PARALLEL ADAPTIVE IMPORTANCE SAMPLING

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

Abstract. Markov chain Monte Carlo methods are commonly used and powerful family of numerical methods for sampling from complex probability distributions. As the applications that these methods are being applied to 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 MCMC algorithms. At each iteration, an importance sampling proposal distribution is formed using the current states of all of the chains within an ensemble. Once weighted samples have been produced from this, 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 using the same number of processors. This demonstrates the added

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

1. Introduction.

Markov chain Monte Carlo (MCMC) methods are a powerful family of tools that allow us to sample from complex probability distributions. MCMC methods were first developed in the 70s [11], and with the development of faster more powerful computers, have become ever more important in a whole range of fields in statistics, science and engineering. However, despite the advances in hardware that have made it possible to fully characteries the posterior distributions of many problems, it remains unfeasible to use MCMC methods for many more. In particular, when considering Bayesian inverse problems, each MCMC step may involve the numerical solution of one or more PDE. As many samples are usually required before Monte Carlo error is reduced to acceptable levels, these types of problem remain frustratingly out of our grasp.

Many advances have been made in the field of MCMC, to design ever more complex methods, which more intelligently propose moves which leads to faster converging methods. Function space versions of standard methods such as the random walk Metropolis-Hastings (RWMH) algorithm or the Metropolis adjusted Langevin algorithm (MALA), whose convergence rates are independent of dimension have been developed [6]. The hybrid (or Hamiltonian) Monte Carlo (HMC) method uses Hamiltonian dynamics in order to propose and accept moves to states which are a long way away from the current position [24], and function space analogues of this have also been proposed [1]. Riemannian 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 [9]. This methodology has been successfully applied to MALA-type proposals and methods which exploit even higher order gradient information [2]. These methods allow us to more fully explore the posterior distribution, at the cost of fewer iterations of the method.

Simultaneously, great strides are continually being made in the development of computing hardware. Moore's law, which predicted that the number of transistors that it is possible to fit on a single microchip will double every two years, has been largely fol-

[*]Department of Mathematics, Imperial College, London, UK

[†]School of Mathematics, University of Manchester, Manchester, UK. e: simon.cotter@manchester.ac.uk

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

lowed since the early 70s [17]. However, the main revolution with regards to scientific computing in recent decades has been the increasing focus on parallel architectures. The efficient exploitation of these facilities is the key to solving many of the computational challenges that we currently face.
As such, the development of efficient parallel MCMC algorithms is an important area for research. Since MCMC methods can be trivially parallelised by simply running many independent chains in parallel, the focus needs to be on the development of methods which gain some added benefit through parallelism. One class of parallel MCMC method uses multiple proposals, with only one of these proposals being accepted. Examples of this approach include multiple try MCMC [15] and ensemble MCMC [18]. In [3], 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.
In this paper, we present a framework for parallelisation of importance sampling, which allows us to use any of the current Metropolis-based methodologies in order to create an efficient target proposal from the current state of all of the chains in the ensemble. In Section **??** we outline some preliminaries, including the general set up for Bayesian inverse problems, the preconditioned Crank-Nicolson Langevin (pCNL) algorithm and a brief review of the optimal transport resampler, both of which we will be employing within the algorithm. In Section **??**, we explore the common problems with MCMC methods, which we intend to address with the development of our new framework. We describe this framework in Section **??**. In Section **??**, we present some numerical experiments which demonstrate the savings that can be made by employing this approach as opposed to a naive/trivial parallelisation of existing MCMC methods. In Section **??**, we will summarise our results and suggest some areas for future investigation.

2. Preliminaries.

2.1. Bayesian inverse problems.

In this paper, we particularly focus on the use of MCMC methods in characterising posterior probability distributions in Bayesian inverse problems. We wish to learn about a particular unknown quantity u, of which we are able to make direct or indirect noisy observations. For now we say that u is a member of a Hilbert space X.
The parameter u is observed through the observation operator $\mathcal{G} : X \to \mathbb{R}^d$. Since observations are rarely, if ever, perfect, we assume that these measurements D are subject to Gaussian noise, so that

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

For example, if u are the rates of reactions in a chemical system, $\mathcal{G}$ might return the times at which each reaction occurs, or some summary of this information.
These modelling assumptions allow us to construct the likelihood of observing the data D given the parameter $u = u^*$. Rearranging (2.1) and using the distribution of ε, we get:

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

where $\|x - y\|_\Sigma$ is the Mahalanobis distance between x and y.
As discussed in [5,27], 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

```
$X = x_0$
for $i = 1, 2, 3, \dots$ do
    $Y = (2+\delta)^{-1}\left[(2-\delta)X_{i-1} - 2\delta\mathcal{C}\nabla\Phi(u) + \sqrt{8\delta}W\right]$, $W \sim \mu_0$
    $a(X_{i-1}, Y) = \min\{1, \exp(\Phi(X_{i-1}) - \Phi(Y))\}$.
    $u \sim U([0,1])$
    if $u < a(X_{i-1}, Y)$ then
        $X_i = Y$
    else
        $X_i = X_{i-1}$
    end if
end for
```

Table 2.1

A pseudo-code representation of the preconditioned Crank-Nicolson Langevin (pCNL) algorithm. $\delta \in (0, 2]$ is a step size parameter.

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:

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

2.2. The preconditioned Crank-Nicolson Langevin (pCNL) algorithm. In recent years, work has been carried out to frame MCMC proposal distributions on function space [6]. These new discretisations perform comparably with the original versions in low dimensions, so we choose one of these distributions to consider. If gradient information regarding the observation operator is available, then a range of MCMC methods are available which exploit this information to improve mixing rates. One example of such an algorithm is MALA. In [6], a function space version of this method was presented, the pCNL algorithm, and is described in full in Table 2.1. The proposal used in this method comes about through a Crank-Nicolson approximation of the Langevin SDE, whose invariant measure is the posterior measure μ_Y.

2.3. Particle filters and resamplers. In several applications, data must be assimilated in an "online" fashion, with up to date observations of the studied system being made available on a regular basis. In these contexts, such as in weather forecasting or oceanography, data is incorporated using a filtering methodology. One popular filtering method is the particle filter, the first of which was dubbed the Bootstrap filter [10]. In this method, a set of weighted particles is used to represent the posterior distribution. The positions of the particles are updated using the model dynamics. Then, when more observations are made available, the relative weights of the particles are updated to take account of this data, using Bayes' formula. Other filtering methods, such as the Kalman filter [13] and ensemble Kalman filter [7], have also been developed which are often used within the data assimilation community.
One advantage of the particle filter is that there are convergence results for this method as the number of particles is increased. The downside is that, the effective sample size decreases at each iteration. One way to tackle this is to employ a resampling scheme. The aim of a successful resample is to take your unevenly weighted

ensemble and return a new ensemble of particles with even weights which is highly correlated to the original samples.

The Ensemble Transform (ET) method proposed by Reich [20] makes use of optimal transportation as described in [28, 29]. The transform takes a sample of weighted particles $\{y_i\}_{i=1}^{M}$ from μ_Y and converts it into a sample of evenly weighted particles $\{x_i\}_{i=1}^{M}$ from μ_X, by means of defining a coupling T^* between Y and X. Given that a trivial coupling T^t always exists in the space of transference plans, $\Pi(\mu_X, \mu_Y)$, we can find a coupling T^* which maximises the correlation between X and Y [4]. This coupling is the solution to a linear programming problem in M^2 variables with $2M-1$ constraints as defined in [20]. Maximising the correlation ensures that the new sample is as much like the original sample as possible with the additional property that the sample is evenly weighted.

As proposed in [20] a Monte Carlo algorithm can be implemented to resample from a weighted ensemble. We create a weighted sample, then solve the optimal transport problem which produces the coupling described above, we can draw a new sample from the evenly weighted distribution. Reich suggests using the mean of the evenly weighted distribution to produce a consistent estimator.

