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 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~~ensemble members, for low dimensional problems, and in fact shows better than linear improvements in convergence rates with respect to the number of ~~processors used~~ 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.

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 [13], and with the development of ~~and with the development of faster~~faster, more powerful computers, have become ever more important in a whole range of fields in statistics, science and engineering. 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, the application of MCMC methods to 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 that propose moves more intelligently, leading to rapidly converging ~~methods~~approximations. 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 [7]. 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 [30], and function space analogues of this have also been proposed [1]. 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 [11]. 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 explore the posterior distribution more fully with fewer iterations.

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

*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

followed since the early 70s [19]. In recent times, it has become necessary to use parallel architectures ~~to exploit Moore's law~~in order for this trend to continue. The efficient exploitation of these ~~facilities~~ architectures 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~~ naively 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 [17] and ensemble MCMC [20]. 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 can be built around ~~any~~ many 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. The idea is to consider the current state on each of a set of parallel chains as an ensemble, and to resample using a transformation based on optimal transport. Samplers based on optimal transport have also been considered in [8].

In Section 2 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~~ introduce some mathematical preliminaries upon which we will ~~be employing within the algorithm. We describe the Parallel Adaptive Importance Sampler (PAIS) in Section 3, and in Section 4 we describe how algorithmic parameters can be automatically tuned to provide optimal convergence~~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 present some numerical ~~experiments that demonstrate the savings available by employing this approach as opposed to a naive/trivial parallelisation of existing MCMC methods. In Section 7, we summarise our results and suggest some areas for future investigation~~examples, before a brief conclusion and discussion in Section 7.

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 ~~in~~ arising from 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~~ x is a member of a Hilbert space X.

The parameter ~~u~~ x is observed through the observation operator $\mathcal{G} : X \to \mathbb{R}^d$. Since observations are never perfect, we assume that these measurements D are subject to Gaussian noise, so that

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

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

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

where ~~$\|x - y\|_\Sigma$~~ $\|y_1 - y_2\|_\Sigma$ is the Mahalanobis distance between ~~x and y~~y_1 and $y_2 \in \mathbb{R}^d$.
As discussed in [6,31], 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:

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

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 [7]. These new discretisations perform comparably with the original versions in low dimensions. 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 [7], a function space version of this method was presented, the pCNL algorithm, and is described in full in Table ??. 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.~~
~~$X = x_0$ $Y = (2+\delta)^{-1}\left[(2-\delta)X_{i-1} - 2\delta\mathcal{C}\nabla\Phi(u) + \sqrt{8\delta}W\right]$, $W \sim \mu_0$ $a(X_{i-1}, Y) = \min\{1, \exp(\Phi(X_{i-1}) - \Phi(Y))\}$.~~
~~$u \sim U([0,1])$ $X_i = Y$ $X_i = X_{i-1}$ A pseudo-code representation of the preconditioned Crank-Nicolson Langevin (pCNL) algorithm. $\delta \in (0, 2]$ is a step size parameter.~~

2.2. 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 [12]. 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 [15] and ensemble Kalman filter [9], 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~~ One downside is that ~~,~~ the required ensemble size increases quickly with dimension, making it difficult to use in high-dimensional problems. Another downside is that the effective sample size

decreases at each iteration, resulting in degeneration of the approximation of the posterior. One way to tackle this is to employ a resampling scheme. The aim of a successful resample is to take ~~your~~ the 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~~ ensemble transform particle filter (ETPF) proposed by Reich [24] makes use of optimal transportation as described in [32, 33]. 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 [5]. This coupling is the solution to a linear programming problem in M^2 variables with $2M-1$ constraints. 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.

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.~~ .

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

However, we argue that this is ~~a far from~~ not an optimal scenario. First of all, unless we have a lot of information about the posterior, we will ~~begin the algorithma long way from statistical equilibrium. Some initial iterations then are not~~ 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 ~~posterior~~target distribution, and must be thrown away. This process is known as the burn-in. In a ~~trivially~~ naively parallelised scenario, each ~~process~~ 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 ~~time~~ 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 ~~trivially~~ naively parallelised MCMC scheme is exactly that - ~~trivial~~naive. Intuition suggests that we can gain some speed up by sharing information across the ~~processes and that is exactly~~ ensemble members, and this is what we wish to demonstrate in this paper.
These deficiencies of the ~~trivial~~ 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. ~~We will then introduce the version that we have implemented, which utilises the resampler recently suggested by Reich [24] described in Section 2.2, and the pCNL proposal distribution described in Section ??.~~

3. The Parallel Adaptive Importance Sampler (PAIS).

~~Important~~ 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 ~~the states of a set of parallel MCMC chains~~ an ensemble of states, each coming from a MCMC chain, to construct a proposal distribution which will be as close as possible to the target density. ~~Given enoughprocessors, the states of all of these chains at one point in time may be reasonably representative sample~~ If this ensemble is large enough, the distribution of states will be 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 ~~MCMC proposals (in this paper, specifically the pCNL proposal from Section ??)~~a sum of MCMC proposal distributions for each of the members of the ensemble. Once the proposal is constructed, we can sample a new set of states from the proposal distribution, and each is assigned a weight given by the ratio of the target density and the proposal mixture distribution 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 ~~the ETMC algorithm. Once we once again have~~ a resampling algorithm. Initially we will use the ETPF algorithm [24], although we will suggest an alternative strategy in Section 5. The output of the resampling algorithm gives us a set of evenly weighted samples that we believe represents the target distribution well, and from this point we can iterate the process once again. The algorithm is ~~given in more detail in~~ summarised in Table 3.1.
We wish to sample states $x \in X$ from a posterior probability distribution ~~μ_Y~~ μ_D, where D represents our data which we wish to use to infer x. Since we have M ~~processes~~ensemble members, 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 might come from an MCMC method, for example the ~~pCNL proposal presented in Section ??~~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 $(\mathbf{W}^{(i)}, \mathbf{Y}^{(i)})_{i=1}^N$ ~~, or their statistics, are stored~~are the samples from the posterior that we will analyse.
The key ~~here~~ is to choose a suitable transition kernel ν such that if $X^{(i)}$ is a ~~decent~~

```
$\mathbf{X}^{(0)} = \mathbf{X}_0 = [x_1^{(0)}, x_2^{(0)}, \dots, x_M^{(0)}]^T$
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~~using ETMC~~: $(\mathbf{W}^{(i)}, \mathbf{Y}^{(i)}) \to (\frac{1}{M}\mathbf{1}, \mathbf{X}^{(i+1)})$
end for
```

Table 3.1

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

good representative sample of the posterior, then the mixture density $\chi(\cdot; \mathbf{X}^{(i)})$ is a ~~reasonable~~ good 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 6, we will demonstrate how the algorithm performs, ~~using pCNL algorithm~~primarily using RWMH proposals. We do not claim that this choice is optimal, but is simply chosen as an example to show that sharing information across ~~processes~~ ensemble members 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. 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 simple examples [29].

Care needs to be taken when choosing the proposal distribution to ensure that the proposals are absolutely continuous with respect to the posterior distribution. In Section 6.3 we will consider an inverse problem with Gamma priors. Since the posterior distribution in this context is heavy-tailed, if we use a lighter tailed distribution for ν, such as a Gaussian kernel, then importance weights in the tails will be unbounded, which will hamper convergence of the algorithm.

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~~ 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. If the proposal distribution is too over-dispersed, then the algorithm will ~~propose states a long way from the current state at the cost of a relatively low acceptance rate, impacting the quality of inferences on the posterior~~often propose states in the tails of the distribution, resulting in larger variance of the weights, and therefore slower convergence to the posterior distribution. Similarly, if the ~~posterior~~ proposal distribution is under-dispersed, the ~~process will~~ proposals will be highly correlated with the previous states, and the algorithm will take a long time to fully explore the ~~space so the mixing rate, and hence convergence rate, will be slow~~parameter space, and worse, will lead regularly to states proposed in the tails of the proposal distribution with very large weights. It is therefore necessary to find a proposal distribution which is slightly over-dispersed to ensure the entire posterior is explored [10], but is as close to the posterior as possible.

~~Proposal distributions which~~ Most commonly used MCMC proposals have parametric dependence which allows the user to control their variance. For example, in the RWMH proposal $y = x + \beta\eta$, the parameter β controls how correlated the proposal state y is to the current state x. Therefore the proposal distributions can be tuned such that they are are slightly over-dispersed, as described above~~, can be found by tuning the variance of the proposal distribution ν~~. 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~~. 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 [26, 27]:~~

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.~~
2. ~~The smallest number of steps for which the kernel has sufficiently converged, from an initial state x, is finite, by bounded convergence.~~

, and have been shown to be convergent provided that they satisfy certain conditions [26, 27].

~~The~~ Algorithms which use mixture proposals, e.g. PAIS, must tune the variance of the ~~proposal distributions that are used~~ individual kernels within the proposal mixture~~distribution plays a key role in how well the proposal distribution represents the target distribution, and therefore how quickly the algorithmconverges. A large over-dispersed proposal distribution will sample a lot from the tails of the distribution, but may also find other regions of high density with respect to the target distribution. A very small variance in the proposal distributions will lead to a very rough proposal mixture distribution. In practise, we wish to find the happy medium in which the proposal distribution is slightly more dispersed than the target density, whilst also being a good representation of the regions which have high density with respect to the target distribution.~~

~~Adaptively choosing the variance of the proposal distributions with a large initial guess allows us to first explore the state space globally, searching for multiple modes. Reducing the proposal~~. This adaptivity during the burn-in has some added benefits over and above finding an optimal parameter regime for the algorithm. If the initial value of the proposal variances is chosen to be very large, then proposed moves will be made far and wide, expediting the early mode-finding stages of the algorithm. Adaptively reducing the proposal variances to an optimal value then allows us to explore each region efficiently. The fact that we have ~~multiple~~ an ensemble of chains allows us to assess quickly and effectively what the optimal variance of the proposal distributions should be.

