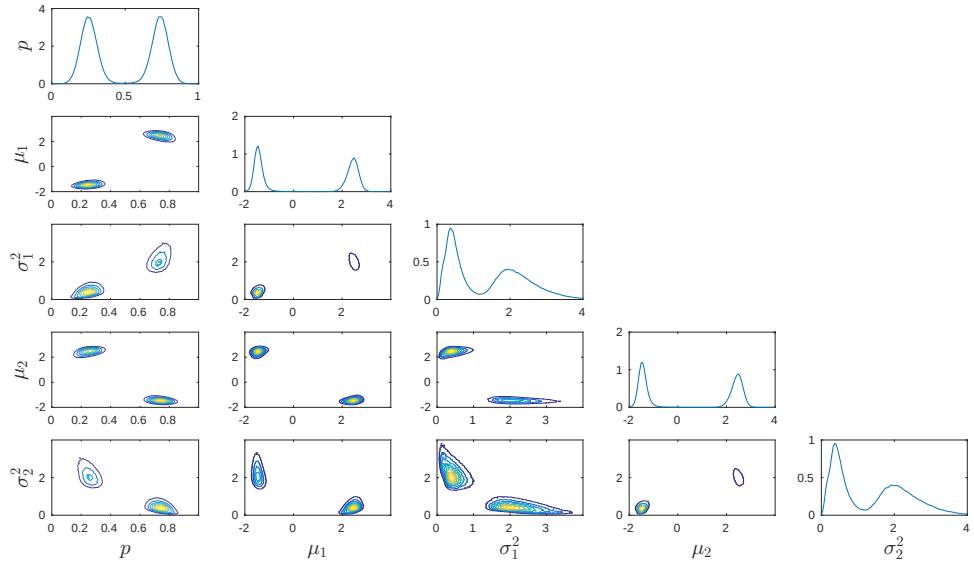


FIG. 7.9. *The posterior distribution of θ as given in Equation (7.2) as found from 10 million samples from the PAIS algorithm. The main diagonal contains the marginal distributions of each θ_i, and the lower triangular contours represents the correlation between pairs of parameters.*

Figure 7.9 presents a visualisation of the posterior distribution for this problem, created from 10million samples produced by the PAIS-RW algorithm.

7.4.2. Implementation. We have no analytic form for the posterior distribution for this model, and MH cannot reliably sample from this type of bimodal distribution. We do however know that the probability density should be evenly divided between the two modes. This is due to the symmetric prior for p, and and the same priors being assigned to μ_1 and μ_2, and also to σ_1^2 and σ_2^2. To decide which mode a sample belongs to we define a plane which bisects the posterior so that each point on this plane lies exactly halfway between the two true solutions to the inverse problem i.e. the value of θ used to generate the data, and also the θ obtained by a relabelling of the parameters. Now that we can assign a sample to a particular mode, we can calculate the density in each mode by summing the weights associated to all samples in that mode,

$$\bar{w}_k = \sum_{i=1}^{N} w_i I_{X_i \in \text{Mode } k}, \quad k = 1, 2,$$

and the relative error in the amount of density in each mode is then

$$w_{\text{error}} = 2\left|\frac{\bar{w}_1}{\bar{w}_1 + \bar{w}_2} - \frac{1}{2}\right|. \tag{7.3}$$

Since the probability p is constrained to lie in the interval $[0, 1]$, and the variances must be positive, it can be wasteful to use Gaussian proposal distributions. For this example we make proposals by matching the proposal distribution to the prior, scaling the variance and centring the proposal at the previous value. So

$$q_p \sim \text{Beta}(\delta^{-2}p, \delta^{-2}(1-p)), \quad q_{\mu_{1,2}} \sim \mathcal{N}(\mu_{1,2}, 4\delta^2) \quad \text{and} \quad q_{\sigma^2_{1,2}} \sim \text{Gamma}(\alpha^*, \beta^*),$$