Analysis of this method shows that as the ensemble size increases, the statistics of the evenly weighted sample approach those of the posterior distribution, at least well enough for a proposal distribution as described in Section 3. The histogram of the evenly weighted sample exhibits small oscillations in the tails of the posterior, and also struggles to deal with discontinuities.

Ideally the dimension of particles in a filter will not affect the convergence of the method. Unfortunately, the approximation of continuous density functions is computationally infeasible in high dimensions and the required ensemble size scales exponentially with the dimension of the problem [25, 26].

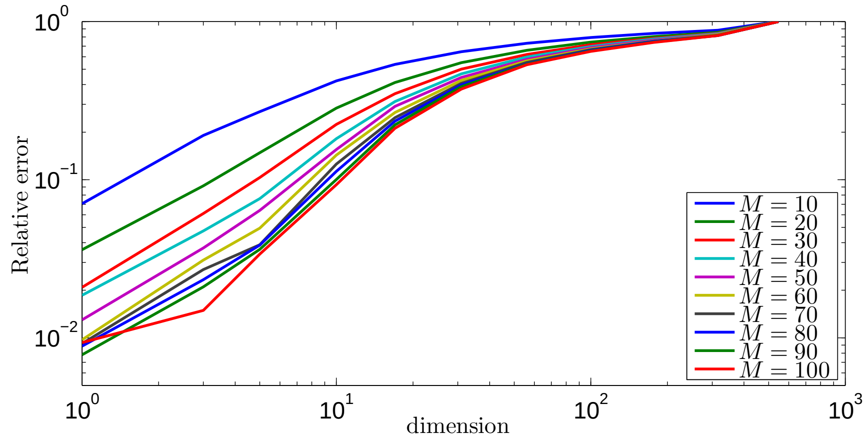

Fig. 2.1. *Demonstration of the ETMC algorithm rapidly failing as the dimension increases. The relative error shown is that between the sampled mean and the true mean.*

Figure 2.1 shows how quickly the ETMC algorithm loses its ability to accurately represent the mean of the posterior distribution as the dimension increases. Clearly for a simple Gaussian problem, we can not resample in much more than 10 dimensions and hope to recover a representative sample.

2.4. Deficiencies of Metropolis-type MCMC schemes. All MCMC methods are trivially parallelisable. One can take a method and simply implement it simultaneously over a set of processes. All of the states of all of the processes can be recorded, and in the time that it takes one process to draw N samples, M processes can draw NM samples.
However, we argue that this is a far from optimal scenario. First of all, unless we have a lot of information about the posterior, we will begin the algorithm a long way from statistical equilibrium. Some initial iterations then are not samples from the posterior, and must be thrown away. This process is known as the burn-in. In a trivially parallelised scenario, each process must perform this process independently, while also tuning the scaling parameter.
Moreover, many MCMC algorithms suffer from poor mixing, especially in multimodal systems. The time 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 trivially parallelised MCMC scheme is exactly that - trivial. Intuition suggests that we can gain some speed up by sharing information across the processes and that is exactly what we wish to demonstrate in this paper.
These deficiencies of the trivial 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. We will then introduce the version that we have implemented, which utilises the resampler recently suggested by Reich [20] described in Section 2.3, and the pCNL proposal distribution.

3. The Parallel Adaptive Importance Sampler (PAIS). In a trivially parallelised MCMC scheme, if we have enough processes all sampling from the target density at the same time, then we can make a very rough approximation of that density by taking the current states of all of the processes as a representative sample. In the PAIS, the proposal density for the importance sampler stage is a mixture of all of the proposal densities for a chosen MCMC scheme, using the current states. If the proposal in the MCMC scheme is appropriate and our number of samples is big enough, we might hope that this mixture of densities is a good enough approximation of the target density, and that new samples drawn from the mixture proposal are approximately draws from the target measure. If this is the case, then the variance of importance weights for these samples will be small, leading to a highly efficient sampler.
The problem at this stage, is that our current states all have varying importance, but in order for us to iterate this whole process again, we require them to be the same. Therefore, we employ a resampler to take our new set of samples with differing weights, and return a set of states with equal weights, which are chosen so that the correlation between the two sets is maximised. At this stage we can start the whole process again. The algorithm is given in more detail in Table 3.1.
We wish to sample states $x \in X$ from a posterior probability distribution μ_Y. Since we have M processes, we represent the current state of all of the Markov chains as a vector $\mathbf{X} = [x_1, x_2, \dots, x_M]^T$. We are also given a transition kernel $\nu(\cdot, \cdot)$, which

```
$\mathbf{X}^{(0)} = \mathbf{X}_0 = [x_1^{(0)}, x_2^{(0)}, \dots, x_M^{(0)}]^T$
for $i = 0, 1, 2, ..., N$ do
    $\mathbf{Y}^{(i)} = [y_1^{(i)}, y_2^{(i)}, \dots, y_M^{(i)}]^T, \quad y_j^{(i)} \sim \nu(\cdot; x_j^{(i)})$
    $\chi(y; \mathbf{X}^{(i)}) = \frac{1}{M}\sum_{j=1}^{M} \nu(y; x_j^{(i)})$.
    $\mathbf{W}^{(i)} = [w_1^{(i)}, w_2^{(i)}, \dots, w_M^{(i)}]^T, \quad w_j^{(i)} = \frac{\pi(y_j^{(i)})}{\chi(y_j^{(i)}; \mathbf{X}^{(i)})}$.
    Resample: $(\mathbf{W}^{(i)}, \mathbf{Y}^{(i)}) \to (\frac{1}{M}\mathbf{1}, \mathbf{X}^{(i+1)})$
end for
```

Table 3.1

A pseudo-code representation of the Parallel Adaptive Importance Sampler (PAIS).

might come from an MCMC method, for example.
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 stored.
The key here is to choose a suitable transition kernel ν such that if $X^{(i)}$ is a decent representative sample of the posterior, the mixture density $\chi(\cdot; \mathbf{X}^{(i)})$ is a reasonable approximation of the posterior distribution. If this is the case, the newly proposed states $\mathbf{Y}^{(i)}$ will also be a good (and relatively independent) sample of the posterior with low variance in the weights $\mathbf{W}^{(i)}$.
In Section 5, we will demonstrate how the algorithm performs, using pCNL algorithm. We do not claim that this choice is optimal, but is simply chosen as an example to show that sharing information across processes can improve on the original MCMC algorithm and lead to convergence 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.

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. One significant difficulty with this approach, as with many kernel approximation approaches, is finding an appropriate ν such that χ is a close approximation to the posterior density π in the right situation. If the proposal distribution is too overdispersed, then the algorithm will propose states a long way from the current state but there will be a relatively low acceptance rate, impacting the quality of inferences on the posterior. Similarly, if the posterior is under-dispersed, the process will take a long time to fully explore the space so the mixing rate, and hence convergence rate, will be slow. It is therefore necessary to find a proposal distribution which is slightly overdispersed to ensure the entire posterior is explored [8], but is as close to the posterior as possible.
Proposal distributions which are slightly over-dispersed as described above, can be found by tuning the variance of the proposal distribution during the burn-in phase of the algorithm. Algorithms which use this method to find optimal proposal distributions are known as adaptive MCMC algorithms. Adaptive MCMC algorithms will converge to the stationary distribution $\pi(\cdot)$, irrespective of whether the adaptive parameter itself converges to an optimal value, under the following conditions [22,23].
Condition 1: The adaptive parameter exhibits diminishing adaptation, which says that the amount of movement in the adaptive parameter decreases with the length of

the chain.
Condition 2: The smallest number of steps for which the kernel has sufficiently converged, from an initial state x, is finite, by bounded covergence.
Adaptively choosing δ with a large initial guess allows us to first explore the state space, searching for multiple modes, then reducing δ to an optimal value for efficient sampling helps combat these problems. This process is effective in the PAIS algorithm since tuning δ using all of the chains is faster than in a single chain. The resampling step causes the processes to spread out across the regions of high density so that when δ is reduced for locally efficient sampling, it is not required for the chains to constantly search out new modes.
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 we have 23.4% for RWMH and 57.4% for MALA [21]. Unlike Metropolis-Hastings algorithms, the PAIS algorithm does not accept or reject proposed values, so we need another method of measuring the optimality of δ. Section 4.1 gives some possible methods of tuning δ.