The alternative to using adaptive procedures to tune the scaling parameters is to perform exploratory simulations to find the optimal regimes by trial and error. This can be very costly and the optimal parameters cannot realistically be found to more than a couple of significant figures. It is therefore important that MCMC algorithms provide a feasible adaptive strategy.

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~~ the optimal rates are 23.4% for RWMH and 57.4% for MALA [25]. 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~~β. Section 4.1 gives some possible methods ~~of tuning δ~~for tuning β.

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, in cases where the posterior distribution can be computed using alternative methods. To assess this, a distance metric on distributions must be chosen. Examples are the relative error between the sample moments and the ~~true~~ posterior's moments, or the relative ~~L2~~ L^2 error between the true density, $\pi(x|D)$, and the constructed histogram. The relative error in the m-th moment is

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

given by:

$$\left|\frac{N^{-1}\sum_{i=1}^{N}x_i^m - \mathbb{E}[X^m]}{\mathbb{E}[X^m]}\right|, \tag{4.1}$$

where $\{x_i\}_{i=1}^N$ is a sample of size N~~produced by the algorithm~~. The relative ~~L2~~ L^2 error between a continuous function to a piecewise constant function, e, ~~is~~ can be given by considering the difference in mass between the normalised histogram of the samples and the posterior distribution over a set of disjoint sets or "bins":

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

where the ~~points~~ 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 ~~a~~ 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 ~~are not practical for finding~~ cannot be used in general to find optimal values of ~~δ~~ β since they require knowledge of the analytic solution, and ~~require that the algorithm~~ the algorithm must be run for a long time to build up a sufficiently large sample. However they can be used to assess the ability of other ~~statistical measures~~ indicators to find the optimal proposal variances in a controlled setting.
~~Another related statistic, is the variance of the sample 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 specifically for importance sampling algorithms.

4.1.2. ~~Determining optimal δ using the~~ 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~~ Importance samplers are most efficient when the ~~posterior~~ target is proportional to the proposal distribution~~, in~~. In this case the weights are ~~constant~~all equal, and so the variance of the weights, $var(w(y))$, is zero. Hence, ~~the optimal value of δ, is~~ we would like to choose the value of β which minimises the variance of the weights,

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

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

4.1.3. ~~Determining optimal δ using the~~ The effective sample size. The effective sample size, n_{eff}, can also 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.~~ cannot achieve a perfect effective sample size of M.
The effective sample size can be 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 we see that when the variance of the weights is zero, $n_{\text{eff}} = M$; this is our ideal scenario. ~~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~~ 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.~~
The statistic n_{eff} ~~converges faster~~ is easier to deal with than the variance of the weights, ~~and so~~ as it varies between 1 and M as opposed to the variance which can vary over many orders of magnitude. Therefore it is preferable as a means of tuning ~~δ. While δ^*_{var} and the optimal value found using~~ the scaling parameter. In all of the numerics which follow, we use 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^*~~ calculated at each iteration and averaged across the entire run to tune the scaling parameters. We do this because when we tune this parameter on the fly, we use the single-iteration average to estimate optimality. The optimal scaling parameter found using the global optimum of the variance of the weights is also included for comparison. Note that for the majority of iterations the value of n_{eff} is an overestimate of this statistic over a larger number of samples. Rare events which result in a large importance weight bring this statistic down, and care must be made not to overfit the proposal variance to n_{eff} on an iteration by iteration basis. This can be accounted for by increasing the variance slightly once the adaptive algorithm has arrived on a value using single iteration values of n_{eff}.
The ~~n_{eff} statistic~~ effective sample size also has another useful property; if we imagine the algorithm in the burn-in phase, for example, we have M ~~processes~~ ensemble members in the tail of a Gaussian curve searching for the area of high density. If

the ~~processes~~ ensemble members are evenly spaced, then the particle closest to the mean will have an exponentially higher weight assigned to it. The effective sample size ratio 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.~~ ensemble populates the regions where the majority of the probability density lies, and the proposal distributions better represent the posterior distribution. This leads to smaller variance in the weights, and a bigger effective sample size. By this argument we can see that rising n_{eff} signals the end of the burn-in period.

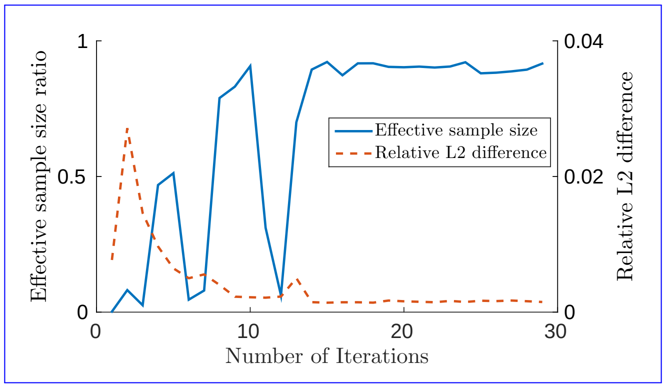
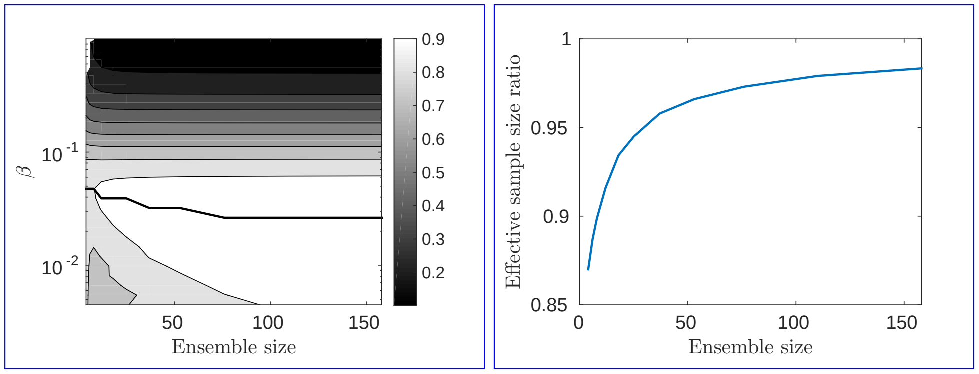


FIG. 4.1. *The effective sample size ratio and relative L^2 difference between the proposal and posterior distributions at each of the first 30 iterations. These numerics are taken from a simulation of the problem in Section 6.3 using the PAIS-Gamma algorithm.*

4.1.4. ~~Behaviour of the effective sample size for varying ensemble size.~~ Figure 4.1 shows that the effective sample size flattens out at the same time as the relative L^2 difference between the posterior distribution and the proposal distribution stabilises close to its minimum. Detecting this stationarity allows us to automatically determine the end of the burn-in phase of the algorithm.

If we are to use the effective sample size ratio as an indicator of how to tune the proposal variance, we need to look at how it behaves in different situations. ~~In Section 6 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 4.2 shows results for the ~~second problem discussed in ??~~ Gaussian posterior discussed in Section 6.1 when using the ~~PAIS-pCNL algorithm to explore the posterior distribution. The left part of the figure~~ PAIS algorithm with RW proposals, each with variance $\beta^2 \in \mathbb{R}_{>0}$. Figure 4.2 (a) shows that as the ensemble size increases, the ~~value of δ~~ scaling parameter which gives the optimal effective sample size ratio decreases. This is to be expected since if we are trying to ~~fit more kernels into the posterior distribution, so the variance of each kernel needs to be decreased.~~ approximate the posterior with a mixture distribution of a small number of Gaussians, the optimal variance will naturally be larger so that the whole of the significant regions are covered by the proposal distribution. As the number increases, the optimal variances decrease so that finer details in the posterior can be better represented in the proposal. Figure 4.2 (b) shows that as the ensemble size increases, the efficiency of the sampler also increases.

