

Accelerating Markov Chain Monte Carlo sampling with diffusion models

N. T. Hunt-Smith,^a W. Melnitchouk,^b F. Ringer,^{b,c} N. Sato,^b A. W. Thomas,^a M. J. White^a

^a*CSSM and ARC Centre of Excellence for Dark Matter Particle Physics, University of Adelaide, North Terrace, SA 5005, Australia*

^b*Jefferson Lab, Newport News, Virginia 23606, USA*

^c*Department of Physics, Old Dominion University, Norfolk, Virginia 23529, USA*

E-mail: nicholas.hunt-smith@adelaide.edu.au, wmelnitc@jlab.org,
fmringer@jlab.org, nsato@jlab.org, anthony.thomas@adelaide.edu.au,
martin.white@adelaide.edu.au

ABSTRACT: Global fits of physics models require efficient methods for exploring high-dimensional and/or multimodal posterior functions. We introduce a novel method for accelerating Markov Chain Monte Carlo (MCMC) sampling by pairing a Metropolis-Hastings algorithm with a diffusion model that can draw global samples with the aim of approximating the posterior. We briefly review diffusion models in the context of image synthesis before providing a streamlined diffusion model tailored towards low-dimensional data arrays. We then present our adapted Metropolis-Hastings algorithm which combines local proposals with global proposals taken from a diffusion model that is regularly trained on the samples produced during the MCMC run. Our approach leads to a significant reduction in the number of likelihood evaluations required to obtain an accurate representation of the Bayesian posterior across several analytic functions, as well as for a physical example based on a global analysis of parton distribution functions. Our method is extensible to other MCMC techniques, and we briefly compare our method to similar approaches based on normalizing flows. A code implementation can be found at <https://github.com/NickHunt-Smith/MCMC-diffusion>.

Contents

1	Introduction	1
2	A brief introduction to diffusion models	3
3	A diffusion-model assisted Markov Chain Monte Carlo sampler	4
4	Analytic test functions	7
4.1	2D Himmelblau function	7
4.2	10D Gaussian mixture	8
4.3	4D EggBox function	10
5	Physics example: global parton distribution function fit	12
6	Conclusions	14
A	Hyperparameters	17
B	4D Rosenbrock function	18

1 Introduction

A large number of scientific theories exist as parametric models, in which predictions depend on the values of a set of free parameters that are *a priori* unknown. To progress our understanding of these theories, it is necessary to infer the values of model parameters by comparing theoretical predictions to one or more sets of observed data. Rigorous global statistical fits of theories of interest in particle astrophysics typically involve tens to hundreds of parameters (if one includes nuisance parameters), and frequently require the simulation of a wide range of data in collider physics, flavor physics, astrophysics and cosmology [1, 2]. The computational expense of this process necessitates the development of fast and accurate sampling techniques that can reduce the time required to adequately explore large parameter space volumes.

In the Bayesian statistical framework, parameter inference is performed by mapping the posterior function, defined as:

$$p(\boldsymbol{\theta}|\mathbf{D}) = \frac{\mathcal{L}(\mathbf{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int d\boldsymbol{\theta} \mathcal{L}(\mathbf{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}, \quad (1.1)$$

where \mathbf{D} is a set of observed data, and $\boldsymbol{\theta}$ is the set of parameters of the model. The *prior* $p(\boldsymbol{\theta})$ gives the *a priori* degree of belief that the parameters take certain values (often chosen to be a flat or logarithmic distribution), and $\mathcal{L}(\mathbf{D}|\boldsymbol{\theta})$ is the likelihood of the data given a set

of parameters, which is conventionally obtained by simulation. The denominator, known as the *Bayesian evidence*, is a normalisation constant that ensures that the function $p(\boldsymbol{\theta}|\mathbf{D})$ is a valid probability density function.

Except for simple examples, the posterior is not known or easily calculable analytically. It can instead be obtained through sampling (see Ref. [3] for a comprehensive review of algorithms). Amongst the most popular class of technique is Markov Chain Monte Carlo sampling [4]. This proceeds by starting at an initial point in the parameter space and proposing new samples from a proposal density that is easy to sample, before accepting them using a condition that must satisfy strict properties. A suitable acceptance condition ensures that the samples are distributed according to the posterior density. In particle astrophysics applications, the Metropolis-Hastings algorithm is particularly popular [5–7].

A well-known deficiency of MCMC methods concerns multi-modal posterior functions. A proposal function based on a local proposal from the current point in the Markov Chain tends to lead to one mode being found, but often has low efficiency for jumping between modes. An uninformed, nonlocal proposal distribution, meanwhile, will typically lead to a high rejection rate. Finding more efficient proposal functions for the exploration of multi-modal posteriors reduces the time required to adequately explore the parameter space, hence allowing problems of greater complexity to be tackled.

In this paper, we present a new algorithm that augments a Metropolis-Hastings sampler with a proposal function based on a diffusion model. A diffusion model is a parametrized Markov chain trained to produce samples matching a set of input data after finite time. Such models have recently been shown to be highly effective in image synthesis, including outperforming previous approaches based on generative adversarial networks [8–15]. By creating a diffusion model from the available samples at regular intervals during a Metropolis-Hastings sampling run, one can develop an approximation of the posterior that can be used as an informed global proposal function to increase the efficiency of MCMC sampling. This is not the only way one could perform posterior sampling using diffusion models; one could instead use stochastic differential equations to gradually approximate the posterior via diffusion processes [16]. An alternative approach based on the use of normalizing flows for lattice QCD was previously presented in Ref. [17–19], and a similar combination of normalizing flows with MCMC sampling was outlined in Ref. [20, 21]. Normalizing flows have in fact been combined with diffusion models, though not in the context of MCMC [22]. We note that there are a number of important differences between diffusion models and normalizing flows:

- A flow-based approach requires the estimation of gradients, while for diffusion models this is unnecessary. This is a highly useful feature when considering, for example, particle astrophysics global fits which often use stochastic simulations where gradient information is impossible to obtain [23–25].
- A fully-trained normalizing flow has direct access to a probability distribution function that approximately matches the target function [17], while a fully-trained diffusion model can only produce samples that are approximately distributed according to the target function.