4.1. Statistics for Determining the Optimal Scaling Parameter.

4.1.1. Determining optimal scaling parameter δ using error analysis.

MCMC algorithms can be assessed by comparing their approximation of the posterior to the analytic distribution. To assess this, a distance metric must be decided on, such as the relative error between the sample moments and the true moments, or the relative L2 error between the true density, $\pi(x|D)$, and the constructed histogram. The relative error in the m-th moment is

$$\left| \frac{N^{-1} \sum_{i=1}^{N} x_i^m - \mathbb{E}[X^m]}{\mathbb{E}[X^m]} \right|,$$

where $\{x_i\}_{i=1}^N$ is a sample of size N produced by the algorithm. The relative L2 error between a continuous function to a piecewise constant function, e, is

$$e^2 = \sum_{i=1}^{n_b} \left[\int_{R_i} \pi(a|D) \, \mathrm{d}a - vB_i \right]^2 \Big/ \sum_{i=1}^{n_b} \left[\int_{R_i} \pi(a|D) \, \mathrm{d}a \right]^2, \tag{4.1}$$

where the points $\{R_i\}_{i=1}^{n_b}$ are the d-dimensional regions the histogram is defined over, 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 a bin, and B_i is the value of the ith bin.
These statistics are usually not practical for finding optimal values of δ since they require knowledge of the analytic solution, and require that the algorithm be run for a long time to build up a sufficiantly large sample.
Another related statistic, is to use the variance of the estimate of the first moment $\hat{\mu}$. Assuming that the algorithm is converging to the invariant distribution, convergence will be fastest when the variance of the estimate is minimised. This fact follows from the convergence rate of Monte Carlo algorithms, $\sigma/\sqrt{n}$. The proposal distribution which minimises the variance, σ^2, of $\hat{\mu}$, is the most desirable. This variance statistic often converges faster than the moment itself, and does not require any knowledge of the true value of the moment.
The following statistics can be used for importance sampling algorithms.

4.1.2. Determining optimal δ using the variance of the weights. Importance samplers assign a weight to each sample they produce, based on a ratio of the posterior to the proposal at that point. This type of sampler is most efficient when the posterior is proportional to the proposal distribution, in this case the weights are constant, and so the variance of the weights, $var(w(y))$, is zero. Hence, the optimal value of δ, is

$$\delta^*_{\text{var}} = \arg\min_{\delta} \text{var}(w(y)).$$

The mean of the estimator $\text{var}(w(y))$ is a smooth function, and so it is used to tune δ during the burn-in phase of MCMC algorithms. The variance of the estimator can be large, especially far away from the optimal value, so it can take a large number of samples to calculate descent directions.

4.1.3. Determining optimal δ using 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 are unlikely to achieve a ratio of exactly 1.
The effective sample size can be defined in the following ways:

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

From the last definition we can see that when the variance of the weights is zero, which is our ideal scenario, $n_{\text{eff}} = M$. Also, in a neighbourhood around δ^*_{var}, n_{eff} decreases, when the variance increases. So if $v(w(y))$ can be equal to zero for a particular problem and proposal distribution, maximising the effective sample size is equivalent to minimising the variance of the weights. In the PAIS algorithm we have a dynamic proposal which is perturbed at every iteration. Hence, our proposal distribution cannot be exactly proportional to the posterior.
The statistic n_{eff} converges faster than the variance of the weights, and so is preferable as a means of tuning δ. While δ^*_{var} and the optimal value found using the effective sample size, δ^*_{eff}, coincide when $v(w(y)) = 0$, this is not usually possible, so the methods will find different optimal values of δ^*.
The n_{eff} statistic also has another useful property; if we imagine the algorithm in the burn-in phase, for example, we have M processes in the tail of a Gaussian curve searching for the area of high density. If the processes are evenly spaced, then the particle closest to the mean will have an exponentially higher weight assigned to it. The effective sample size in this scenario will be close to 1. As the algorithm burns in, the processes find the flatter area near the mean so the number of samples contributing information to the posterior will increase. By this argument we can see that rising n_{eff} signals the end of the burn-in period.

4.1.4. Behaviour of the effective sample size for varying ensemble size. If we are to use the effective sample size ratio, we need to look at how it behaves in different situations. In the next section we see how it behaves for different observation operators. Here we look at how the statistic behaves as we vary the ensemble size, M. Figure 4.1 shows results for the second problem discussed in 5.2 when using the PAIS-pCNL algorithm to explore the posterior distribution. The left part of the

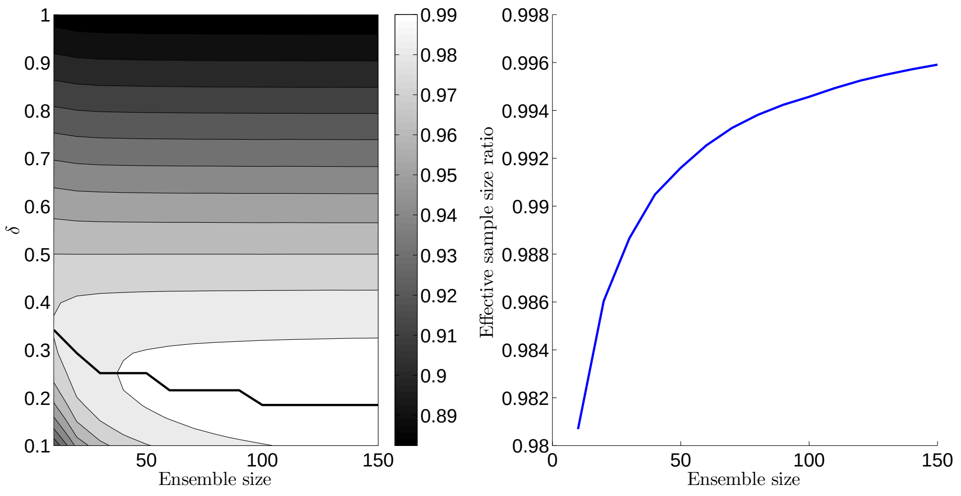

FIG. 4.1. *Left: A contour plot showing how δ_{eff} varies with the ensemble size. The black line highlights the optimal value of δ. Right: The value of the effective sample size ratio for the optimal δ at each ensemble size. i.e. the value of the effective sample size ratio along the black line in the left figure. Set up for this problem is given in Section 5.2.*

figure shows that as the ensemble size increases, the value of δ which gives the optimal effective sample size ratio decreases. This is to be expected since we are trying to fit more kernels into the posterior distribution, so the variance of each kernel needs to be decreased.