4.2. ~~The~~ Adaptive ~~Algorithm~~PAIS. A popular approach for adaptive MCMC algorithms is to view the scaling parameter as a random variable which we can sample



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

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

FIG. 4.2. *~~Left: A contour plot showing how δ_{eff} varies with the ensemble size.~~ The ~~black line highlights the optimal value~~ behaviour of ~~δ. Right: The value of~~ the effective sample size ~~ratio for~~ as the ~~optimal δ at each~~ ensemble size increases. ~~i.e.~~ This analysis is from the ~~value of the effective sample size ratio along the black line~~ Gaussian posterior in ~~the left figure. Set up for this problem is given in~~ Section ~~??~~6.1 using the PAIS-RW algorithm.*

during the course of the MCMC iterations. However, it can be slow to converge to the optimal value, and we may need an uninformative prior for the scaling parameter. 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~~ evolution of the chain. This results in some iterations which make larger global moves in the state space ~~but never converges~~ between modes in the target distribution, and some which make local moves. Algorithms of this type do not converge to an optimal value of the scaling parameter. We choose to use a divide and conquer scheme which optimises the effective sample size (or any other diagnostic). Some more sophisticated examples are described in [27] and [14]. ~~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.~~

From ~~hereon~~ here on in, we will ~~assume that we are using the proposal from the pCNL algorithm given in Section ??, which is given by:~~ use the random walk proposal,

$$\underline{Y}y = \underline{(2+\delta)^{-1}(2-\delta)X}x_{i-1}\underline{-2\delta\mathcal{C}\nabla\Phi(u)}+\underline{\sqrt{8\delta}W}\beta\omega_{i-1}, \qquad \underline{W}\omega_{i-1} \sim \underline{\mu_0}\mathcal{N}(0,\Sigma) \qquad \text{i.i.d.} \tag{4.3}$$

where ~~X~~ x_{i-1} is our current state~~and Y~~, y is our proposed state ~~. The scaling parameter δ dictates the variance in the proposal distribution.~~
~~In the adaptive algorithm described in Table ?? in Appendix ??~~and Σ is a covariance operator, which could come from the prior distribution. In the numerics section we will refer to β as the scaling parameter.
Using an adaptive strategy, we calculate a sequence ~~$\{\delta^{(k)}\}_{k=1}$ which converges~~ $\{\beta^{(k)}\}_{k=1}$ which converges roughly to the optimal scaling parameter ~~δ^*~~β^*, resulting in the optimal transition density χ for our MCMC algorithm. This optimal value will differ depending on the criterion we are optimising. We must choose some sequence of iterations, $\{n_k\}_{k=1}$, 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.~~ β, and due to the constraints on adaptive MCMC algorithms [26,27], these n_k must grow exponentially

further apart. This same adaptive approach can also be applied to the trivially parallelised MCMC algorithms to adaptively calculate their optimal scaling parameter β^*.

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 ~~given for the PAIS method but a similar method is used to adaptively calculate δ^* for the pCNL algorithm~~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).

~~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. However, if it is chosen to be too large, then the method may become unstable and inefficient. We also choose two iteration numbers at which we change our adaptive strategy. At iteration N_{join} we resample using the full ensemble size instead of using to subsets of the full ensemble~~ 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:

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

with associated states y_n. We wish to find M vectors $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_M\} \subset \mathbb{R}^M_{\geq 0}$ 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 $\mathbf{p}_i = [p_{i,1}, p_{i,2}, \dots, p_{i,M}]$ 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:

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

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.

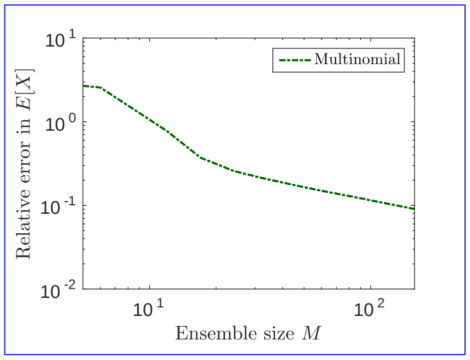
There are of course infinitely many different ways that one could use to split the original multinomial up into M parts, some of which will be far from optimal. The method that we have chosen is loosely based on the idea of optimal transport. We search out states with the largest weights, and choose 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.

```
$\mathbf{z} = M\bar{\mathbf{w}}$
for $i = 1, 2, \dots, M$ 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
        $K = \arg\min_{k \in \{k | z_k > 0\}} \|y_J - y_k\|$
        $p_{i,K} = \min\{1 - \sum_j p_{i,j}, z_K\}$
        $z_K = z_K - p_{i,K}$
    end while
    $x_i = \sum_k p_{i,k} y_k$. ~~This allows us to fine-tune the scaling parameter using the full resampler, but slows down the rate of convergence of the scaling parameter. Finally we choose a time, $N_{\text{stop}}$, at which we stop updating our parameter to guarantee ergodicity of~~
end for
```

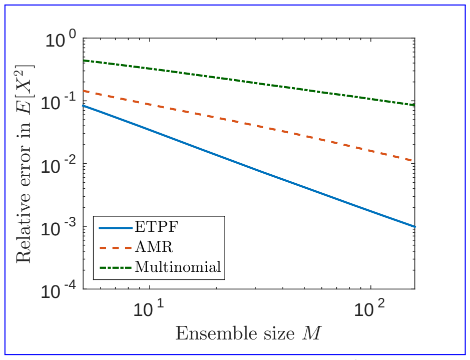
TABLE 5.1
The approximate multinomial resampler (AMR) algorithm.

Table 5.1 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.

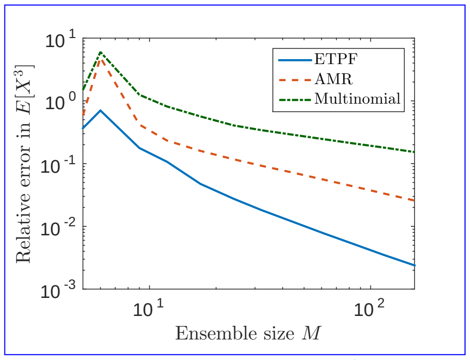
To test the accuracy and speed of the three resamplers (ETPF, bootstrap and AMR), we drew a sample of size M from the proposal distribution $\mathcal{N}(1, 2)$. Importance weights were assigned, based on a target distribution of $\mathcal{N}(2, 3)$. The statistics of the resampled outputs were compared 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



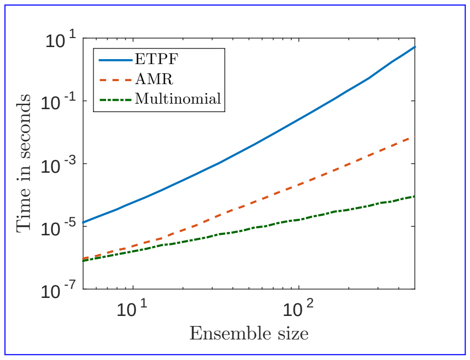
(a) Relative error in $\mathbb{E}(X)$



(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. *Finding optimal values of β for the problem in Section 6.1. The setup is as in Section 6.1.2.*

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 ~~MCMC~~ total.
We will use the AMR in the numerics in Section 6.3.3 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. Numerical Examples.

6.1. 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 [7]~~ compare the naively parallelised RWMH algorithm with its PAIS variant, the ~~PAIS-pCNL algorithm. We consider several statistics for measuring the efficiency~~ PAIS-RW algorithm. The PAIS algorithm is implemented using the ETPF to perform the resampling step. We assess the performance of the PAIS algorithm ~~compared to existing Metropolis-Hastings algorithms when applied to a simple one dimensional Gaussian posterior~~using the relative L^2 error defined in (4.2), as well as the relative error in the first moment (4.1).
Since we are comparing against ~~trivially parallelised MCMC~~ naively parallelised MH algorithms, we also need to decide which ~~statistic $T(\delta)$ to optimise for these approaches~~statistics

$T(\beta)$ provide the best criterions for obtaining the optimal scaling parameters. In the examples which follow, we have optimised the naively parallelised ~~pCNL~~ RWMH algorithm using the optimal acceptance rate ~~$\hat{\alpha} = 0.75$~~$\hat{\alpha} = 0.5$. This value ~~was found by using the L2 error minimums to calculate δ^*. This means that we are minimising the statistic~~ differs from the theoretical asymptotic value of 0.234 which applies in higher dimensions, but this higher acceptance rate is commonly used for one dimensional Gaussian posteriors [28]. To find the optimal scaling parameter we minimise the statistic

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

where ~~$N_{\mathrm{acc}}(\delta)$~~ $N_{\mathrm{acc}}(\beta)$ is the number of accepted moves and N_{total} is the total number of samples produced. For the PAIS algorithm, we ~~are maximising~~ maximise the effective sample size as discussed in Section 4.1.3~~, so $T_{\mathrm{PAIS}}(\delta) = n_{\mathrm{eff}}(\delta)$~~.

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

$$\mathrm{law}(\mu_{Y\,D}) = \pi(u x|D) \propto \exp\left(-\frac{1}{2}\left\|u x - D\right\|^2_{\Sigma\,\sigma^2} - \frac{1}{2}\left\|u x\right\|^2_{\mathcal{T}\,\tau^2}\right), \tag{6.1}$$

where ~~Σ and $\mathcal{T}$ are the covariances~~ σ^2 and τ^2 are the variances 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 $\sigma^2 = 0.1$~~$\tau^2 = 0.01$ and $\sigma^2 = 0.01$, and we observe ~~$u = -2.5$~~ $x_{\mathrm{ref}} = 4$ noisily such that

$$D = \mathcal{G}(u x) + \eta \sim \mathcal{N}(\mathcal{G}(u x_{\mathrm{ref}}), \Sigma\sigma^2) = \mathcal{N}(4, 0.01).$$

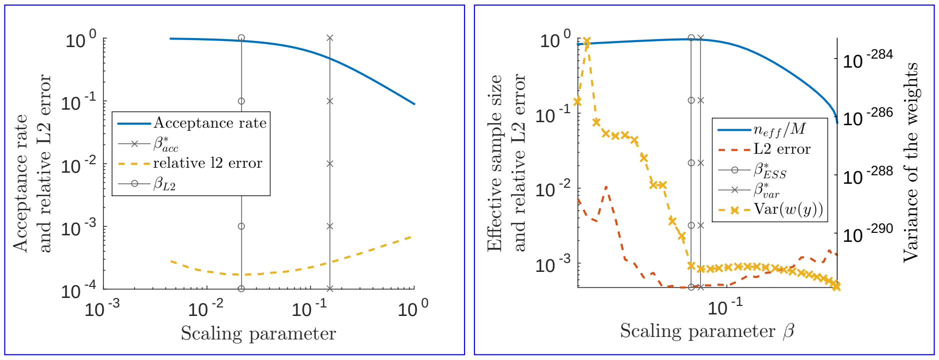
These values result in a posterior density in which the vast majority of the density is ~~contained inside the high density region~~ out in the tails of the prior ~~. This means that it should be straightforward for the algorithm to find the stationary~~ distribution. The Kullback-Leibler (KL) divergence, which gives us a measure of how different the prior and posterior are, is ~~$D_{KL}(\mu_Y\|\mu_0) = 2.67$~~ $D_{KL}(\mu_D\|\mu_0) = 4.67$ for this problem. A KL divergence of zero indicates that two distributions are identical almost everywhere.