where $\alpha^* = \sigma^2_{1,2}\beta^*$, $\beta^* = \sigma^2_{1,2}/2\delta^2$ and δ is a scaling parameter to be tuned. This means that our proposal distributions will not be a mixture of multivariate Gaussians, but independent mixtures of univariate Beta, Gamma and Gaussian distributions.
In the numerics which follow we have increased the ensemble size from $M = 50$ to $M = 500$ to compensate for the increase in dimension. We also use the AMR algorithm for the resampling step because of the reduced computational cost. The method otherwise remains the same as in previous examples. We perform test runs to find the optimal scaling parameters considering convergence to modes with equal density. We then calculate the convergence rates of the algorithms by producing 10 million samples from the posterior with each algorithm, and repeat the simulation 32 times.

Algorithm	MH	PAIS
δ^*	1.3e-1	2.3e-1

TABLE 7.4

Optimal values of the scaling parameter. The MH algorithm is optimised using the acceptance rate, and the PAIS algorithm is optimised using the effective sample size.

7.4.3. Convergence of MH vs PAIS. The optimal scaling parameters for the MH and PAIS algorithms with the proposal distributions described in Section 7.4.2 are given in Table 7.4.
Convergence of the relative error for the two algorithms is displayed in Figure 7.10. PAIS converges at the expected $\mathcal{O}(1/\sqrt{N})$ rate, whereas the MH algorithm converges to locally smooth histograms but with the wrong proportion of samples in each mode. The relatively low value of the error for the MH example is due to the priors covering the sample space evenly, however since transitions are near impossible with a small value of the scaling parameter, this error will take a very long time to reduce. This problem was discussed in Section 7.3.

8. Discussion and Conclusions. We have explored the application of parallelised MCMC algorithms in low dimensional inverse problems. We have demonstrated numerically that these algorithms converge faster than the analogous naively parallelised Metropolis-Hastings algorithms. Further experimentation with the Metropolis Adjusted Langevin Algorithm (MALA), preconditioned Crank-Nicolson (pCN), preconditioned Crank-Nicolson Langevin (pCNL) and Hamiltonian Monte Carlo (HMC) proposals has yielded similar results [35].
Importantly, we have compared the efficiency of our parallel scheme with a naive parallelisation of serial methods. Thus our increase in efficiency is over and above an

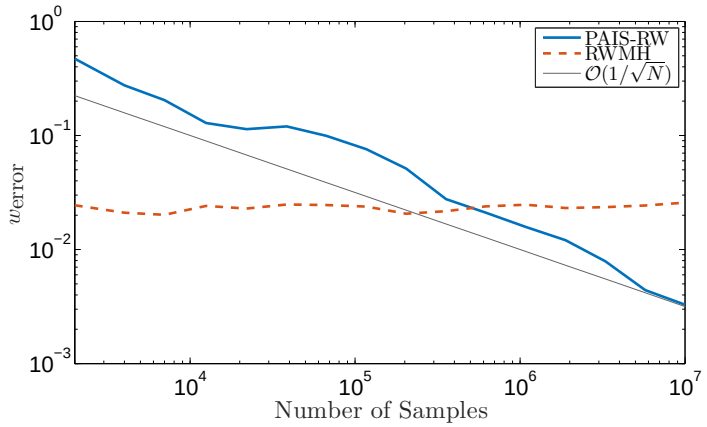


FIG. 7.10. *Convergence of the PAIS algorithm for the mixture model described in Section 7.4, convergence judged using the criterion in Equation* (7.3)*. Implementation described in Sections 7.4.2 and 7.4.3. Resampling is performed using the AMR scheme.*

M-fold increase, where M is the number of ensemble members used. Our approach demonstrates a better-than-linear speed-up with the number of ensemble members used.

The PAIS has a number of favourable features, for example the algorithm's ability to redistribute, through the resampling regime, the ensemble members to regions which require more exploration. This allows the method to be used to sample from complex multimodal distribution.