- Normalizing flows can be subject to topological limitations, particularly for multi-modal functions [26]. Diffusion models are more successful in matching the shape of sharply-peaked or multi-modal functions.
- Diffusion models can be slow to train compared to normalizing flows given the need to train on many time-discretization steps [22]. We address this concern by fitting parameters that define the variance of the noise added in each time step, rather than training a neural network. This speeds up the training process dramatically.

We benchmark our improved technique against regular Metropolis-Hastings sampling, but one could easily accelerate other MCMC samplers using the same diffusion model technique presented here.

This paper is structured as follows. In Section 2, we provide a brief, self-contained introduction to diffusion models. In Section 3, we present our algorithm that marries a diffusion model to an MCMC sampler. In Section 4, we perform a series of numerical tests of our new algorithm by attempting to map various analytic test functions. We compare the performance of our algorithm in these tests to other basic MCMC samplers, as well as to a specific normalizing flow algorithm. In Section 5, we extend the numerical tests to a physical example relating to a global fit of parton distribution function (PDF) parameters. We summarize our conclusions in Section 6.

2 A brief introduction to diffusion models

Diffusion models are a form of *generative model* that, after training on a set of data, are able to reproduce examples of that data. Starting with, for example, an image, the key concept in a diffusion model is to systematically decay the image into pure noise through the repeated addition of a noise component (usually chosen to be Gaussian). It is then possible to learn the reverse of this noise addition process and recover the original information. Examples of valid images can then be obtained by feeding pure noise distributions through the learned reverse process.

To put this on a more precise footing, assume that we start with a data vector \mathbf{x}_0 . We can then refer to a data probability distribution $q(\mathbf{x}_0)$, which is the distribution of possible “real” data vectors. Any sample from this distribution is itself a valid data vector, and a sample from this distribution is written as $\mathbf{x}_0 \sim q(x_0)$. We now want to add Gaussian noise for T successive time steps. We can define a *forward diffusion process* $q(\mathbf{x}_t | \mathbf{x}_{t-1})$ which adds the noise at time step t . Note that we can formally define this as a Markov chain, since the prediction of the probability density at time t only depends on the immediate predecessor at time $t - 1$. In principle, the width of the Gaussian noise that we add at each time step can be different, so we further define a *variance schedule* $0 < \beta_1 < \beta_2 < \dots < \beta_T < 1$ where each β_t is set to the same value across each parameter, then define the forward diffusion process as:

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}). \quad (2.1)$$

As time progresses, each new data vector at time step t is drawn from a Gaussian distribution with mean $\mu_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1}$ and variance $\sigma_t^2 = \beta_t$. Starting from the original vector

\mathbf{x}_0 , we generate a series of increasingly noisy vectors $\mathbf{x}_1, \dots, \mathbf{x}_t, \dots, \mathbf{x}_T$, where the aim is to have \mathbf{x}_T be pure Gaussian noise.

We know the forward process very well, since it is easy to apply Gaussian noise. However, what about the reverse process? Let the probability distribution that takes us backwards along the chain of images be $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$. In general, $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ cannot be known analytically, since we would need to know the distribution of all possible images in order to calculate it. Instead, we can introduce an approximation to the reverse transformation $p_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t)$, with ϕ representing the parameters of a given model for the reverse process. The model is typically provided by a trained neural network, although in principle one could parametrize a model that appropriately describes each reverse step.

For Gaussian diffusion in the continuous limit (*i.e.*, in the limit of small step size, β), the reversal of the diffusion process has the identical functional form as the forward process [8, 27]¹. The reverse probability distribution can therefore be described by a Gaussian distribution with a mean $\boldsymbol{\mu}_\phi$ and a variance Σ_ϕ :

$$p_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\phi(\mathbf{x}_t, t), \Sigma_\phi(\mathbf{x}_t, t)). \quad (2.2)$$

The mean and variance are also functions of the time t , since the reverse transformation is slightly different at each time slice.

3 A diffusion-model assisted Markov Chain Monte Carlo sampler

For testing the effectiveness of diffusion model accelerated MCMC sampling, we paired the diffusion model with a simple Metropolis-Hastings (MH) MCMC algorithm. In a MH algorithm, the proposed next point $\boldsymbol{\theta}'$ in the MCMC chain is obtained by sampling from a proposal function $Q(\boldsymbol{\theta}'|\boldsymbol{\theta})$ that depends only on the previous point $\boldsymbol{\theta}$. For a given posterior P , the proposed point is then accepted with probability [28]

$$a = \min \left(1, \frac{P(\boldsymbol{\theta}')}{P(\boldsymbol{\theta})} \cdot \frac{Q(\boldsymbol{\theta}|\boldsymbol{\theta}')}{Q(\boldsymbol{\theta}'|\boldsymbol{\theta})} \right). \quad (3.1)$$

For Q we adopt the typical choice of a Gaussian proposal function centred on $\boldsymbol{\theta}$, which as a symmetric function reduces the Q term in Eq. (3.1) to unity. We identify this algorithm with a Gaussian Q as *pure MH*. To incorporate a diffusion model into this process, we occasionally replace the Gaussian proposal function with a diffusion model, which is periodically re-trained as the MCMC chain accumulates samples. These “diffusion proposal samples” are generated from random noise via the diffusion model as described in Section 2, and as such each diffusion proposal sample is completely independent of the previous sample. This is a special case of MH, called *independence MH*, which simplifies the Q factors in Eq. (3.1) to $Q(\boldsymbol{\theta})/Q(\boldsymbol{\theta}')$ [29]. As long as the diffusion proposal samples are run through the MH accept/reject step, the distribution of samples is guaranteed to satisfy detailed balance and converge to the posterior eventually [29, 30]. In addition, as the overall samples get closer to the true posterior, the diffusion model is continually being trained to produce

¹Note that this is also true of binomial diffusion.

samples that more closely resemble the true posterior. Eventually, the diffusion model will be asymptotically exact — that is, the diffusion model samples will be distributed such that they approximate the target posterior, in which case $P(\boldsymbol{\theta}) \approx Q(\boldsymbol{\theta})$ [18]. In the limit of many samples, the acceptance rate of the diffusion samples will be close to 1. One can think of this process as a method of correcting the proposal distribution to match the desired distribution P .