6.1.2. Numerical implementation. In each of the following simulations, we perform three tasks. First we calculate the optimal value of ~~δ~~ β by optimising the statistics described in Section 4.1. ~~Once we have these parameters, we~~ We then run the algorithms with optimal parameters to calculate and compare the convergence ~~speeds of the algorithms~~rates. Finally, we implement the adaptive algorithms described in Section 4.2 and compare the convergence rates of these algorithms with the nonadaptive algorithms.
(1) Finding the optimal parameters: To find the optimal parameters we choose 32 values of ~~δ~~ β evenly spaced on a log scale ~~between~~ in the interval $[10^{-5}, 2]$. We run the ~~PAIS-pCNL algorithm for 10,000 iterations and pCNL for 100,000~~ PAIS-RW and RWMH algorithms for one million iterations, each with an ensemble size of $M =$

50~~processes~~. We took 32 repeats of both algorithms and then used the ~~medians of the~~ geometric means of the sample statistics to find the optimal parameters.

(2) Measuring convergence of nonadaptive algorithms: We run the algorithms in Section 6.1 ~~and Section ?? for 10~~ for one million iterations, again with ~~50 processes. The algorithms are run~~ $M = 50$. The simulations are repeated 32 times using the optimal parameters found in (1). The ~~relative L2 error (Equation 4.2) is used as a measure of accuracy. The simulation for each algorithm was repeated 24 times.~~ performance of the algorithms is judged by the convergence of the relative L^2 error statistic in (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 ~~L2~~ L^2 error to compare efficiency. The ~~initial value of δ is given in the discussion of each simulation.~~ adaptive algorithms are initialised with $\beta^{(1)} = 1$.



(a) RWMH algorithm. (b) PAIS-RW algorithm.

Fig. 6.1. *Finding optimal values of ~~δ~~ β for the ~~pCNL (left) algorithm and PAIS-pCNL (right) algorithm for the~~ problem in Section 6.1. The setup is as in Section 6.1.2. Resampling is performed using the ETPF.*

Statistic	~~pCNL~~ RWMH
~~δ^*_{L2}~~ β^*_{L2}	~~3.7e-3~~ 2.1e-2
~~$\delta^*_{\text{var}(\hat{\mu})}$~~ $\beta^*_{\%}$	~~5.8e-2~~ 1.5e-1
Acceptance Rate (~~δ^*_{L2}~~ β^*_{L2})	~~9.9~~9.0e-1
Acceptance Rate (~~$\delta^*_{\text{var}(\hat{\mu})}$~~ $\beta^*_{\%}$)	~~7.4~~5.0e-1

Statistic	~~PAIS-pCNL~~ PAIS-RW
~~δ^*_{eff}~~ β^*_{eff}	~~1.5~~4.7e-2
~~$\delta^*_{\text{var}(w(y))}$~~ $\beta^*_{\text{var}(w(y))}$	~~6.4~~5.8e-2
~~δ^*_{L2}~~ β^*_{L2}	~~1.7~~3.9e-2

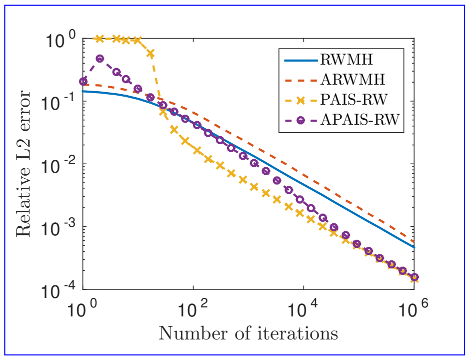
Table 6.1

Optimal values of ~~δ~~ β summarised from Figure ~~??~~6.1. 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 $\beta^*_{\text{var}(w(y))}$ optimise the effective sample size and variance of the weights statistics.

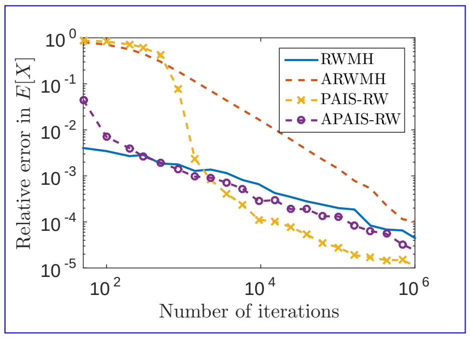
6.1.3. Optimal values of ~~δ~~β. Figure ~~?? (left~~6.1 (a) shows the two values of ~~δ which may be optimal for the pCNL algorithm, found at the turning points. The first~~ β which are optimal according to the acceptance rate and relative L^2 error criteria for the RWMH algorithm. The smaller estimate comes from the relative ~~L2~~ L^2 error, and the ~~second comes from the variance of the estimate of the mean. The optimal acceptance rates of the function space algorithms are expected to be slightly higher~~

~~than their finite dimensional versions. Since the L2 errorestimate of the histogram gives an acceptance ratewhich is near to 100%, we say that the optimal acceptance rate is that as given by the variance of the mean, roughly 75%. The results in Figure ?? are summarised in Table ?? (left).~~ larger from the acceptance rate. The results in Figure 6.1 are summarised in Table 6.1. 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 ~~?? (right~~6.1 (b) shows the effective sample size ratio compared to the error analysis and the variance of the weights. ~~Although the L2~~ The relative L^2 error graph is noisy, but it is clear that the maximum in the effective sample size ~~is~~ and the minimum in the variance of the weights are both close to the minimum in the ~~L2 error. The minimum in the variance of the weights however is a long way away. We choose the~~ relative L^2 error. Due to this we say that the estimate of the effective sample size ~~as the best estimator of~~ found by averaging the statistic over each iteration is a good indicator for the optimal scaling parameter~~because of this , and because the effective sample size statistic converges to a smooth graph faster than either the variance of the weights or the L2 error~~. In general this indicator overestimates the value of n_{eff} found by using the entire sample.



(a) Relative L^2 error.



(b) Relative error in first moment.

Fig. 6.2. *~~Relative error in~~ Error analysis for the ~~first moment (right) and histograms (left) produced by the~~ (A)~~pCNL~~ RWMH and (A)~~PAIS-pCNL~~ PAIS-RW algorithms~~against iterations for problem 6.1~~. The setup is as in Section 6.1.2 (2, 3). Resampling is performed using the ETPF.*

6.1.4. Convergence of ~~pCNL~~ RWMH vs ~~PAIS-pCNL~~PAIS-RW. Figure 6.2 shows that the ~~PAIS-pCNL~~ PAIS-RW algorithm converges to the posterior distribution significantly faster than the ~~standard function space pCNL~~ RWMH algorithm, in both ~~L2~~ L^2 error and relative error in the moments. A description of the speed up attained by this algorithm is given in Section 6.2.4.

Both adaptive algorithms are run with initial values of ~~$\delta_0 = 0.1$. From~~ $\beta = 1$. Figure 6.2 ~~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~~ shows that after an initial burn-in ~~phase, some chains find themselves a long way out in the tails where~~ period the APAIS-RW algorithm catches up to the PAIS-RW algorithm, and by the end of the simulation window is matching its performance. The ARWMH algorithm does not perform quite as well, this is possibly due to the ~~high gradient they will overshoot the high density region and reject almost all proposed values~~fact that the acceptance rate cost function is not particularly smooth at the optimal value making it difficult to minimise.

6.1.5. Scaling of the PAIS algorithm with ensemble size. Throughout this ~~paper~~example, we use an ensemble size $M = 50$, but it is interesting to see how the PAIS algorithm scales ~~if we were to~~ when we increase the ensemble size, and if there is some limit below which the algorithm fails. We implement the problem in Section 6.1, using the ~~pCNL algorithm~~ RWMH and PAIS-RW algorithms with ensemble sizes ~~ranging from $M = 2$ up to $M = 494$~~in the interval $M \in [1, 160]$.

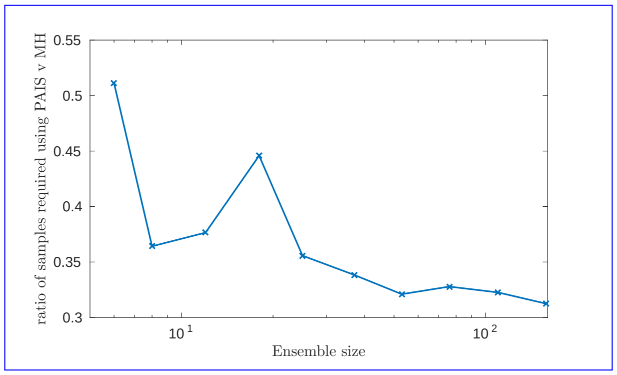


Fig. 6.3. *Ratio of ~~PAIS-pCNL~~ PAIS-RW samples required to reach the same tolerance as the ~~pCNL~~ RWMH algorithm.*

Figure 6.3 was produced using the method of finding optimal ~~δ~~ β described in Section ~~6.1.2(1)~~6.1.2 (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 ensemble size continues to reduce the number of iterations required in comparison with ~~naive MCMC. When $M < 8$, the algorithm takes a long time to reach stationarity. This~~ naively parallelised MH. The decreasing trend indicates 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.