4.2. The Adaptive Algorithm. A popular method for an adaptive MCMC algorithm is to view the scaling parameter as a random variable which we can sample during the course of the MCMC iterations. However, it can be slow and we may need an uninformative prior. Alternatively, the parameter may be randomly sampled from some interval at various points during the chain, a benefit being that it allows some exploration of the state space but never converges to an optimal value of the scaling parameter. We choose to use a divide and conquer scheme which optimises the effective sample size. Some more sophisticated examples are described in [23] and [12]. The proposed algorithm, is not presented as an optimal strategy but as an example of the benefits of tuning the algorithm using the effective sample size statistic.
In the adaptive algorithm described in Table 4.1 we calculate a sequence $\{\delta^{(k)}\}_{k=1}$ which converges to the optimal scaling parameter δ^*, resulting in the optimal transition density χ for our MCMC algorithm. We must choose some sequence of iterations at which to update $\delta^{(j)}$, $\{n_k\}_{k=1}$, which can be decided using a sequence in which the terms grow exponentially further apart. The algorithm is given for the PAIS method but a similar method is used to adaptively calculate δ^* for the pCNL algorithm.
When implemented, there are a number of parameters which must be chosen to allow efficient tuning of δ. Firstly, the initial value of δ should be chosen so that the chains spread out quickly across the state space. If we always choose $\delta^{(0)} = 2$ the chains may miss the high density region completely, so a smaller value of δ must be chosen. This can be seen for example in the optimisation graph for the problem described in Section 5.1, Figure **??**, where there is a sudden jump at $\delta \approx 0.1$.
Another choice to be made is which statistic $T(\delta)$ to optimise. In the examples which

```
$\mathbf{X}^{(0)} = \mathbf{X}_0 = [x_1^{(0)}, x_2^{(0)}, \dots, x_M^{(0)}]^T$
Choose $\delta^{(0)} \in (0,2]$, $\delta_L^{(0)} = 0.99\delta^{(0)}$, $\delta_U^{(0)} = 1.01\delta^{(0)} \wedge 2$.
for $i = 0, 1, 2, \dots, N$ do
    Sample $y_j^{(i)} \sim \nu(\cdot; x_j^{(i)}, \delta_L^{(i)})$ for $j = 1, \dots, M/2$.
    Sample $y_j^{(i)} \sim \nu(\cdot; x_j^{(i)}, \delta_U^{(i)})$ for $j = M/2+1, \dots, M$.
    $\chi(y; \mathbf{X}^{(i)}) = M^{-1}\left(\sum_{j=1}^{M/2} \nu(y; x_j^{(i)}, \delta_L^{(i)}) + \sum_{j=M/2+1}^{M} \nu(y; x_j^{(i)}, \delta_U^{(i)})\right)$.
    $\mathbf{W}^{(i)} = [w_1^{(i)}, w_2^{(i)}, \dots, w_M^{(i)}]^T, \quad w_j^{(i)} = \frac{\pi(y_j^{(i)})}{\chi(y_j^{(i)}; \mathbf{X}^{(i)})}$.
    if $i$ is in $\{n_k\}_{k=1}$ then
        For $w_{kj} = w_j^{(k)}$, $N$ iterations since last update,
        $T_L = NM(\sum_{k=i-N}^{N} \sum_{j=1}^{M/2} w_{kj})^2 / (\sum_{k=i-N}^{N} \sum_{j=1}^{M/2} w_{kj}^2)$.
        $T_U = NM(\sum_{k=i-N}^{N} \sum_{j=M/2+1}^{M} w_{kj})^2 / (\sum_{k=i-N}^{N} \sum_{j=M/2+1}^{M} w_{kj}^2)$.
        $\delta^{(i+1)} = \delta^{(i)} - \Delta\mathcal{D}T$.
    else
        $\delta^{(i+1)} = \delta^{(i)}$.
    end if
    if $i < K$ then
        Resample: $(w_j^{(i)}, y_j^{(i)})_{j=1}^{M/2} \to (\frac{1}{M/2}, x_j^{(i+1)})_{j=1}^{M/2}$
        Resample: $(w_j^{(i)}, y_j^{(i)})_{j=M/2+1}^{M} \to (\frac{1}{M/2}, x_j^{(i+1)})_{j=M/2+1}^{M}$
    else
        Resample: $(w_j^{(i)}, y_j^{(i)})_{j=1}^{M} \to (\frac{1}{M}, x_j^{(i+1)})_{j=1}^{M}$
    end if
end for
```

Table 4.1
A pseudo-code representation of the adaptive PAIS algorithm.

follow, we have optimised the naively parallelised pCNL algorithm using the optimal acceptance rate $\hat{\alpha} = 0.75$. This value was found by using the L2 error minimums to calculate δ^*. This means that we are minimising the statistic

$$T_{\mathrm{MH}}(\delta) = \left| \frac{N_{\mathrm{acc}}(\delta)}{N_{\mathrm{total}}} - \hat{\alpha} \right|,$$

where $N_{\mathrm{acc}}(\delta)$ is the number of accepted moves and N_{total} is the total number of samples produced. For the PAIS algorithm, we are maximising the effective sample size as discussed in Section 4.1.3, so $T_{\mathrm{PAIS}}(\delta) = n_{\mathrm{eff}}(\delta)$.
In the numerics that follow in Section 5, we utilise the adaptive scheme in Table 4.1 for finding the optimum value of δ in the proposal step in the PAIS.

5. Numerical Examples.

5.1. Sampling from a one dimensional Gaussian distribution. In this example we look at how PAIS compares to naively parallelised MCMC. We compare the pCNL algorithm [6] with its PAIS variant, the PAIS-pCNL algorithm. We consider several statistics for measuring the efficiency of the PAIS algorithm compared to existing Metropolis-Hastings algorithms when applied to a simple one dimensional Gaussian posterior.

5.1.1. Target distribution. Consider the simple case of a linear observation operator $\mathcal{G}(u) = u$, where the prior on u and the observational noise follow a Gaussian distribution. Then, following Equation 2.2, the Gaussian posterior has the resulting form

$$\text{law}(\mu_Y) = \pi(u|D) \propto \exp\left(-\frac{1}{2}\left\|u - D\right\|_{\Sigma}^{2} - \frac{1}{2}\left\|u\right\|_{\mathcal{T}}^{2}\right), \tag{5.1}$$

where Σ and $\mathcal{T}$ are the covariances of the observational noise and prior distributions respectively. In the numerics which follow, we choose $\Sigma = \sigma^2 I_N$ and $\mathcal{T} = \tau^2 I_d$ with $\tau^2 = 2$ and $\sigma^2 = 0.1$, and we observe $u = -2.5$ with noise drawn from the prior, resulting in $D = -2.6738662$. These values result in a posterior density in which the vast majority of the density is contained inside the high density region of the prior. This means that it should be straightforward for the algorithm to find the stationary distribution. The Kullback-Leibler (KL) divergence for this problem is $D_{KL}(\mu_Y||\mu_0) = 2.67$.

5.1.2. Numerical implementation. In each of the following simulations, we perform three tasks. First we calculate the optimal value of δ by optimising the statistics described in Section 4.1. Once we have these parameters, we run the algorithms and compare the convergence speeds of the algorithms. Finally, we implement the adaptive algorithms described in Section 4.2 and compare the convergence of these algorithms with the nonadaptive algorithms.

(1) Finding the optimal parameters: To find the optimal parameters we choose 32 values of δ evenly spaced in $[10^{-5}, 2]$. We run the PAIS-pCNL algorithm for 10,000 iterations and pCNL for 100,000 iterations, each with $M = 50$ processes. We took 32 repeats of both algorithms and then used the medians of the statistics to find the optimal parameters.

(2) Measuring convergence of nonadaptive algorithms: We run the algorithms in Section 5.1 and Section 5.2 for 10 million iterations, again with 50 processes. The algorithms are run using the optimal parameters found in (1). The relative L2 error (Equation 4.1) is used as a measure of accuracy. The simulation for each algorithm was repeated 24 times.

(3) Measuring convergence of adaptive algorithms: We run the adaptive algorithms under the same conditions as the nonadaptive algorithms, and again use the relative L2 error to compare efficiency. The initial value of δ is given in the discussion of each simulation.

Statistic	pCNL
δ^*_{L2}	3.7e-3
$\delta^*_{\text{var}(\hat{\mu})}$	5.8e-2
Acceptance Rate (δ^*_{L2})	9.9e-1
Acceptance Rate ($\delta^*_{\text{var}(\hat{\mu})}$)	7.4e-1

Statistic	PAIS-pCNL
δ^*_{eff}	1.5e-2
$\delta^*_{\text{var}(w(y))}$	6.4e-2
δ^*_{L2}	1.7e-2

Table 5.1
Optimal values of δ summarised from Figure 5.1. Statistics calculated as described in Section 4.1.