Since the diffusion proposal samples are distributed according to our knowledge of the posterior, they will only be symmetric in the cases where the posterior is also symmetric. Therefore, in most situations the Q term will not simply reduce to unity when using the diffusion model as a proposal function. We can approximate the probability of proposing a given sample $Q(\boldsymbol{\theta})$ with a Gibbs sampling method. For a given proposal vector in D dimensions $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \dots, \theta_D)$, we first draw a number of diffusion proposal samples. We then take a 1D histogram of those samples to find the approximate marginal distribution of the θ_1 parameter. $Q(\theta_1)$ is then estimated by dividing the number of samples in the bin that θ_1 falls into by the total number of samples. Following this step, we set θ_1 at its specific value and simulate from the next parameter θ_2 , before taking another 1D histogram of the θ_2 parameter to estimate $Q(\theta_2|\theta_1)$. We repeat this D times, before multiplying the probabilities together according to Bayes' rule,

$$Q(\boldsymbol{\theta}) = Q(\theta_1) \times Q(\theta_2|\theta_1) \times Q(\theta_3|\theta_1, \theta_2) \times \cdots \times Q(\theta_D|\theta_1, \theta_2, \theta_3, \dots, \theta_{D-1}). \quad (3.2)$$

This method was chosen primarily to reduce computation time and avoid having to make use of high-dimensional histograms.

Retaining a mix of diffusion model proposals and Gaussian proposals is necessary because the diffusion model can only mimic the distribution of samples it has been provided with. Hence, at early points in the MCMC run, the diffusion model will not provide an adequate approximation of the final posterior, and therefore samples taken from it will not have a particularly high acceptance rate. Later in the run, however, the diffusion model will provide a more accurate representation of the posterior, and the acceptance rate will improve. Inspired by the algorithmic structure in Refs. [20, 21], we therefore designed Algorithm 1, which uses pure MH some of the time alongside the diffusion proposal samples.

In Algorithm 1, n is the total number of samples to be generated, p_{diff} is the probability of using a diffusion proposal sample as opposed to a local Gaussian proposal sample, and τ is the number of samples to generate before retraining the diffusion model. τ is tunable to the posterior one wishes to sample from, with lower values of τ resulting in better performance at the cost of computation time to retrain the diffusion model.

The primary advantage of using the diffusion model over standard MCMC is the non-local nature of its proposed points within a single chain. Where a standard MCMC chain might get stuck in a local mode, the diffusion model can jump to any point in parameter space. Algorithms such as Explore-Exploit MCMC (Ex²MCMC) [31] and normalizing flows (described in Sec. 1) make use of this global transition kernel. For multi-modal posteriors, this means one can accurately characterize the relative weights between the modes with only a single chain. Our only requirement occurs if the modes are distanced too far apart for the Gaussian proposal to jump between them with a significant probability. In that case,

Algorithm 1

```
1: for  $i = 1, \dots, n$  do
2:   Generate a random number  $r$  from the uniform distribution  $\mathcal{U}(0, 1)$ 
3:   if  $r < p_{\text{diff}}$  then
4:     Metropolis-Hastings with diffusion model proposal
5:   else
6:     Pure Metropolis-Hastings with Gaussian proposal
7:   end if
8:   if  $i \bmod \tau = 0$  then
9:     Retrain diffusion model on existing samples
10:  end if
11: end for
```

all the important modes must be located beforehand in order to seed the diffusion model with points at those modes for initial training, allowing transitions between the modes.

The diffusion model can also accelerate the MCMC process even for posteriors with a single mode. The proposed diffusion samples are more likely to be located in strongly peaked regions of the parameter space, meaning that if the MCMC chain strays far away from the mode the diffusion samples will bring it back to the region of interest. This is particularly significant for computationally expensive posteriors, where reducing the number of likelihood evaluations is the highest priority. A further advantage of the diffusion model is that, once the sampling algorithm has converged, the diffusion model has already been trained to produce samples that closely resemble the posterior. We therefore obtain a fast way of generating as many approximate posterior samples as desired. This is particularly useful in applications where a rough initial scan can be used to define a much smarter prior for a subsequent scan. Global fits of parameter spaces which exhibit a degree of fine-tuning are an obvious use-case.

Applying a diffusion model to MCMC samples rather than images is a considerably simpler task in terms of algorithmic complexity. A single high-quality image consists of millions of pixels in several color channels which must be converted into a high-dimensional tensor, and millions of training images must be supplied in order for the diffusion model to learn and then generate an image that looks somewhat like the training images. On the other hand, our MCMC samples are simple low-dimensional arrays with a length equal to the number of dimensions of the problem; rarely more than a few hundred for realistic physics examples. We have therefore constructed a diffusion model that is tailored towards MCMC samples rather than attempting to adapt existing image-based diffusion models.

When we are training the diffusion model, we define the forward Gaussian noise process by adding noise to the samples in the same way as Eq. (2.1). In other words, each vector in the overall set of samples that form the Markov chain is corrupted slightly at each time step. This forward process will be concluded once the distribution of the vectors has been converted from a shape that resembles the posterior to pure Gaussian noise. This introduces two hyperparameters which can be tuned as appropriate: the number of time steps and

the amount of noise progressively added at each time step, β_T . Generally, increasing these values will further reduce the minimum loss, but also increase the computation time needed to find the minimum.

To train the diffusion model, we first perform the forward diffusion process according to Eq. (2.1) until we are left with pure Gaussian noise. On an individual sample level,

$$\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}, \quad (3.3)$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. We then define a new data vector \mathbf{y}_T drawn from pure Gaussian noise with the same length as \mathbf{x}_T . Each reverse step is then computed using

$$\mathbf{y}_{t-1} = \mathbf{y}_t - (\mathbf{x}_t - \mathbf{x}_{t-1})\phi_t. \quad (3.4)$$

Here, the difference in samples between each time step ($\mathbf{x}_{t-1} - \mathbf{x}_t$) is multiplied by a free parameter ϕ_t to be trained. Repeating this operation T times leads to the denoised samples \mathbf{y}_0 . We can define a loss function L as

$$L = \sum_t (\mathbf{y}_{t-1}(\phi_t) - \mathbf{x}_{t-1})^2, \quad (3.5)$$

summed over the total number of samples. Rather than training a neural network, we therefore perform a series of fits of the ϕ_t parameters to minimise the sum of squared differences in each time step between the denoised samples we have generated and the original MCMC samples. There are $T \times D$ parameters to be fit simultaneously for each retrain of the diffusion model. To minimise the loss function we use the `SciPy minimize` function.