~~6.2. Sampling from a one dimensional Gaussian with a higher KL divergence.~~ ~~In this example we use the same setup used in Section 6.1. 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.~~

~~6.1.1. Target distribution.~~ ~~As in the previous example, we use the identity observation operator $\mathcal{G}(u) = u$, which results in the Gaussian posteriors in Equation 6.1. 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 observational noise drawn from $\mathcal{N}(0, \sigma^2)$.~~

6.1.1. ~~Optimal values of δ~~. ~~Finding optimal values of δ for the pCNL (left) algorithm and PAIS-pCNL (right) algorithm for the problem in Section **??**. The setup is as in Section 6.1.2.~~

~~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 Optimal values of δ summarised from Figure **??**. Statistics calculated as described in Section 4.1.~~

~~We find the optimal values of δ using the same methods described in Sections 6.1.2 and 6.1.3. The results are displayed in Figure **??** and Table **??**. 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~~ This decrease is linked to the increasing 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 **??**, 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. The variance of the weights becomes so small in the critical region that we get numerical zeroes, which means that we could not use this method to tune the scaling parameter adaptively.~~ shown in Figure 4.2 (b).

6.1.1. ~~Convergence of pCNL vs PAIS-pCNL~~. ~~Relative error in the first moment (right) and histograms (left) produced by the (A)pCNL and (A)PAIS-pCNL algorithms against iterations for problem **??**. The setup is as described in Section 6.1.2 (2,3).~~

~~Figure **??** shows 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.~~

6.2. Sampling from Bimodal Distributions. In this section we investigate the behaviour of the PAIS algorithm when applied to bimodal problems. ~~Metropolis-Hastings~~ MH methods can struggle with multimodal problems, particularly where switches between the modes are rare, resulting in incorrectly proportioned modes in the histograms~~, for example. With~~ . This example demonstrates that the PAIS algorithm ~~, we see that the resampling step~~ redistributes chains to new modes as they are found. This means that we expect the number of chains in a mode to be approximately proportional to the probability ~~mass~~ density in that mode. As a result, reconstructed posteriors with disproportional modes, as is familiar with the ~~Metropolis-Hastings~~ MH algorithms, are not produced. ~~We again~~

6.2.1. Target Distribution. We look at an ‘easy’ problem, ~~BM(1)~~ B_1, which has a KL divergence of 0.880, and a ‘harder’ problem, ~~BM(2)~~ B_2, which has a KL divergence of 3.647. Problem ~~BM(1)~~ B_1 has two modes which are ~~separated by a smaller energy barrier. In BM(2)~~ not too far apart. In B_2 we increase the distance between the two modes which has the effect of increasing the ~~required energy~~ expected number of iterations that it takes for a MCMC chain to jump between modes. These posteriors are shown in Figure 6.4.

6.2.2. ~~Target Distribution~~. The following setup is the same for both problems. We consider ~~an observation operator $\mathcal{G}(u) = u^2$~~ a non-linear observation operator $\mathcal{G}(x) = x^2$, and assign the prior ~~$u \sim \mu_0 = \mathcal{N}(0, \tau^2 = 0.25)$~~ $x \sim \mu_0 = \mathcal{N}(0, \tau^2 = 0.25)$.

We assume that a noisy reading, D, is taken according to ~~$D = \mathcal{G}(u) + \varepsilon$~~$D = \mathcal{G}(x_{\text{ref}}) + \varepsilon$, where $\varepsilon \sim \mu_\varepsilon = \mathcal{N}(0, \sigma^2 = 0.1)$. This results in the non-Gaussian posterior

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

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

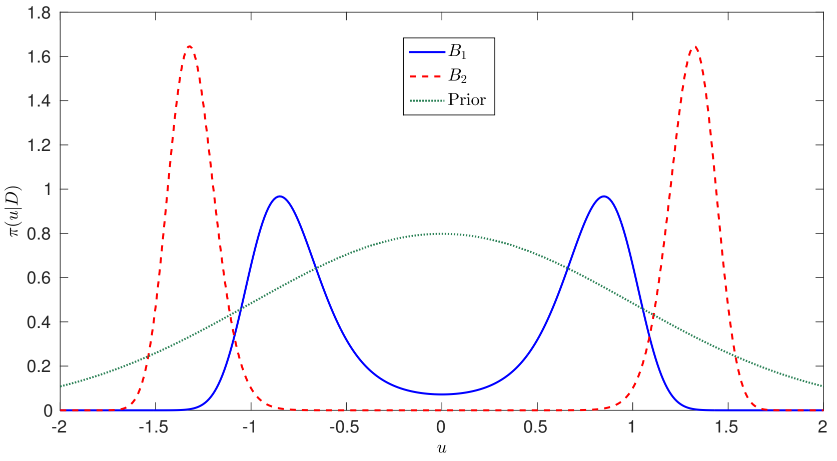


FIG. 6.4. *Posterior distributions for ~~problem BM(1)~~ problems B_1 and ~~BM(2). Problem BM(1) has a noisy data value of 0.921312 which results in an energy barrier which is relatively easy to cross,~~ B_2 as ~~well as their common prior distribution~~described in Section 6.2.1.*

6.2.2. ~~Numerical Implementation~~Calculating values of Optimal β^*. ~~The numerical implementation for most of the following simulations follow the same setup as described in Section 6.1.2. The only exception is that the convergence plots for both adaptive and nonadaptive algorithms are run for 10^6 iterations instead of 10^7.~~

6.2.3. ~~Calculating values of Optimal δ^*~~. Calculating the optimal values of the scaling parameters for this problem is similar to the ~~Gaussian case~~previous example; we check only the acceptance rate to find the optimal values for ~~pCNL~~ RWMH and we use the effective sample size to find the optimal values for ~~PAIS-pCNL~~PAIS-RW. Table ~~??~~ 6.2 gives the optimal values of ~~δ~~ β for both problems. The subscript on β refers to the criterion which has been optimised.

Algorithm	~~δ^*_{acc}~~ β^*_{acc}	~~δ^*_{eff}~~ β^*_{eff}	Algorithm	~~δ^*_{acc}~~ β^*_{acc}	~~δ^*_{eff}~~ β^*_{eff}	~~δ^*_{L2}~~ β^*_{L2}
~~pCNL~~ RWMH	~~1.9~~4.8e-1	-	~~pCNL~~ RWMH	~~5.8e-2~~ 2.3e-1	-	~~9.1~~9.3e-1
~~PAIS-pCNL~~ PAIS-RW	-	~~3.9e-2~~1.0e-1	~~PAIS-pCNL~~ PAIS-RW	-	~~2.6~~5.1e-2	~~2.6e-2~~1.3e-1

TABLE 6.2
Optimal values of ~~δ~~ β for ~~BM~~B_1 ~~(1)~~ (left) and ~~BM~~B_2 ~~(2)~~ (right).

~~In problem BM(1), the pCNL algorithm has the higher value of δ, which corresponds to a close to independence sampling, effectively sampling from the prior. This is~~

~~because the regions with the majority of the target density are well covered by the prior. The PAIS-pCNL algorithm samples more efficiently from its mixture χ, and so a lower value of δ is more efficient.~~

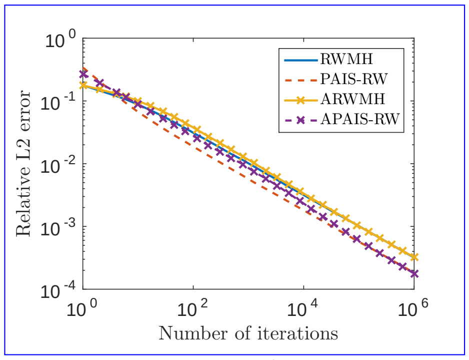
~~Problem BM(2)~~It is relatively simple to find optimal scaling parameters for problem B_1. These values are given in Table 6.2 (left). However problem B_2 is much harder ~~than BM(1);~~ as transitions between the modes are extremely unlikely for the standard ~~pCNL~~ RWMH algorithm. This means that we need to consider the convergence on two levels; we should consider the algorithm's ability to find ~~all~~ both the modes, and ~~to sample them thoroughly and~~ also whether it can sample them in the correct proportions.

To get correctly proportioned modes with the ~~pCNL~~ RWMH algorithm it is important that the chains can transition between the modes frequently, which means that ~~δ~~ β must be large. However, this leads to a lower acceptance rate, and so we sacrifice convergence locally. ~~The prior distribution μ_0 is not a good approximation of the posterior distribution, and so too large a value of δ (which leads to an independence sampler using the prior as a proposal distribution) leads to an inefficient method sampling. For these reasons, the pCNL~~ For this reason, the RWMH algorithm is very slow to converge for problems of this type.