5.1.3. Numerical optimal values of δ. Figure 5.1 (left) shows the two values of δ which may be optimal for the pCNL algorithm. The first estimate comes from the relative L2 error, and the second comes from the variance of the estimate of the mean. Heuristically, the optimal acceptance rates of the function space algorithms

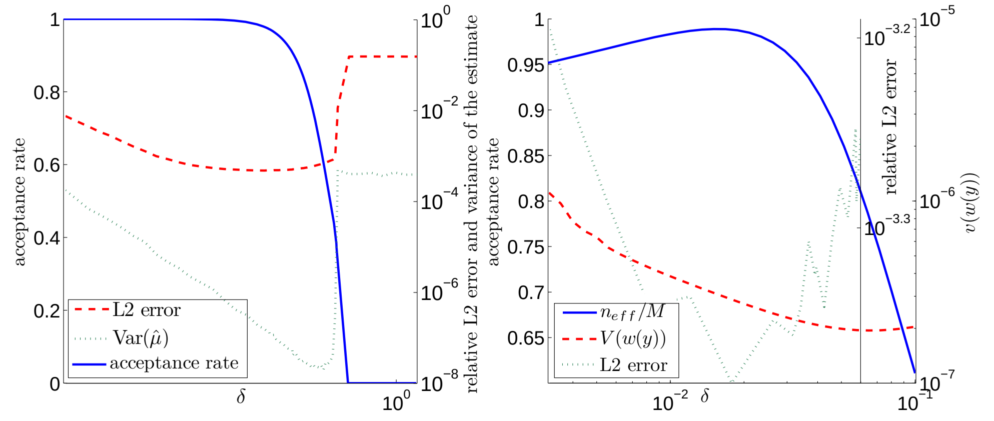

FIG. 5.1. *Finding optimal values of δ for the pCNL (left) algorithm and PAIS-pCNL (right) algorithm for the problem in Section 5.1. The setup is as in Section 5.1.2.*

are expected to be slightly higher than their finite dimensional versions. In light of this, it is clear that the variance of the estimate of the mean is a better indicator of the scaling parameter. The results in Figure 5.1 are summarised in Table 5.1 (left). Figure 5.1 (right) shows the effective sample size ratio compared to the error analysis and the variance of the weights. It is clear to see that the effective sample size does a much better job of finding the optimal value of δ as given by the error analysis than the variance of the weights does.

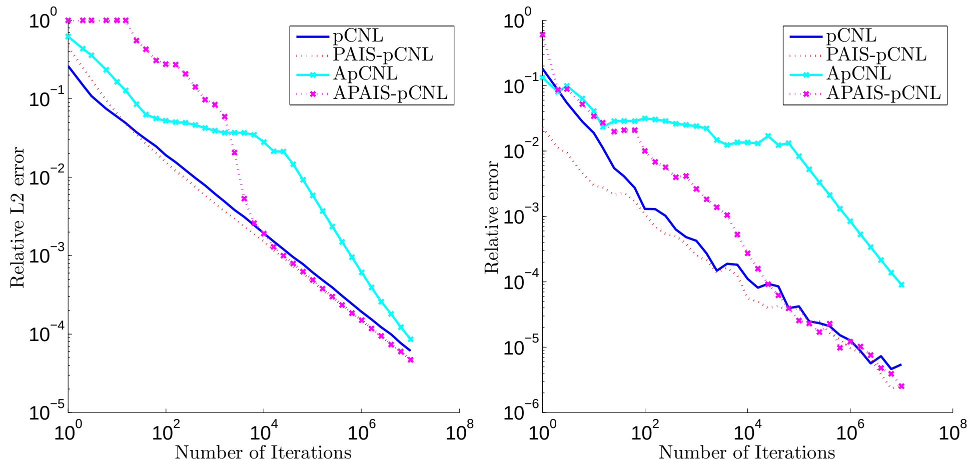

FIG. 5.2. *Relative error in the moments (right) and histograms (left) produced by the (A)pCNL and (A)PAIS-pCNL algorithms against iterations for problem 5.1. The setup is as in Section 5.1.2 (2, 3).*

5.1.4. Convergence of pCNL vs PAIS-pCNL. Figure 5.2 shows that the PAIS-pCNL algorithm converges to the posterior distribution faster than the standard function space pCNL algorithm, in both L2 error and relative error in the moments.

To reach a desired tolerance, after convergence the PAIS-pCNL algorithm requires 40% fewer iterations than the pCNL algorithm.
The adaptive algorithms are run with initial values of $\delta_0 = 0.1$ for the APAIS-pCNL algorithm, and $\delta_0 = 2$ for the ApCNL algorithm. From Figure 5.2 we can see that the APAIS-pCNL algorithm converges at least as quickly as the PAIS-pCNL algorithm. The ApCNL algorithm initially has trouble converging to the posterior; during the initial burn-in phase, some chains find themselves a long way out in the tails where due to the high gradient they will overshoot the high density region and reject almost all proposed values.

5.1.5. Scaling of the PAIS algorithm with ensemble size. Throughout this paper, we use an ensemble size $M = 50$, but it is interesting to see how the PAIS algorithm scales if we were to increase the ensemble size, and if there is some limit below which the algorithm fails. We implement the problem in Section 5.1, using the pCNL algorithm with ensemble sizes ranging from $M = 2$ up to $M = 494$.

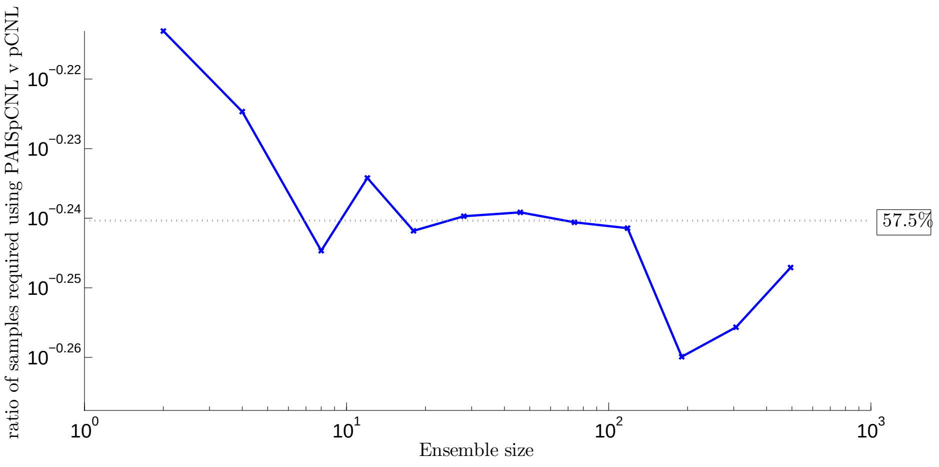

FIG. 5.3. *Ratio of PAIS-pCNL samples required to reach the same tolerance as the pCNL algorithm.*

Figure 5.3 was produced using the method of finding optimal δ described in Section 5.1.2(1), then running 32 repeats at each ensemble size. The convergence rates are then found by regressing through the data. The graph is still very noisy but demonstrates that increasing the sample size continues to reduce the number of iterations required. When $M < 8$, the algorithm takes a long time to reach stationarity.

5.2. Sampling from a one dimensional Gaussian with a higher KL divergence. In this example we use the same setup used in Section 5.1. We choose a posterior distribution which has most of its mass far out in the tails of the prior distribution. The KL divergence is $D_{KL}(\mu_Y||\mu_0) = 4.670$. This means that the algorithm will have to "work harder" to find the area of high probability in the posterior density.

5.2.1. Target distribution. As in the previous example, we use the identity observation operator $\mathcal{G}(u) = u$, which results in the Gaussian posterior s in Equation 5.1. However, this time we choose $\sigma^2 = 0.01$ and $\tau^2 = 0.01$ so the posterior distribution is $\mathcal{N}(D/(1+\sigma^2/\tau^2), \tau^2\sigma^2/(\sigma^2+\tau^2)) = \mathcal{N}(D/2, 0.005)$ which for large D

is a long way out in the tail of the prior with a very small variance. For the simulations which follow we observe a reading of $u = 4$, with noise drawn from $\mathcal{N}(0, \sigma^2)$, resulting in $D = 3.9979631942$.