We then perform the reverse process by generating another new data vector \mathbf{z}_T from pure Gaussian noise, before applying Eq. (3.4) to obtain \mathbf{z}_0 . We can repeat this process indefinitely, generating as many samples as required almost instantaneously. This reverse process is considerably simpler than the typical method of parametrizing reverse Gaussian transformations described in Eq. (2.2). Empirically, we found it to be highly effective at generating unique diffusion samples approximately distributed according to the target distribution.

4 Analytic test functions

In the test functions below, we explore the advantages Algorithm 1 has over pure MH, as well as over a normalizing flow MCMC algorithm. The values assigned for each hyperparameter in each function are contained in Appendix A. A further test comparing Algorithm 1 and pure MH for the 4D Rosenbrock function can be found in Appendix B.

4.1 2D Himmelblau function

The Himmelblau function is given by

$$f(\boldsymbol{\theta}) = (\theta_1^2 + \theta_2 - 11)^2 + (\theta_1 + \theta_2^2 - 7)^2, \quad (4.1)$$

which consists of 4 sharply peaked modes of equal height spaced a large distance away from each other, presenting a significant challenge for MCMC samplers. Without multiple chains, we have found that a simple MCMC algorithm like pure MH will either get stuck in one of the modes if the Gaussian proposal is too narrow, or have a very low acceptance rate if the Gaussian proposal is too wide. Augmenting the MH sampler with a diffusion model as in Algorithm 1 allows the chain to jump between these modes, while accurately characterizing the relative weights between the modes. This behavior is shown in Fig. 1, which depicts 10,000 samples of the 2D Himmelblau from a single chain using Algorithm 1. The nonlocal jumps made by the diffusion model are also represented visually by the black lines connecting the 4 modes, of which there are around 1,000 in this case. The jumps between modes ensure that the posterior is very well described, with all of the modes mapped out equally while requiring relatively few function evaluations.

Before the chain began, the diffusion model was seeded by providing 50 points located near each of the modes. Given that the modes are positioned far from each other, it is necessary to do this to allow the diffusion model to propose jumps between modes at the start. These initial points are increasingly unlikely to be proposed as the chain progresses since there are many more samples being generated, and they are removed at the end of the chain.

4.2 10D Gaussian mixture

Here we compare our results using Algorithm 1 to results obtained from a normalizing flow MCMC algorithm for a high-dimensional Gaussian mixture function with 2 components. In the same way as in [20], we define a mixture of 2 Gaussians in 10D,

$$f(\theta_i) = w_A \frac{e^{-\frac{1}{2}|\theta_i - \theta_{A,i}|^2}}{(2\pi)^{d/2}} + w_B \frac{e^{-\frac{1}{2}|\theta_i - \theta_{B,i}|^2}}{(2\pi)^{d/2}}, \quad (4.2)$$

with weights $w_A = 2/3$, $w_B = 1/3$ and non-zero means only in the first two dimensions $\theta_{A1,2} = (8, 3)$, $\theta_{B1,2} = (-2, 3)$. We therefore have to sample two modes in the first two dimensions situated significantly far away from one another, with different relative weights. As shown in Fig. 2, Algorithm 1 converges to the truth within 5,000 samples, which is very few function evaluations given the high-dimensional nature of this problem. Once again, the diffusion model was initially seeded with 50 points near the two modes.

Importantly, our diffusion model MCMC sampler outperforms a similar normalizing flow MCMC sampler for this same problem by requiring fewer training iterations in order to reach a high acceptance rate. Figure 3 shows that Algorithm 1 is able to reach close to 80% acceptance within less than 100 training iterations, whereas the flow-based approach requires at least 500 iterations to reach a comparable acceptance rate [20].

We do however point out that neither the total number of samples nor the number of samples for each training iteration were specified for the normalizing flow algorithm. We further note that the full structure of the normalizing flow algorithm is significantly more complex than our Algorithm 1. A normalizing flow is an invertible map that takes a base density such as a Gaussian with unit variance, and pushes it forward towards a target density. In [20], neural networks are trained to learn the mapping between the base and

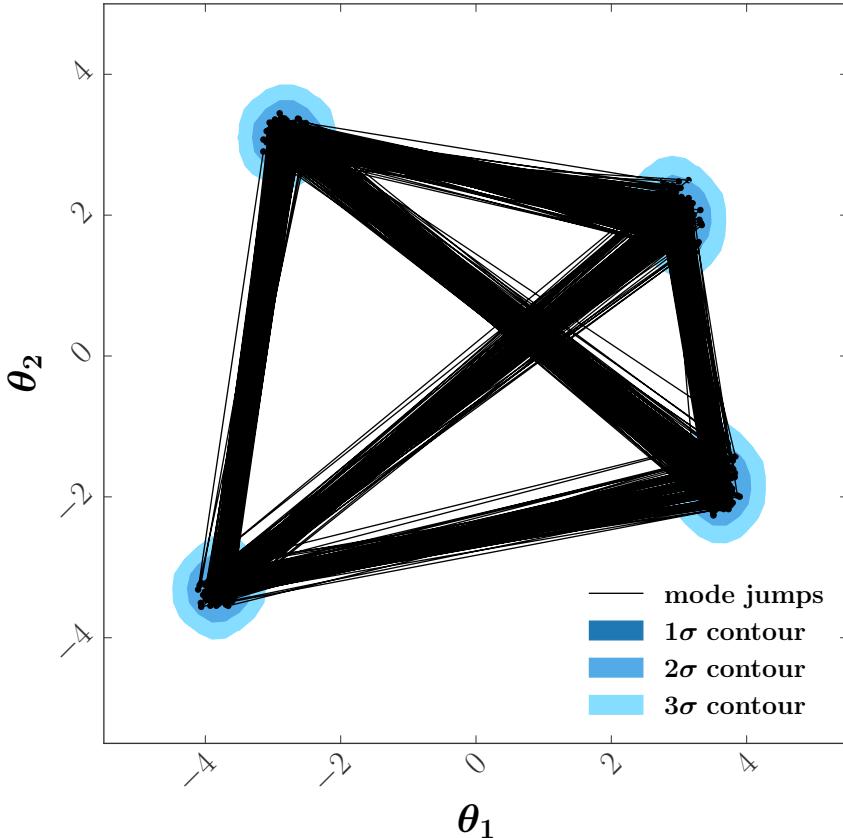


Figure 1: 2D Himmelblau function sampled with Algorithm 1 (10,000 samples). Black lines depict jumps between modes due to the diffusion model.

target densities by minimizing the Kullback-Leibler divergence. The normalizing flow is therefore supervised at the PDF level rather than the sample level, which may explain the increased performance on the side of the diffusion model.