Another strength of the PAIS is that it can also be used with any MCMC proposal. There are a growing number of increasing sophisticated MCMC algorithms (non-reversible Langevin/HMC proposals, Riemann manifold MCMC etc) which could be incorporated into this framework, leading to even more efficient algorithms, and this is another opportunity for future work.

One limitation of the PAIS approach as described above is that a direct solver of the ETPF problem (such as FastEMD [28]) has computational cost $\mathcal{O}(M^3 \log M)$, where M is the number of particles in the ensemble. As such, we introduced a more approximate resampler the approximate multinomial resampler, which allows us to push the approach to the limit with much larger ensemble sizes. The PAIS framework is very flexible in terms of being able to use any combination of proposal distributions and resampling algorithms that one wishes.

REFERENCES

[1] C. ANDRIEU, É. MOULINES, ET AL., *On the ergodicity properties of some adaptive MCMC algorithms*, The Annals of Applied Probability, 16 (2006), pp. 1462–1505.

[2] E. ANGELINO, E. KOHLER, A. WATERLAND, M. SELTZER, AND R. ADAMS, *Accelerating MCMC via parallel predictive prefetching*, arXiv preprint arXiv:1403.7265, (2014).

[3] J. BIERKENS, *Non-reversible Metropolis-Hastings*, Statistics and Computing, (2015), pp. 1–16.

[4] M. BINK, M. BOER, C. TER BRAAK, J. JANSEN, R. VOORRIPS, AND W. VAN DE WEG, *Bayesian analysis of complex traits in pedigreed plant populations*, Euphytica, 161 (2008), pp. 85–96.

[5] M. BUGALLO, L. MARTINO, AND J. CORANDER, *Adaptive importance sampling in signal processing*, Digital Signal Processing, 47 (2015), pp. 36–49.

[6] T. BUI-THANH AND M. GIROLAMI, *Solving large-scale PDE-constrained Bayesian inverse problems with Riemann manifold Hamiltonian Monte Carlo*, Inverse Problems, 30 (2014),