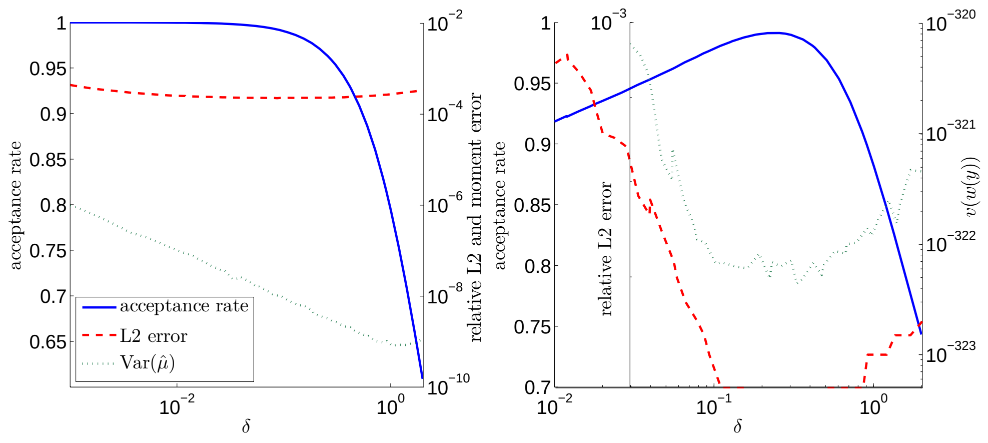

Fig. 5.4. *Finding optimal values of δ for the pCNL (left) algorithm and PAIS-pCNL (right) algorithm for the problem in Section 5.2. The setup is as in Section 5.1.2.*

Statistic	pCNL
δ^*_{L2}	8.6e-2
$\delta^*_{\mathrm{var}(\hat{\mu})}$	9.1e-1
Acceptance Rate (δ^*_{L2})	9.9e-1
Acceptance Rate ($\delta^*_{\mathrm{var}(\hat{\mu})}$)	8.1e-1

Statistic	PAIS-pCNL
δ^*_{eff}	2.6e-1
$\delta^*_{\mathrm{var}(w(y))}$	2.6e-1
δ^*_{L2}	2.8e-1

Table 5.2
Optimal values of δ summarised from Figure 5.4. Statistics calculated as described in Section 4.1.

5.2.2. Numerical optimal values of δ. We find the optimal values of δ using the same methods described in Sections 5.1.2 and 5.1.3. The results are displayed in Figure 5.4 and Table 5.2. There is a huge difference between the two error estimates of δ for pCNL, although the corresponding variance graph is very flat making it sensitive to Monte Carlo error. The variance of the mean estimate has an 81% acceptance rate, which is larger than the pCNL acceptance rate found previously.
For the PAIS-pCNL algorithm, it is again clear that the effective sample size ratio is a useful statistic for judging the optimal value of δ. The relative L2 error estimate of δ^*, shown in Table 5.2, is slightly higher than the other two estimates, but from the graph there again seems to be a fairly flat wide minimum which is in the same region as the optimal effective sample size ratio.

5.2.3. Convergence of pCNL vs PAIS-pCNL. Figure 5.5 that the PAIS-pCNL algorithm, converges to the posterior distribution faster than the pCNL algorithm. The adaptive algorithms both struggle with the first moment for the first million iterations, but produce better estimates after 10 million iterations. We see that to obtain a given tolerance level, the PAIS-pCNL algorithm requires 35% fewer iterations than the pCNL algorithm.

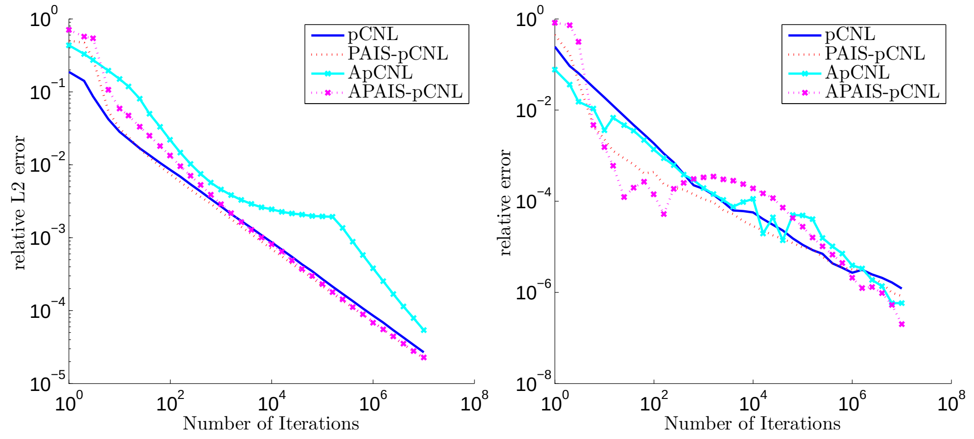

FIG. 5.5. *Relative error in the moments (right) and histograms (left) produced by the (A)pCNL and (A)PAIS-pCNL algorithms against iteratons for problem 5.2. The setup is as described in Section 5.1.2 (2,3).*

5.3. Sampling from Bimodal Distributions. In this section we investigate the behaviour of the PAIS algorithm when applied to symmetric bimodal problems. We see that the resampling step allows the algorithm to redistribute chains to new modes as they are found. This means that we expect the number of chains in a mode to be related to the probability mass in that mode. As a result reconstructed posteriors with disproportional modes, as is familiar with the Metropolis-Hastings algorithms, are not produced. We again look at an 'easy' problem, BM(1), which has a KL divergence of 0.880, and a 'harder' problem, BM(2), which has a KL divergence of 3.647. Problem BM(1) has two modes which are separated by a smaller energy barrier. In BM(2) we increase the distance between the two modes which has the effect of increasing the required energy to jump between modes. These posteriors are shown in Figure 5.6.

5.3.1. Target Distribution. The following setup is the same for both problems. We consider an observation operator $\mathcal{G}(u) = u^2$, and assign the prior $u \sim \mu_0 = \mathcal{N}(0, \tau^2 = 0.25)$. We assume that a noisy reading, D, is taken according to $D = \mathcal{G}(u) + \varepsilon$, where $\varepsilon \sim \mu_\varepsilon = \mathcal{N}(0, \sigma^2 = 0.1)$. This results in the non-Gaussian posterior

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

To create the 'easy' problem we say that the true value of $\mathcal{G}(u) = 0.75$, and the 'hard' problem is generated using $\mathcal{G}(u) = 2$. In the numerics which follow we draw noise from μ_ε to generate our data point, we obtain $D = 0.92131223$ for BM(1) and $D = 1.948664$ for BM(2).

5.3.2. Numerical Implementation. The numerical implementation for most of the following simulations follow the same setup as described in Section 5.1.2. The only exception is that the convergence plots for both adaptive and nonadaptive algorithms are run for 10^6 iterations instead of 10^7.