Where we have paired our diffusion model with a basic local Gaussian transition kernel, the normalizing flow algorithm is combined with a more sophisticated local MALA sampler. Furthermore, the diffusion model algorithm operates in a single chain, whereas hundreds of independent walkers are initialized and split between each mode at the start of the normalizing flow algorithm. We therefore expect our results to be conservative in performance - a more sophisticated transition kernel and algorithmic structure would likely result in better performance in our case. The acceptance rate comparison is merely suggestive of an advantage the diffusion model may have over normalizing flows for this particular example, an exact comparison is considered beyond the scope of this paper.

Figure 3 also shows the property of asymptotic exactness shared by independent MH samplers such as these. As the MCMC chain progresses and the diffusion model/normalizing

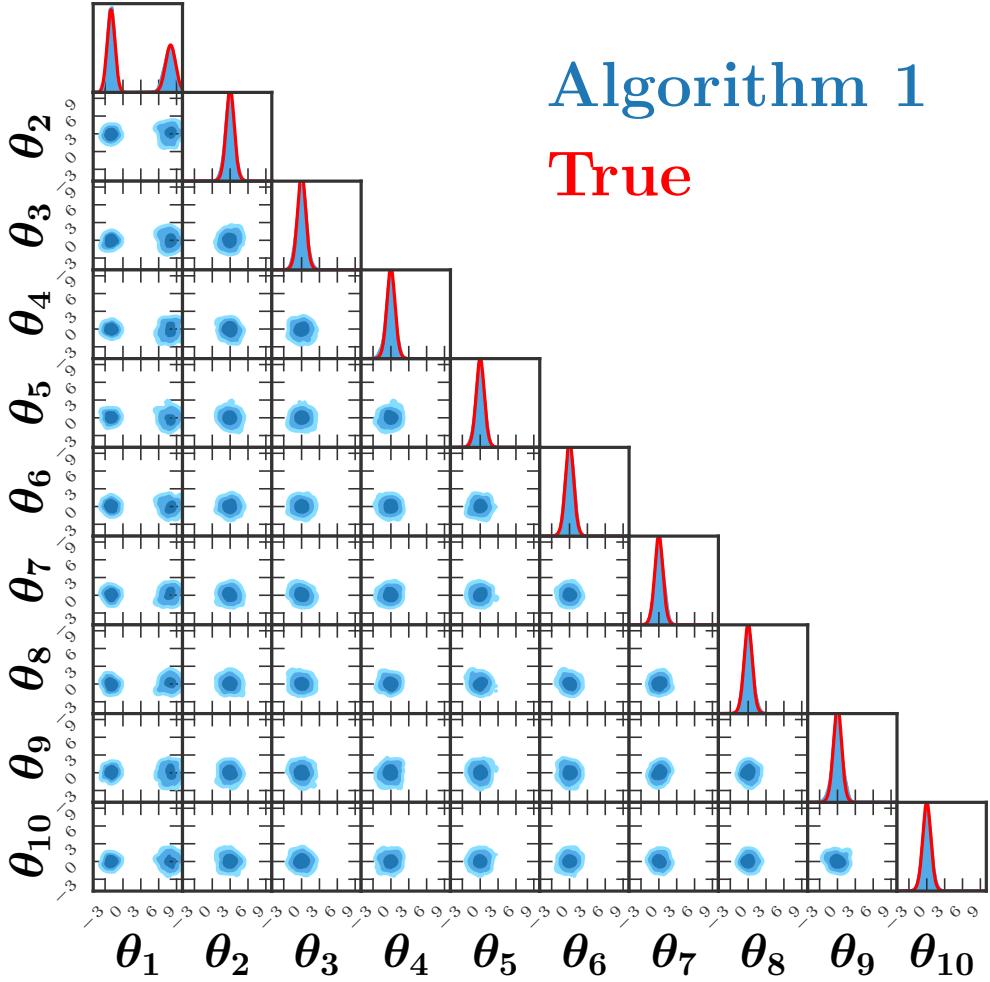


Figure 2: Corner plot of Algorithm 1 for a 10D Gaussian mixture with 2 components (5,000 samples). The true marginalized posteriors are shown in red, blue contours represent sample density.

flow is repeatedly retrained, the acceptance rate increases well beyond 70%. This indicates that the proposal diffusion model is closely approximating the target posterior, as explained in Sec. 3. It also highlights an important feature of algorithms of this kind - at the conclusion of the MCMC chain, one obtains a very fast approximation to the posterior from which one can draw as many samples as required.

4.3 4D EggBox function

The EggBox function is given by

$$f(\boldsymbol{\theta}) = \left(2 + \prod_{i=1}^D \left[\cos\left(\frac{\theta_i}{2}\right) \right] \right)^5, \quad (4.3)$$

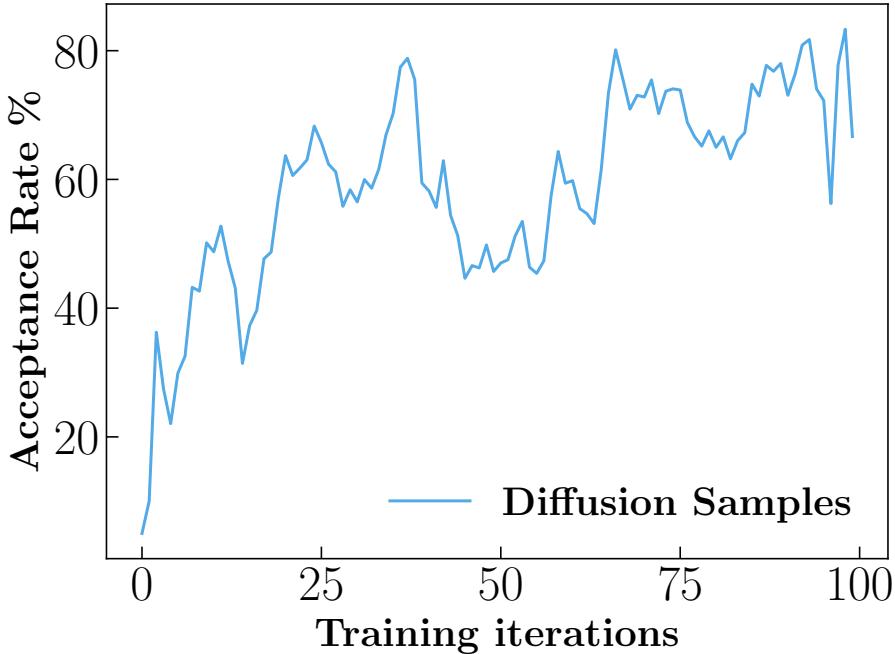


Figure 3: Acceptance rate with Algorithm 1 for a 10D Gaussian mixture with 2 components.