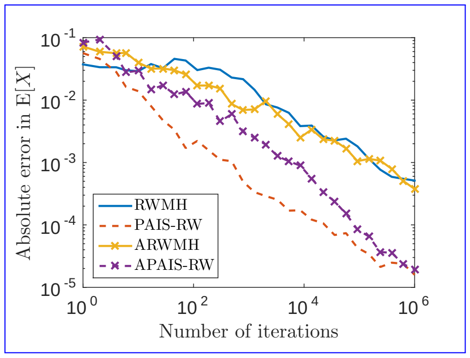
We can achieve these two regimes in ~~pCNL by tuning δ~~ RWMH by tuning β using the acceptance rate for local convergence, and by ~~L2~~ L^2 error for global convergence. Similarly in ~~PAIS-pCNL~~ PAIS-RW we can use the effective sample size for local convergence, and the ~~L2~~ L^2 error for global convergence.

From Table ~~??~~ 6.2 (right) we see that there is a large difference between the optimal value of ~~δ~~ β for each regime ~~, meaning that both~~ in RWMH, and so will result in inefficient sampling. The ~~PAIS-pCNL~~ PAIS-RW algorithm manages to sample the local detail and the large scale behaviour with ~~the same value of δ^*:~~ similar values of β^*; a clear advantage to using this algorithm for this problem.

6.2.3. Convergence of ~~pCNL~~ RWMH vs ~~PAIS-pCNL~~PAIS-RW. ~~We~~



(a) Relative L^2 error.



(b) Absolute error in first moment.

FIG. 6.5. *Convergence of the PAIS-RW and RWMH algorithms for Problem B_1. Set up described in Section 6.1.2. Resampling is performed using the ETPF.*

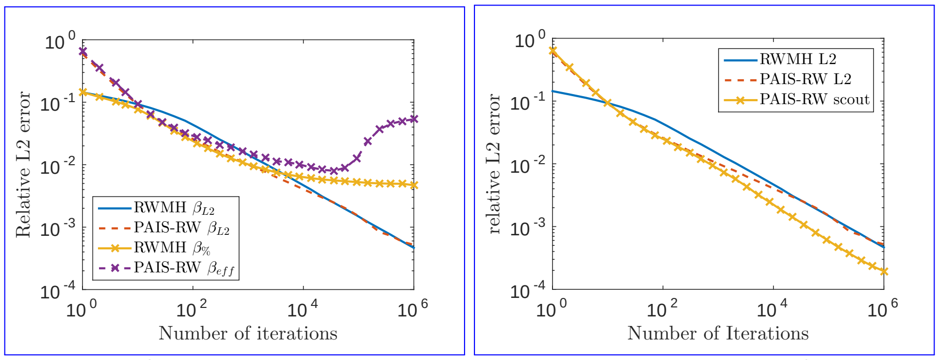
As in the Gaussian example we see a significant speed ~~with the PAIS-pCNL algorithm for BM(1)~~up with the PAIS-RW algorithm for problem B_1. Figure 6.5 shows the adaptive and nonadaptive convergence rates.

~~Error analysis for the PAIS-pCNL and pCNL algorithms for problem BM(1). 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 ?? (2,3).~~

We can see that the adaptive algorithms compare closely with the respective non-adaptive algorithms and the improvement PAIS offers remains significant.

~~For BM(2)~~



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

(b) Comparison of the relative L^2 error of the algorithms optimised by L^2 minimisation with locally optimised PAIS-RW with a scout chain.

FIG. 6.6. *Convergence of the PAIS-RW and RWMH algorithms for Problem B_2. Set up described in Section 6.1.2. Resampling is performed using the ETPF.*

For B_2 the algorithms are run with the global optimal value of ~~δ^*~~ β^*, and with the local optimal value of ~~δ^*~~ β^*. Figure 6.6 (a) shows that the algorithms using the globally optimal ~~δ^* convergence diagnostics converge slowly towards the true posterior after a long burn-in period~~ β^* convergence at the desired rate, whereas the algorithms ~~using the local optimal δ^* converge quickly, but get stuck in one mode meaning that the convergence rate flattens~~ optimised using the acceptance rate and effective sample size initially converge faster but at some point forget the location of one of the modes, causing the convergence to flatten out.

~~pCNL and PAIS-pCNL convergence statistics using locally and globally optimal δ, for problem BM(2). The setup is as described in Section ?? (2).~~

~~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.~~ Using the PAIS algorithm we can minimise the impact of forgotten modes by constantly allowing the algorithm to search them out. Since we have parallel chains, we can run PAIS-RW with the majority of chains using the locally optimal value of β, and one or two chains with a larger scaling parameter. 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 6.6 (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 other problems. Other methods of mode searches are described in [16]~~, and the regeneration method is applicable [21]~~.

~~Figure 6.7 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~~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 24 ensembles with the same proposal distributions, and each group will also have a 'scout' chain with a scaling parameter ten times that of the rest of the group.

~~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 **??** (3).~~

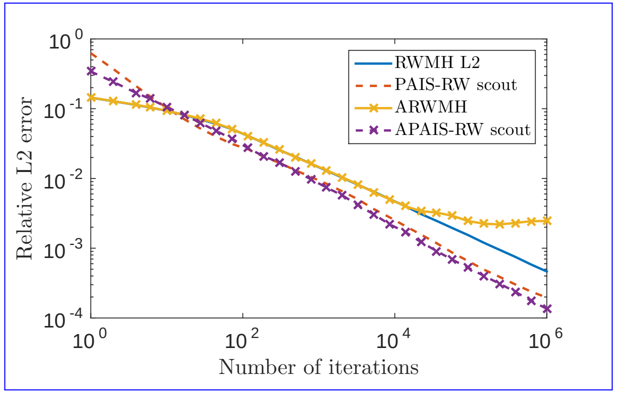


FIG. 6.7. *Convergence of the relative L^2 error for problem B_2, comparing the globally optimised nonadaptive algorithms with the locally optimised adaptive algorithms. The setup is as described in Section 6.1.2 (3). Resampling is performed using the ETPF.*

Comparing the convergence of the adaptive algorithms against the nonadaptive algorithms in Figure 6.7 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 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.

6.2.4. Calculating the Speed Up in Convergence. The graphs in the previous section clearly show that the ~~PAIS-pCNL~~ PAIS-RW algorithm converges faster than the ~~pCNL~~ RWMH algorithm when both are parallelised with the same ~~number of threads~~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 6.8 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 points where the graph has not finished burning in.
A summary of the percentage of iterations required using the PAIS algorithm compared with the respective Metropolis-Hastings algorithms is given in Table 6.3. The blank entries correspond to occasions when ~~the MH algorithms have~~either the MH algorithm or PAIS algorithm hasn't converged to the posterior distribution.

6.2.5. 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)~~ B_2 is that it is unlikely that the correct

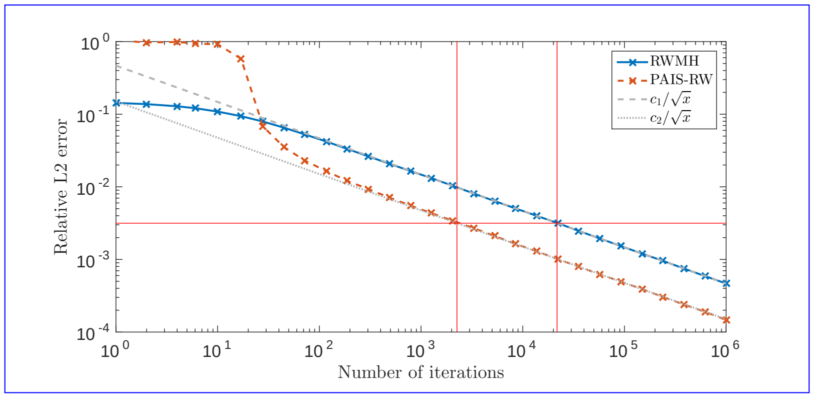


Fig. 6.8. *Illustration of calculating the number of PAIS-pCNL iterations required to reach a tolerance of ~~10^{-2}~~ $10^{-2.8}$ as a percentage of ~~pCNL~~ MH iterations. The relative ~~L2~~ L^2 error graphs are from the Gaussian problem in Section 6.1.*

	Gaussian	~~Low KL div~~ B_1	~~High KL div~~ B_2
RWMH	~~14~~10%	~~14%~~ 32%	12.6% (scout)
pCN	~~33%~~ -	~~MALA 41% 42% pCNL 62% 66% RWMH 40% 44% pCN~~ 32%	-
MALA	42%	36%	40%
pCNL	66%	56%	-

Table 6.3

Iterations for the PAIS algorithms required to achieve a desired tolerance as a percentage of the number of iterations required by the respective ~~Metropolis~~ MH algorithms. The pCN and pCNL proposal distributions are taken from [7].

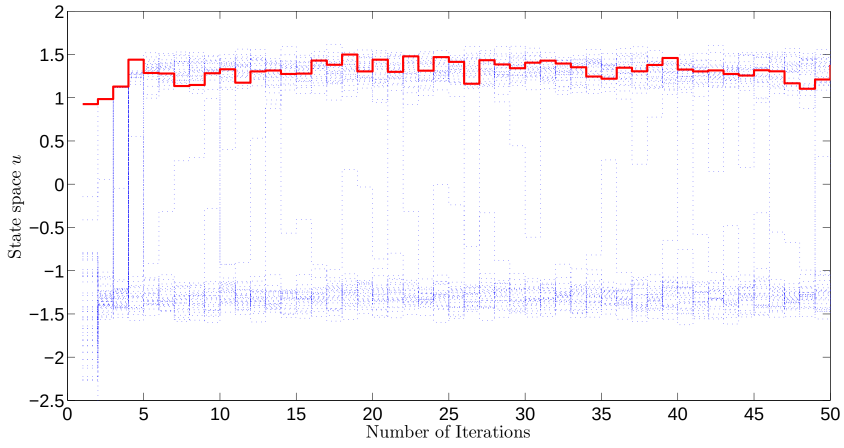


Fig. 6.9. *This figure demonstrates the redistribution property of the PAIS algorithm. Initially there is one chain in the positive mode, and 49 chains in the negative mode.*

ratio of chains will ~~occur~~ 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 6.9 shows Problem ~~BM(2)~~ B_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.