5.3.3. Calculating values of Optimal δ^*. Calculating the optimal values of the scaling parameters for this problem is similar to the Gaussian case; we check only

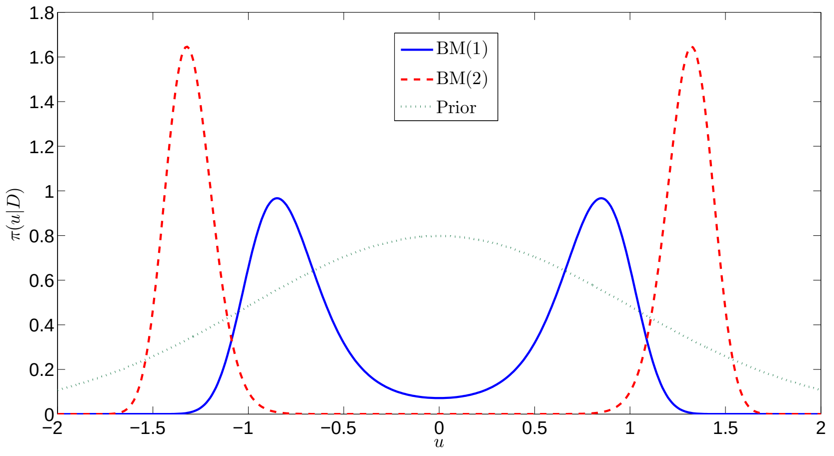

FIG. 5.6. *Posterior distributions for problem BM(1) and BM(2). Problem BM(1) has a noisy data value of 0.921312 which results in an energy barrier which is relatively easy to cross, as well as their common prior distribution.*

the acceptance rate to find the optimal values for pCNL and we use the effective sample size to find the optimal values for PAIS-pCNL. Table 5.3 gives the optimal values of δ for both problems.

Algorithm	δ^*_{acc}	δ^*_{eff}
pCNL	1.9e-1	-
PAIS-pCNL	-	3.9e-2

Algorithm	δ^*_{acc}	δ^*_{eff}	δ^*_{L2}
pCNL	5.8e-2	-	9.1e-1
PAIS-pCNL	-	2.6e-2	2.6e-2

TABLE 5.3
Optimal values of δ for BM(1) (left) and BM(2) (right).

We expect the PAIS algorithm to generally have higher values of the scaling parameter δ. In BM(1), the pCNL algorithm has the higher value of δ, this is because the prior is more efficient to sample from than any single proposal kernel. The PAIS-pCNL algorithm samples more efficiently from its mixture χ.

Problem BM(2) is much harder than BM(1); transitions between the modes are extremely unlikely. This means that we need to consider the density on two levels; we should consider the overall convergence to two correctly proportioned modes as well as local convergence to smooth modes.

To get correctly proportioned modes with the pCNL algorithm it is important that the chains can transition between the modes. Otherwise mode height will be decided by chance. This means that δ must be large. The prior distribution μ_0 is not a good approximation of the posterior distribution in this case leading to an inefficient method of sampling.

We can achieve these two regimes in pCNL by tuning δ using the acceptance rate for local convergence, and by L2 error for global convergence. Similarly in PAIS-pCNL we can use the effective sample size for local convergence, and the L2 error for global convergence.

From Table 5.3 (right) we see that there is a large difference between the optimal value of δ for each regime, meaning that both will result in inefficient sampling. The

PAIS-pCNL algorithm manages to sample the detail and the large scale behaviour with the same value of δ^*: a clear advantage to using this algorithm for this problem.

5.3.4. Convergence of pCNL vs PAIS-pCNL. We see a significant speed with the PAIS-pCNL algorithm for BM(1). Figure 5.7 shows the adaptive and non-adaptive convergence rates.

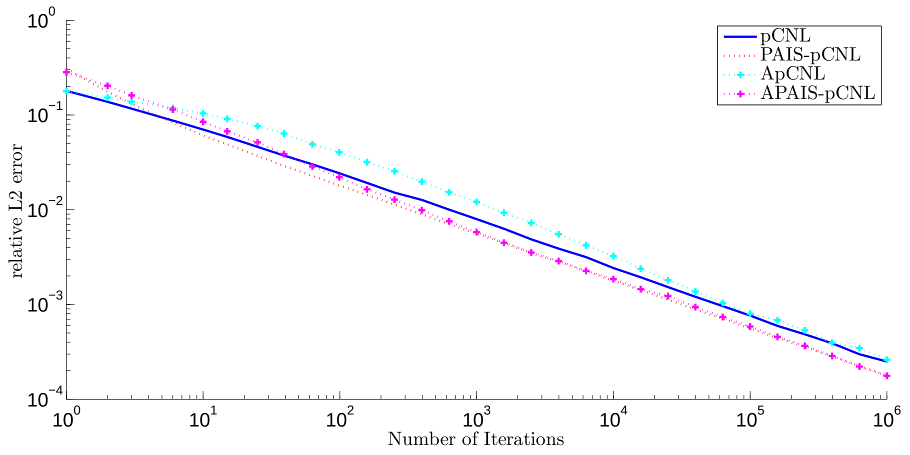

FIG. 5.7. *Error analysis for the PAIS-pCNL and pCNL algorithms. The solid blue line, and dashed red line compare the algorithms with fixed optimal scaling parameters, and the blue crossed and magenta crossed lines compare the adaptive algorithms. The setup is as described in Section 5.3.2 (2,3).*

We can see that the adaptive algorithms compare closely with the respective nonadaptive algorithms and the improvement PAIS offers remains significant. The PAIS-pCNL algorithm requires 45% fewer iterations than the standard pCNL algorithm.
For BM(2) the algorithms are run with the global optimal value of δ^*, and with the local optimal value of δ^*. Figure 5.8 shows that the algorithms using the globally optimal δ^* convergence diagnostics converge slowly towards the true posterior after a long burn-in period, whereas the algorithms using the local optimal δ^* converge quickly, but get stuck in one mode meaning that the convergence rate flattens out.
The adaptive algorithm as stated in Section 4.2 is one way of combining the two regimes. Another method uses a small number of 'scout' chains with a large δ to continually search out new modes. Other methods of mode searches are described in [14], and the regeneration method is applicable [19].
Figure 5.9 shows the success of the (A)PAIS-pCNL algorithms in converging to the posterior, compared to the (A)pCNL algorithms. We can see that using the pCNL algorithm for this problem would be infeasible.

5.3.5. A useful property of the PAIS algorithm for multimodal distributions. The biggest issue for the Metropolis-Hastings algorithms when sampling from a posterior such as the one in BM(2) is that it is unlikely that the same number of chains will occur 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 ap-

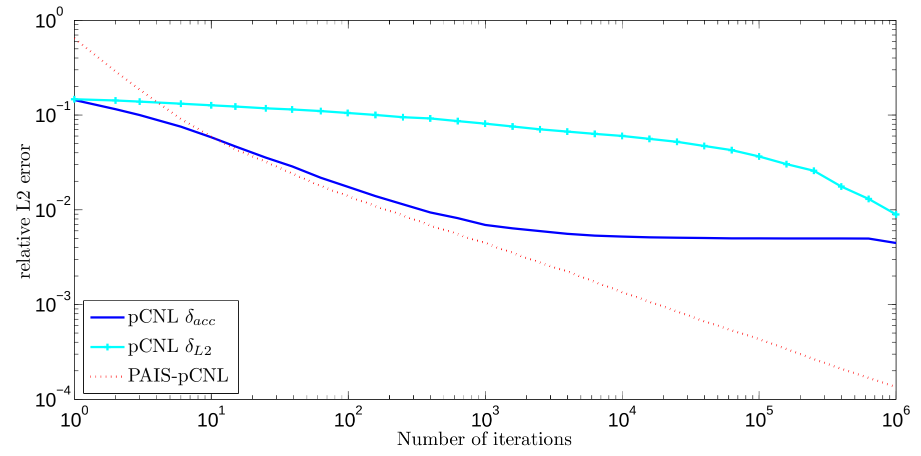

FIG. 5.8. *pCNL and PAIS-pCNL convergence statistics using locally and globally optimal δ. The setup is as described in Section 5.3.2 (2).*