which consists of many local modes of equal height spread at regular intervals throughout parameter space. The challenge for an MCMC algorithm is to find all of the modes while weighting them equally. Given a wide enough Gaussian proposal, pure MH is eventually able to successfully find all the minima, however the inclusion of the diffusion model accelerates this process dramatically. Figure 4 shows a comparison between pure MH and Algorithm 1 for the heavily multi-modal 4D EggBox function over 0.1M samples. For such a low number of samples, pure MH is unable to map a substantial fraction of the total modes (it produced samples in around 55% of modes), particularly at the edges of the parameter space. On the other hand, Algorithm 1 is able to find most of the modes (samples produced in around 80% of modes), with the 1D marginal distributions showing roughly equal peaks across each parameter compared to pure MH. We found that pure MH would have to perform at least 5 times as many function evaluations to obtain similar samples as for Algorithm 1 in Fig. 4, demonstrating a significant acceleration of the MCMC sampling process in this case. In this case, the diffusion model was seeded by providing 1,000 random points drawn from a uniform distribution.

It is also worth comparing our diffusion-accelerated MCMC algorithm to an algorithm that makes use of multiple chains, such as the widely-used `emcee` algorithm [32]. After running the default `emcee` algorithm on the EggBox function with 8 chains for a total of 0.1M samples, we created a comparison plot with Algorithm 1, see Fig. 5. We found that there was only a slight improvement over the single chain of pure MH, with around 60% of modes found. This demonstrates that Algorithm 1 can outperform algorithms with multiple

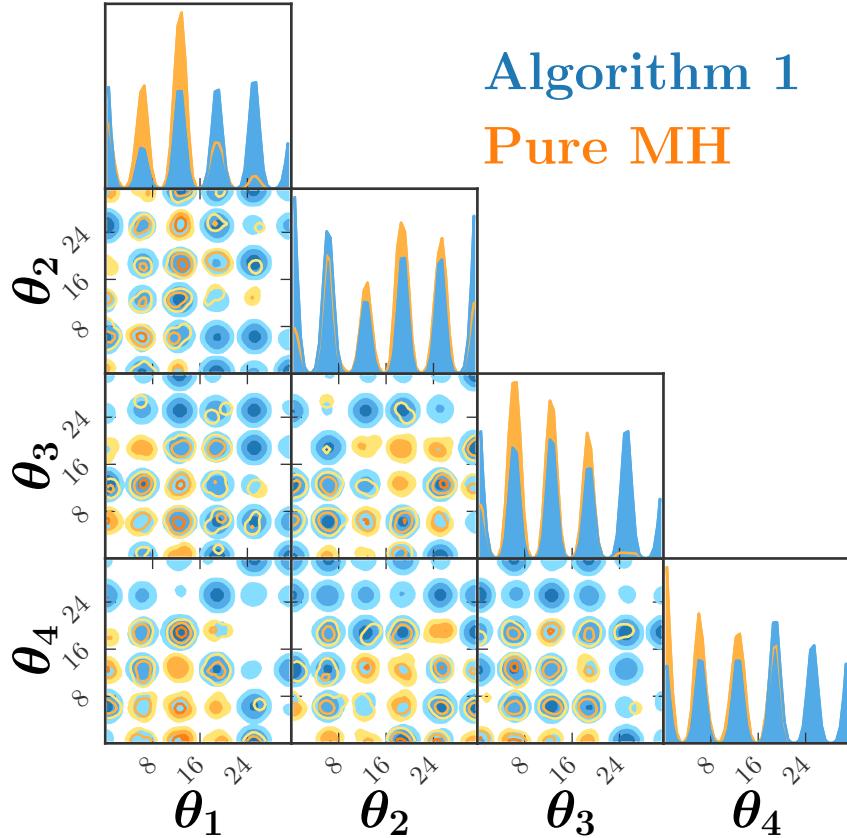


Figure 4: Comparison of Algorithm 1 and pure MH for the 4D EggBox function (0.1M samples). Coloured contours represent sample density for each algorithm, with more identified modes representing greater success at characterising the shape of the true posterior.

chains for heavily multi-modal functions.

5 Physics example: global parton distribution function fit

We now apply Algorithm 1 and pure MH to a simple physics example intended to reflect the salient features of a global PDF fit. PDFs describe the probability of finding a particular parton (quark or gluon) in a hadron as a function of the energy-momentum fraction x carried by the parton and the energy scale Q for a given collision. By parametrizing the PDFs, we can make theoretical predictions for physical observables (typically cross sections) and compare to experimental data. A global PDF fit then involves constructing a goodness of fit measure, such as the χ^2 , that describes the difference between theory and experiment, from which the likelihood can be defined as $\mathcal{L} = \exp(-\frac{1}{2}\chi^2)$. This likelihood is highly computationally expensive to evaluate for real data, and global PDF analyses often use frequentist statistics to construct approximate uncertainty bands on the PDFs through repeated maximization of the likelihood. However, in principle a purely Bayesian approach, such as a Metropolis-Hastings sampler, would also provide valid uncertainty