6.3. Sampling from non-Gaussian bivariate distributions. In this section we apply the PAIS algorithm to a more complicated posterior distribution. The field of biochemical kinetics gives rise to multiscale stochastic problems which remain a challenge both theoretically and computationally. Biochemical reactions occur in single cells between a number of chemical populations and the rates of these reactions can vary on vastly different timescales. It is these reaction rates which we are interested in finding descriptions for.

It is often possible to isolate which reactions are occurring more frequently (the fast reactions) and which are occurring less frequently (the slow reactions). The quasi-steady-state assumption (QSSA) is the assumption that the fast reactions converge in distribution on a timescale which is negligible with respect to the rate of occurrence of the slow reactions. This assumption allows us to approximate the dynamics of the slowly changing quantities in the system by assuming that the fast quantities are in equilibrium with respect to the fast reactions in isolation. This kind of model reduction can be used to approximate the likelihood in an inverse problem where we wish to recover the reaction parameters in the system.

Let us consider a simple example by introducing the following simple chemical system involving two chemical species S_1 and S_2:

$$\emptyset \xrightarrow{k_1} S_1 \underset{k_3}{\overset{k_2}{\rightleftarrows}} S_2 \xrightarrow{k_4} \emptyset. \tag{6.2}$$

Each arrow represents a reaction from a reactant to a product, with some rate constant k_i, and where the rates of the reactions are assumed to follow mass action kinetics. We denote the concentration of species S_i by X_i. We assume that we are in a parameter regime such that the reactions $S_1 \to S_2$ and $S_2 \to S_1$ occur much much more frequently than the other reactions. Notice that both chemical species are involved in fast reactions. However, the quantity $\mathcal{S} = X_1 + X_2$ is conserved by both of the fast reactions, and as such, this is the slowly changing quantity in this system. The effective dynamics of $\mathcal{S}$ can be represented as follows.

$$\emptyset \xrightarrow{k_1} \mathcal{S} \xrightarrow{\hat{k}_4} \emptyset. \tag{6.3}$$

Here, the new reaction rate $\hat{k}_4$ is approximated through application of the QSSA to be

$$\hat{k}_4(s) = \mathbb{E}\left[k_4 X_2 | \mathcal{S} = s\right] = \frac{k_2 k_4 \mathcal{S}}{k_2 + k_3}.$$

The value of $\mathbb{E}[k_4X_2|\mathcal{S}=s]$ is approximated by finding the steady state of the ODE representing the fast subsystem of reactions:

$$S_1 \underset{k_3}{\overset{k_2}{\rightleftarrows}} S_2, \qquad X_1 + X_2 = s$$

If we assume that we know the rate constants k_1 and k_4, and observe the system in (6.3), we indirectly observe the rates k_2 and k_3 through the effective rate $\hat{k}_4$ of the degradation of $\mathcal{S}$. Our observations are uninformative about these reaction rates, as there are surfaces in parameter space along which the effective rate $\hat{k}_4$ is invariant, leading to a highly ill-posed inverse problem. Making the assumption that the errors in our observations of the value of $\mathcal{S}$ are Gamma distributed with some variance σ^2, this results in a long and thin posterior distribution on k_2 and k_3. This type of problem is notoriously difficult to sample from using standard MH algorithms, as the algorithms quickly find a point on this manifold on which $\hat{k}_4$ is invariant, but exploration along its length is slow.

6.3.1. Target Distribution. We now formalise the posterior of interest. We look for a distribution over the parameter $\mathbf{k} = (k_2, k_3)^T$, given that we know $k_1 = 100$ and $k_4 = 1$. We generate our data by making ten observations of the system at $t_i = 2, 4, \dots, 20$, here simulated by solving the full system, (6.2), governed by the differential equations

$$\frac{\mathrm{d}S_1}{\mathrm{d}t} = k_1 - k_2X_1(t) + k_3X_2(t),$$

$$\frac{\mathrm{d}S_2}{\mathrm{d}t} = k_2X_1(t) - (k_3 + k_4)X_2(t),$$

with initial conditions $X_1(0) = X_2(0) = 0$, and parameter values $\mathbf{k} = (50, 100)^T$. We then add noise taken from a Gamma distribution with variance $\sigma^2 = 225$ and centred at each $\mathcal{S}(t_i) = X_1(t_i) + X_2(t_i)$.
In our modelling we use the QSSA to simplify this system of differential equations into the one dimensional system based on (6.3),

$$\frac{\mathrm{d}\mathcal{S}}{\mathrm{d}t} = k_1 - \frac{k_2k_4}{k_2 + k_3}\mathcal{S}(t).$$

The observation operator, $\mathcal{G} : (\mathbf{k}, t) \mapsto S(t)$, maps from parameter space onto population space at a time t. This means that for the ith observation we assume,

$$D_i \sim \text{Gamma}(\alpha_i, \beta_i),$$

where

$$\alpha_i = \frac{\mathcal{G}(\mathbf{k}, t_i)^2}{\sigma^2}, \quad \beta_i = \frac{\mathcal{G}(\mathbf{k}, t_i)}{\sigma^2}, \quad i = 1, \dots, 10.$$

These are parameters required to give us a distribution with mean $\mathcal{G}(\mathbf{k}$ and variance σ^2. We assign Gamma priors to $\mathbf{k}$ with mean $\alpha_0/\beta_0 = 75$ and variance $\alpha_0/\beta_0^2 = 100$

in both coordinates, resulting in the posterior

$$\pi(\mathbf{k}|\mathbf{D}) \propto \left[\prod_{i=1}^{10} \mathrm{Gamma}(D_i; \alpha_i, \beta_i)\right] \mathrm{Gamma}(k_2; \alpha_0, \beta_0)\mathrm{Gamma}(k_3; \alpha_0, \beta_0).$$

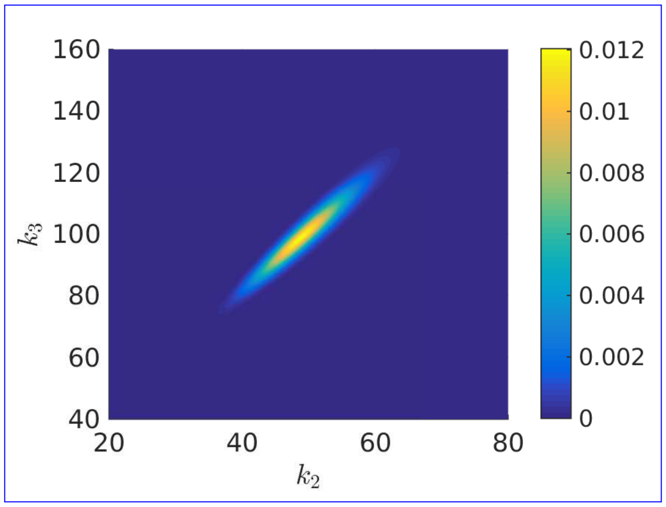


FIG. 6.10. *The posterior distribution on the parameters k_2 and k_3 given the data and priors described in Section 6.3.1.*

Figure 6.10 presents a visualisation of the posterior distribution for this problem, found by exhaustive MH simulation.

6.3.2. Implementation. For this problem there is no analytic form for the normalisation constant, and numerical methods implemented in MATLAB and Mathematica have proven to be unreliable. To demonstrate the convergence of the algorithm, we first run a MH algorithm for much longer than we normally would (8×10^{10} samples), and consider the histogram produced to be a well-converged approximation of the true posterior distribution. We then perform our usual simulations with the PAIS and MH algorithms to compare the rate at which the histograms converge to the posterior over the same mesh.
In this problem we modify our PAIS algorithm to use a mixture of Gamma distributions in the proposal distribution instead of a Gaussian mixture. The PAIS-Gamma and MH-Gamma algorithms use a Gamma proposal distribution with mean centred at the previous state,

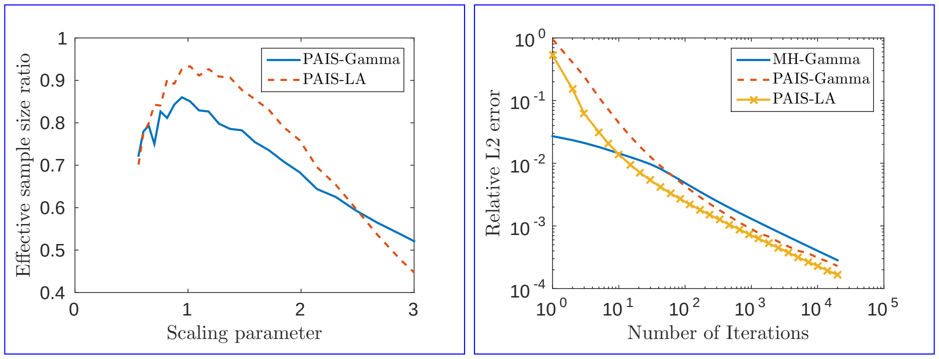
$$y \sim \mathrm{Gamma}(\cdot; \alpha^*, \beta^*) \quad \text{where} \quad \frac{\alpha^*}{\beta^*} = x \text{ and } \frac{\alpha^*}{(\beta^*)^2} = \beta^2.$$

Similarly the PAIS-LA algorithm uses the Langevin proposal distribution from the standard MALA algorithm with a perturbationperturbation taken from a Gamma distribution,

$$y \sim \mathrm{Gamma}(\cdot; \alpha^*, \beta^*), \quad \text{where} \quad \frac{\alpha^*}{\beta^*} = x + \frac{1}{2}\beta^2 \nabla \log \pi(X) \text{ and } \frac{\alpha^*}{(\beta^*)^2} = \beta^2.$$

This ensures that we only propose positive parameters. Having a heavier right tail in the proposal distribution also means that the posterior is absolutely continuous with respect to the proposal. When this is not the case, the weight function can tend to zero or infinity when $k_i \to \infty$, which results in poor, very spiky approximations of the tails of the posterior distribution leading to slow convergence.
We now significantly increase the ensemble size we are using from $M = 50$ to $M = 2500$. This allows us to build a better approximation of the posterior distribution for our proposals. Following the discussion in Section 5, it is clear that to keep the runtime of the resampler negligible compared with the calculation of the posterior we must switch from the ETPF resampler to the AMR algorithm.