p. 114014.
[7] B. Calderhead, *A general construction for parallelizing Metropolis- Hastings algorithms*, Proceedings of the National Academy of Sciences, 111 (2014), pp. 17408–17413.
[8] O. Cappé, A. Guillin, J. Marin, and C. Robert, *Population Monte Carlo for ion channel restoration*.
[9] O. Cappé, A. Guillin, J. Marin, and C. Robert, *Population monte carlo*, Journal of Computational and Graphical Statistics, (2012).
[10] G. Celeux, J. Marin, and C. Robert, *Iterated importance sampling in missing data problems*, Computational statistics & data analysis, 50 (2006), pp. 3386–3404.
[11] J. Cornuet, J. Marin, A. Mira, and C. Robert, *Adaptive multiple importance sampling*, Scandinavian Journal of Statistics, 39 (2012), pp. 798–812.
[12] C. Cotter and S. Reich, *Ensemble filter techniques for intermittent data assimilation-a survey*, in Large Scale Inverse Problems. Computational Methods and Applications in the Earth Sciences, Walter de Gruyter, Berlin, 2012, pp. 91–134.
[13] S. Cotter, M. Dashti, J. Robinson, and A. Stuart, *Bayesian inverse problems for functions and applications to fluid mechanics*, Inverse Problems, 25 (2009), p. 115008.
[14] S. Cotter, G. Roberts, A. Stuart, and D. White, *MCMC methods for functions: modifying old algorithms to make them faster*, Statistical Science, 28 (2013), pp. 424–446.
[15] G. Evensen, *Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics*, Journal of Geophysical Research: Oceans (1978–2012), 99 (1994), pp. 10143–10162.
[16] M. Girolami and B. Calderhead, *Riemann manifold Langevin and Hamiltonian Monte Carlo methods*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73 (2011), pp. 123–214.
[17] H. Haario, E. Saksman, and J. Tamminen, *Componentwise adaptation for high dimensional MCMC*, Computational Statistics, 20 (2005), pp. 265–273.
[18] W. Hastings, *Monte Carlo sampling methods using Markov chains and their applications*, Biometrika, 57 (1970), pp. 97–109.
[19] T. House, A. Ford, S. Lan, S. Bilson, E. Buckingham-Jeffery, and M. Girolami, *Bayesian uncertainty quantification for transmissibility of influenza, norovirus and ebola using information geometry*, Journal of The Royal Society Interface, 13 (2016), p. 20160279.
[20] M. Isard and A. Blake, *Condensationconditional density propagation for visual tracking*, International journal of computer vision, 29 (1998), pp. 5–28.
[21] C. Ji and S. C. Schmidler, *Adaptive Markov Chain Monte Carlo for Bayesian Variable Selection*, Journal of Computational and Graphical Statistics, 22 (2013), pp. 708–728.
[22] J. Liu, *Monte Carlo strategies in scientific computing*, Springer Science & Business Media, 2008.
[23] J. Liu, F. Liang, and W. Wong, *The multiple-try method and local optimization in Metropolis sampling*, Journal of the American Statistical Association, 95 (2000), pp. 121–134.
[24] J.-M. Marin, K. Mengersen, and C. Robert, *Bayesian modelling and inference on mixtures of distributions*, Handbook of statistics, 25 (2005), pp. 459–507.
[25] L. Martino, V. Elvira, D. Luengo, and J. Corander, *An adaptive population importance sampler: Learning from uncertainty*, IEEE Transactions on Signal Processing, 63 (2015), pp. 4422–4437.
[26] G. Moore et al., *Cramming more components onto integrated circuits*, Proceedings of the IEEE, 86 (1998), pp. 82–85.
[27] R. Neal, *MCMC using ensembles of states for problems with fast and slow variables such as Gaussian process regression*, arXiv preprint arXiv:1101.0387, (2011).
[28] O. Pele and M. Werman, *Fast and robust Earth mover's distances*, in Computer Vision, 2009 IEEE 12th International Conference on, IEEE, 2009, pp. 460–467.
[29] S. Reich, *A nonparametric ensemble transform method for Bayesian inference*, SIAM Journal on Scientific Computing, 35 (2013), pp. A2013–A2024.
[30] C. Robert and G. Casella, *Monte Carlo statistical methods*, Springer Science & Business Media, 2013.
[31] G. Roberts and J. Rosenthal, *Optimal scaling for various Metropolis-Hastings algorithms*, Statistical science, 16 (2001), pp. 351–367.
[32] ———, *Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms*, Journal of Applied Probability, 44 (2007), pp. 458–475.
[33] ———, *Examples of adaptive MCMC*, Journal of Computational and Graphical Statistics, 18 (2009), pp. 349–367.
[34] J. Rosenthal, *Optimal proposal distributions and adaptive MCMC*, Handbook of Markov Chain Monte Carlo, (2011), pp. 93–112.

[35] P. Russell, *PhD Thesis*, PhD thesis, School of Mathematics, University of Manchester, 2017.
[36] J. Sexton and D. Weingarten, *Hamiltonian evolution for the hybrid Monte Carlo algorithm*, Nuclear Physics B, 380 (1992), pp. 665–677.
[37] J. Sirén, P. Marttinen, and J. Corander, *Reconstructing population histories from single nucleotide polymorphism data*, Molecular biology and evolution, 28 (2011), pp. 673–683.
[38] A. Solonen, P. Ollinaho, M. Laine, H. Haario, J. Tamminen, H. Järvinen, et al., *Efficient MCMC for climate model parameter estimation: parallel adaptive chains and early rejection*, Bayesian Analysis, 7 (2012), pp. 715–736.
[39] A. Stuart, *Inverse problems: a Bayesian perspective*, Acta Numerica, 19 (2010), pp. 451–559.