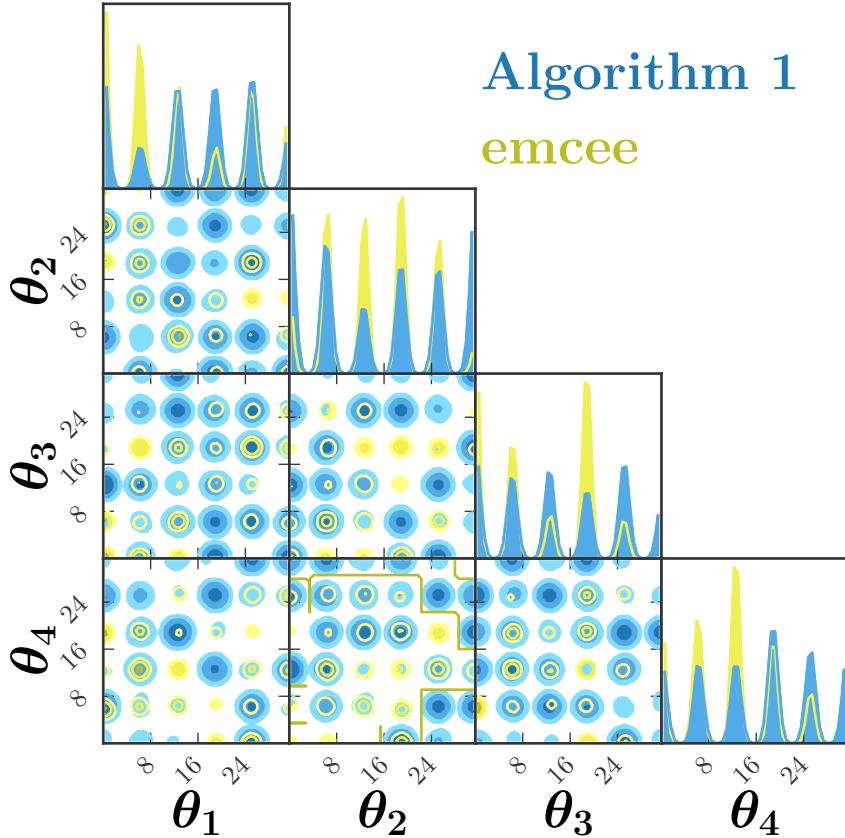


Figure 5: Comparison of Algorithm 1 and the emcee algorithm for the 4D EggBox function (0.1M samples). Coloured contours represent sample density for each algorithm, with more identified modes representing greater success at characterising the shape of the true posterior.

bands for this likelihood given enough time. For our simple PDF example, we will use a likelihood function that is much faster to evaluate in order to compare the uncertainty bands obtained via Algorithm 1 and pure MH.

The details of the construction of this simple PDF example can be found in Ref. [33]; here, we summarize the most important elements. We first construct a set of two toy “quark” PDFs with x dependence parametrized by a basic functional form,

$$q_i(x) = N_i x^{\alpha_i} (1-x)^{\beta_i}, \quad i = 1, 2. \quad (5.1)$$

These functions could, for example, represent the u and d quark PDFs in the proton, with free parameters α_i and β_i . We can then model toy “cross sections” σ_j using a linear combination of the two PDFs,

$$\sigma_j = \sum_{i=1,2} c_{ji} q_i. \quad (5.2)$$

These could, for instance, correspond to inclusive deep-inelastic scattering (DIS) cross sections for a proton and a neutron, with the coefficients c_{ij} proportional to the squares of

the quark charges, in which case we would assign $c_{11} = 4c_{12} = 4c_{21} = c_{22}$. The values of x are taken in the range $x = 0.1 - 0.9$ at regular intervals, depending on the number of toy data points we wish to generate. For each x value we calculate the corresponding q_1 and q_2 PDFs, from which the true σ_1 and σ_2 are determined. Toy cross section data points are then generated by drawing randomly from a Gaussian distribution centred on the true cross section values, and uncertainties are assigned to be 0.1 times the magnitude of each toy data point.

To fit this pseudodata, we define a 4D model following the traditional $\sim x^\alpha(1-x)^\beta$ parametrization, which matches the true underlying law in Eq. (5.1). We can then define

$$\chi^2 = \sum_{i=1,2} \sum_j^{n_{\text{data}}} \left(\frac{\sigma_i^{\text{data}}(x_j) - \sigma_i^{\text{model}}(x_j, \boldsymbol{\theta})}{\Delta\sigma_i(x_j)} \right)^2, \quad (5.3)$$

where i is the index of the two cross sections, j is the index of the data points summing to a total number of points n_{data} in the sample, σ_i^{data} is the toy cross section data with uncertainty $\Delta\sigma_i$, and σ_i^{model} is the model cross section [Eq. (5.2)] for a given set of parameters $\boldsymbol{\theta}$.

The likelihood is then given by

$$\mathcal{L} = \exp\left(-\frac{1}{2}\chi^2\right). \quad (5.4)$$

Using this likelihood, we can now sample the posterior using Algorithm 1 and pure MH, and generate uncertainty bands for the cross sections and PDFs. The likelihood in this case has a single mode.

Figure 6 shows a comparison between Algorithm 1 and pure MH for our 4D PDF physics example. We also include the uncertainty bands generated in the limit of many samples, as an estimate of the true uncertainties. We found that both Algorithm 1 and pure MH agree in this limit, meaning that our diffusion-augmented algorithm does converge to the purely Markovian distribution for this example. In general, one ought to be careful when using past samples taken from the existing chain as inputs to generate new samples as we do in Algorithm 1. For example, adaptive MCMC algorithms require certain conditions to be met in order to ensure ergodicity of the chain, see [34] for further details.

For the 10,000 samples shown here, pure MH heavily overestimates the uncertainty bands. On the other hand, pairing MH with the diffusion model results in samples that approximate the true uncertainty bands very closely. 3 times as many samples would be required for pure MH to produce a fit of comparable quality. One advantage the diffusion model has over pure MH is that it can make nonlocal jumps to key regions of interest and avoid getting stuck in the tails of the distribution, which explains why Algorithm 1 is able to converge to the truth in fewer likelihood evaluations even for a function with a single mode. The diffusion model was initially seeded with 100 points located near the mode.