6.3.3. Convergence of PAIS-Gamma and PAIS-LA vs MH with Gamma proposals. In the numerics which follow we have used the AMR algorithm to resample with an ensemble size of $M = 2500$. The method otherwise remains the same as in previous sections. We perform test runs to find the optimal scaling parameters for both Gamma proposals and MALA-type proposals. We then calculate the convergence rates of the algorithms by producing 50 million samples from the posterior with each algorithm, and repeat the simulation 32 times.



(a) Optimal values of the scaling parameter. (b) Relative L^2 error against number of iterations.

FIG. 6.11. *Convergence of the PAIS-Gamma and PAIS-LA algorithms for the chemical system problem described in Section 6.3. Implementation described in Sections 6.3.2 and 6.3.3. Resampling is performed using the AMR scheme.*

Algorithm	MH-Gamma	PAIS-Gamma	PAIS-LA
β^*	2.7e-0	9.4e-1	1.0e-0

TABLE 6.4
Optimal values of the scaling parameter. The MH algorithm is optimised using the acceptance rate, and the PAIS algorithms are optimised using the effective sample size.

In Figure 6.11 (a) we see that the two PAIS algorithms have similar optimal values of the scaling parameter, also shown in Table 6.4. However, the MALA-type proposal achieves a higher effective sample size with the same ensemble size. Convergence after the first 50 million samples is shown in Figure 6.11 (b) and demonstrates that the PAIS algorithm converges faster than the MH algorithm. The slightly unstable convergence in the PAIS-Gamma algorithm is the effect of spikes appearing in the tails of the posterior distribution. This is also what has caused the slightly lower effective sample size.

7. 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 ~~Random Walk Metropolis-Hastings (RWMH),~~ Metropolis Adjusted Langevin Algorithm (MALA)~~and~~ , preconditioned Crank-Nicolson (pCN), preconditioned Crank-Nicolson Langevin (pCNL) and Hamiltonian Monte Carlo (HMC) proposals has yielded similar results [29].

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~~ M-fold increase, where ~~N~~ M is the number of ~~cores or processors at our disposal~~ensemble members used. Our approach demonstrates a better-than-linear speed-up with the number ~~processors/cores used. Thus, our approach is not only embarrassingly parallel (as coined by Cleve Moler in [18]), but humiliatingly so, provided that an optimal transport algorithm can be efficiently implemented in parallel and so does not dominate the communication costs.~~ 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 ~~chains~~ 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 (HMC, 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 disadvantage of parallelised algorithms is that often different processors will complete their tasks in different amount of times, for a number of reasons, often due to communication between the processors. One approach which could be applied to the PAIS to avoid this is for each processor to immediately start a new iteration, only using the last values of the other processors that were communicated to it. This incomplete approach would still be valid, and could lead to more efficient use of the computer architecture.~~

One limitation of the PAIS approach as described above is that a direct solver of the ETPF problem (such as FastEMD [22]) has computational cost ~~$\mathcal{O}(n^3 \log n)$, where n~~ $\mathcal{O}(M^3 \log M)$, where M is the number of particles in the ensemble. ~~The use of approximate solutions from iterative solvers might allow the PAIS to be used in higher dimensional problems, and is an avenue for future investigation. A second limitation is associated with the problem of accurately approximating measures with empirical measures in high dimensional phase space. This is a well-known issue in filtering, where one possible solution is to "localise" the impact of observations, as rigorously analysed in [23]. An established technique in Ensemble Kalman filters, localisation has also been incorporated into the Ensemble Transform Particle Filter [4]. This issue could be addressed in the PAIS by using localisation when computing the transform, but keeping the unlocalised weights in the importance sampling for consistency.~~ 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] A. Beskos, F. Pinski, J. Sanz-Serna, and A. Stuart, *Hybrid Monte Carlo on Hilbert spaces*, Stochastic Processes and their Applications, 121 (2011), pp. 2201–2230.

[2] 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.

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

[4] Y. Chen and S. Reich, *Data assimilation: a dynamical system perspective*, Frontiers in Applied Dynamical Systems, (to appear).

[5] 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.

[6] 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.

[7] 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.

[8] T. El Moselhy and Y. Marzouk, *Bayesian inference with optimal maps*, Journal of Computational Physics, 231 (2012), pp. 7815–7850.

[9] 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.

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

[11] 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.

[12] 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.

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

[14] 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.

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

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

[17] 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.

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

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

[20] 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).

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

[22] 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.

[23] P. Rebeschini and R. Van Handel, *Can local particle filters beat the curse of dimensionality?*, arXiv preprint arXiv:1301.6585, (2013).

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

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

[26] ———, *Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms*, Journal of

Applied Probability, 44 (2007), pp. 458–475.
[27] ———, *Examples of adaptive MCMC*, Journal of Computational and Graphical Statistics, 18 (2009), pp. 349–367.
[28] J. Rosenthal, *Optimal proposal distributions and adaptive MCMC*, Handbook of Markov Chain Monte Carlo, (2011), pp. 93–112.
[29] P. Russell, *PhD Thesis*, PhD thesis, School of Mathematics, University of Manchester, 2017.
[30] J. Sexton and D. Weingarten, *Hamiltonian evolution for the hybrid Monte Carlo algorithm*, Nuclear Physics B, 380 (1992), pp. 665–677.
[31] A. Stuart, *Inverse problems: a Bayesian perspective*, Acta Numerica, 19 (2010), pp. 451–559.
[32] C. Villani, *Topics in Optimal Transportation*, Graduate studies in mathematics, American Mathematical Society, 2003.
[33] ———, *Optimal Transport: Old and New*, Grundlehren der mathematischen Wissenschaften, Springer Berlin Heidelberg, 2008.

8. ~~The adaptive PAIS algorithm.~~

~~$x_j^{(0)} \sim \mu_0$, for $j \in L \cup U$, where $L = \{j\}_{j=1}^{M/2}$, $U = \{j\}_{j=M/2+1}^{M}$. Choose $\delta^{(1)} \in (0,2]$. Set $\delta_{L,U}^{(1)} = (1 \pm 0.01)\delta^{(1)} \wedge 2$. $y_j^{(i)} \sim Q(x_j^{(i-1)}, \delta_L^{(i)})$ for $j \in L$, and $y_j^{(i)} \sim Q(x_j^{(i-1)}, \delta_U^{(i)})$ for $j \in U$. Calculate~~

$$w_j^{(i)} = \frac{\pi(y_j^{(i)})}{\nu(y_j^{(i)}; X^{(i-1)})},$$

~~where~~

$$\nu(y; X) = \frac{1}{M}\sum_{j\in L} q(y; x_j, \delta_L) + \frac{1}{M}\sum_{j \in U} q(y; x_j, \delta_U).$$

~~For $w_{kj} = w_j^{(k)}$, $S = n_k - n_{k-1}$, $T_L = SM(\sum_{k=i-S}^{S}\sum_{j\in L} w_{kj})^2/(\sum_{k=i-S}^{S}\sum_{j\in L} w_{kj}^2)$. $T_U = SM(\sum_{k=i-S}^{S}\sum_{j\in U} w_{kj})^2/(\sum_{k=i-S}^{S}\sum_{j\in U} w_{kj}^2)$. $\delta^{(i+1)} = \delta^{(i)} - \Delta t \dfrac{T_U - T_L}{\delta_U^{(i)} - \delta_L^{(i)}}$. $\delta^{(i+1)} = \delta^{(i)}$.~~

~~Resample~~

$$(w_j^{(i)}, y_j^{(i)})_{j\in L} \to (\frac{1}{M}, x_j^{(i)})_{j \in L}, \quad (w_j^{(i)}, y_j^{(i)})_{j\in U} \to (\frac{1}{M}, x_j^{(i)})_{j \in U}.$$

~~Resample $(w^{(i)}, Y^{(i)}) \to (\frac{1}{M}\mathbf{1}, X^{(i)})$. $\delta_{L,U}^{(i)} = \delta^{(i)} \pm 2\sqrt{2}\delta^{(i)}/\sqrt{NM} \wedge 2$. Resample $(w^{(i)}, Y^{(i)}) \to (\frac{1}{M}\mathbf{1}, X^{(i)})$. $\delta_L^{(i+1)} = \delta_U^{(i+1)} = \delta^{(i+1)}$. A pseudo-code representation of the adaptive PAIS algorithm.~~