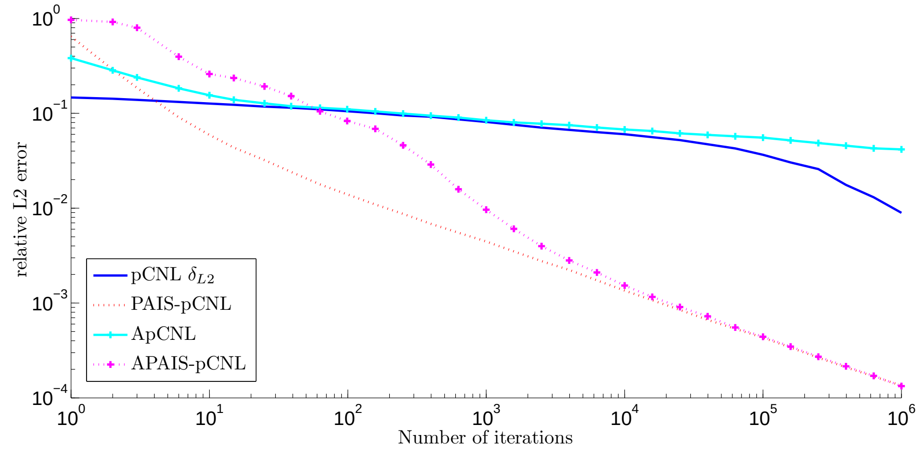

FIG. 5.9. *Convergence graphs for Problem BM(2), the (A)pCNL and (A)PAIS-pCNL algorithms have been run with optimal δ^* for the L2 error and the adaptive algorithm described in Section 4.2. The setup is as described in Section 5.3.2 (3).*

proximation 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 weight. Figure 5.10 shows Problem BM(2) 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.

6. Concluding Remarks. 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 Metropolis-

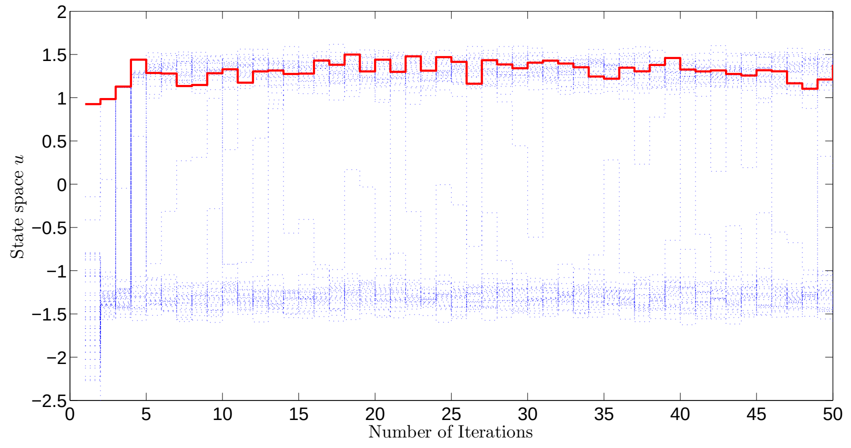

FIG. 5.10. *This figure demonstrates the redistribution property of the PAIS algorithm. Initially there is one chain in the positive mode, and 49 chains in the negative mode.*

Hastings algorithms, further experimentation with the Random Walk Metropolis-Hastings (RWMH), Metropolis Adjusted Langevin Algorithm (MALA) and preconditioned Crank-Nicolson (pCN) proposals has yielded similar results.

One exception where the proposal distribution makes a difference to the convergence of the PAIS algorithm is where the KL divergence is large e.g. the PAIS-pCN algorithm struggles to sample from the distribution in Section 5.2, while the pCN manages to produce a sample, albeit with a slow burn-in period. Additionally the PAIS-RWMH algorithm has no problems sampling from this distribution.

7. More conclusions to add.

Incomplete PAIS (i.e. each processor only needs last reported value of the other chains)

Using less than optimal resampler (lower cost, Schefzik, Roman, Thordis L. Thorarinsdottir, and Tilmann Gneiting. "Uncertainty quantification in complex simulation models using ensemble copula coupling." Statistical Science 28.4 (2013): 616-640.) COLIN TO DO THIS

Use of other MCMC proposals (Riemann, HMC, etc)

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 N-fold increase, where N is the number of cores or processors at our disposal. Our approach demonstrates a better-than-linear speed-up with the number processors/cores used. Thus, our approach is not only embarrasingly parallel (as coined by Cleve Moler in [16]), but humiliatingly so.

REFERENCES

[1] ALEXANDROS BESKOS, FRANK J PINSKI, JESÚS MARIA SANZ-SERNA, AND ANDREW M STUART, *Hybrid monte carlo on hilbert spaces*, Stochastic Processes and their Applications, 121 (2011), pp. 2201–2230.

[2] TAN BUI-THANH AND MARK GIROLAMI, *Solving large-scale pde-constrained bayesian inverse problems with riemann manifold hamiltonian monte carlo*, Inverse Problems, 30 (2014), p. 114014.

[3] Ben Calderhead, *A general construction for parallelizing metropolis- hastings algorithms*, Proceedings of the National Academy of Sciences, 111 (2014), pp. 17408–17413.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] A. Gelman and D. B. Rubin, *Inference from Iterative Simulation Using Multiple Sequences*, Statistical Science, 7 (1992), pp. 457–472.

[9] Mark Girolami and Ben Calderhead, *Riemann manifold langevin and hamiltonian monte carlo methods*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73 (2011), pp. 123–214.

[10] N. Gordon, D. Salmond, and A. Smith, *Novel approach to nonlinear/non-Gaussian Bayesian state estimation*, in IEE Proceedings F (Radar and Signal Processing), vol. 140, IET, 1993, pp. 107–113.

[11] W. Hastings, *Monte Carlo sampling methods using Markov chains and their applications*, Biometrika, 57 (1970), pp. 97–109.

[12] 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.

[13] R. Kalman, *A new approach to linear filtering and prediction problems*, Journal of Fluids Engineering, 82 (1960), pp. 35–45.

[14] S. Lan, J. Streets, and B. Shahbaba, *Wormhole Hamiltonian Monte Carlo*, ArXiv e-prints, (2013).

[15] Jun S Liu, Faming Liang, and Wing Hung Wong, *The multiple-try method and local optimization in metropolis sampling*, Journal of the American Statistical Association, 95 (2000), pp. 121–134.

[16] Cleve Moler, *Matrix computation on distributed memory multiprocessors*, Hypercube Multiprocessors, 86 (1986), pp. 181–195.

[17] Gordon E Moore et al., *Cramming more components onto integrated circuits*, Proceedings of the IEEE, 86 (1998), pp. 82–85.

[18] Radford M Neal, *Mcmc using ensembles of states for problems with fast and slow variables such as gaussian process regression*, arXiv preprint arXiv:1101.0387, (2011).

[19] E. Nummelin, *General Irreducible Markov Chains and Non-Negative Operators*, Cambridge University Press, 1984. Cambridge Books Online.

[20] S. Reich, *A nonparametric ensemble transform method for Bayesian inference*, SIAM Journal on Scientific Computing, 35 (2013), pp. A2013–A2024.

[21] G. Roberts and J. Rosenthal, *Optimal scaling for various Metropolis-Hastings algorithms*, Statistical science, 16 (2001), pp. 351–367.

[22] ———, *Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms*, Journal of Applied Probability, 44 (2007), pp. 458–475.

[23] ———, *Examples of adaptive MCMC*, Journal of Computational and Graphical Statistics, 18 (2009), pp. 349–367.

[24] JC Sexton and DH Weingarten, *Hamiltonian evolution for the hybrid monte carlo algorithm*, Nuclear Physics B, 380 (1992), pp. 665–677.

[25] B. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman & Hall/CRC Monographs on Statistics & Applied Probability, Taylor & Francis, 1986.

[26] C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson, *Obstacles to High-Dimensional Particle Filtering*, Monthly Weather Review, 136 (2008), pp. 4629–4640.

[27] A. Stuart, *Inverse problems: a Bayesian perspective*, Acta Numerica, 19 (2010), pp. 451–559.

[28] C. Villani, *Topics in Optimal Transportation*, Graduate studies in mathematics, American Mathematical Society, 2003.

[29] ———, *Optimal Transport: Old and New*, Grundlehren der mathematischen Wissenschaften, Springer Berlin Heidelberg, 2008.