6 Conclusions

We have developed a diffusion model tailored for low-dimensional data, pairing it with a Metropolis-Hastings sampler to create a new accelerated Markov Chain Monte Carlo algorithm. We find that Algorithm 1 is many times faster than pure Metropolis-Hastings

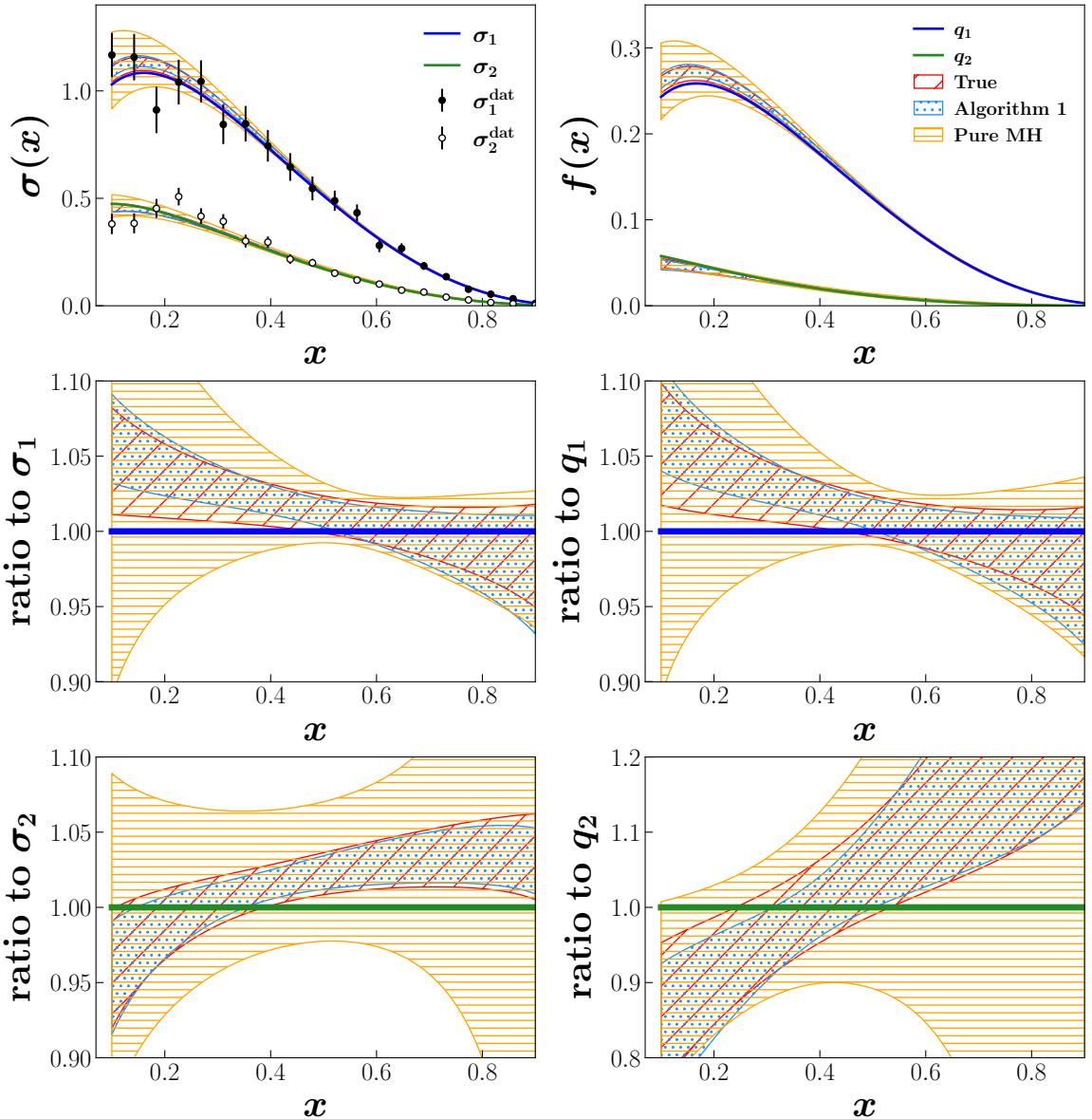


Figure 6: Comparison of fit results for Algorithm 1 and pure MH (10,000 samples) to the many-sample limit (1M samples). The top row contains cross section and parton distributions, while the middle and bottom rows show ratios to the true values for each distribution.

at obtaining accurate posterior samples for test functions with any number of modes, as evidenced by a significant reduction of the number of likelihood evaluations required to adequately explore the Himmelblau, 10D Gaussian mixture, and EggBox functions. Algorithm 1 has also shown significant improvement over a normalizing flow algorithm for a specific test function. In the context of a real physical example based on a global analysis of parton distribution functions, Algorithm 1 is able to characterize uncertainty bands much more accurately than pure Metropolis-Hastings for a limited number of samples.

Our technique does not require gradient information, and is therefore applicable to a wide range of posteriors encountered in physics applications. Furthermore, our proof of principle indicates that a combination of a diffusion model with a more sophisticated MCMC or other sampler could be expected to yield even better results.

Acknowledgements

This research was supported by the ARC Centre of Excellence CE200100008 and by the DOE with contract No. DE-AC05-06OR23177, under which Jefferson Science Associates, LLC operates Jefferson Lab. The work of N.S. was supported by the DOE, Office of Science, Office of Nuclear Physics in the Early Career Program. We wish to thank Andrew Fowlie for multiple useful insights regarding Gibbs sampling and MCMC algorithms.

A Hyperparameters

Here we outline each of the hyperparameters in Algorithm 1. If their values varied between test problems, the different settings are contained in Table 1.

- The total number of retrains was set to 100 for each test problem.
- τ is the number of samples per retrain of the diffusion model. τ multiplied by (retrains) gives n , the total number of samples generated.
- p_{diff} is the probability of drawing a diffusion proposal sample rather than a Gaussian proposal sample.
- σ_{MH} is the width of the Gaussian proposal distribution.
- β is the amount of noise progressively added in the forward diffusion process, and is set to increase linearly from 0.1 to 0.3 for each test problem.
- `nsteps` is the number of noising steps in the forward/reverse diffusion process, and is set to 20 for each of the test problems.
- `bins` is the number of bins in the 1D histograms when calculating Eq. (3.2), and is set to 20 for each test problem.

Table 1: Algorithm 1 hyperparameter values assigned for each test problem.

Hyperparameter	2D Himmelblau	10D Gaussian	4D EggBox	4D PDF
τ	100	500	1000	100
p_{diff}	0.83	0.1	0.5	0.5
σ_{MH}	0.15	0.5	0.6	0.1

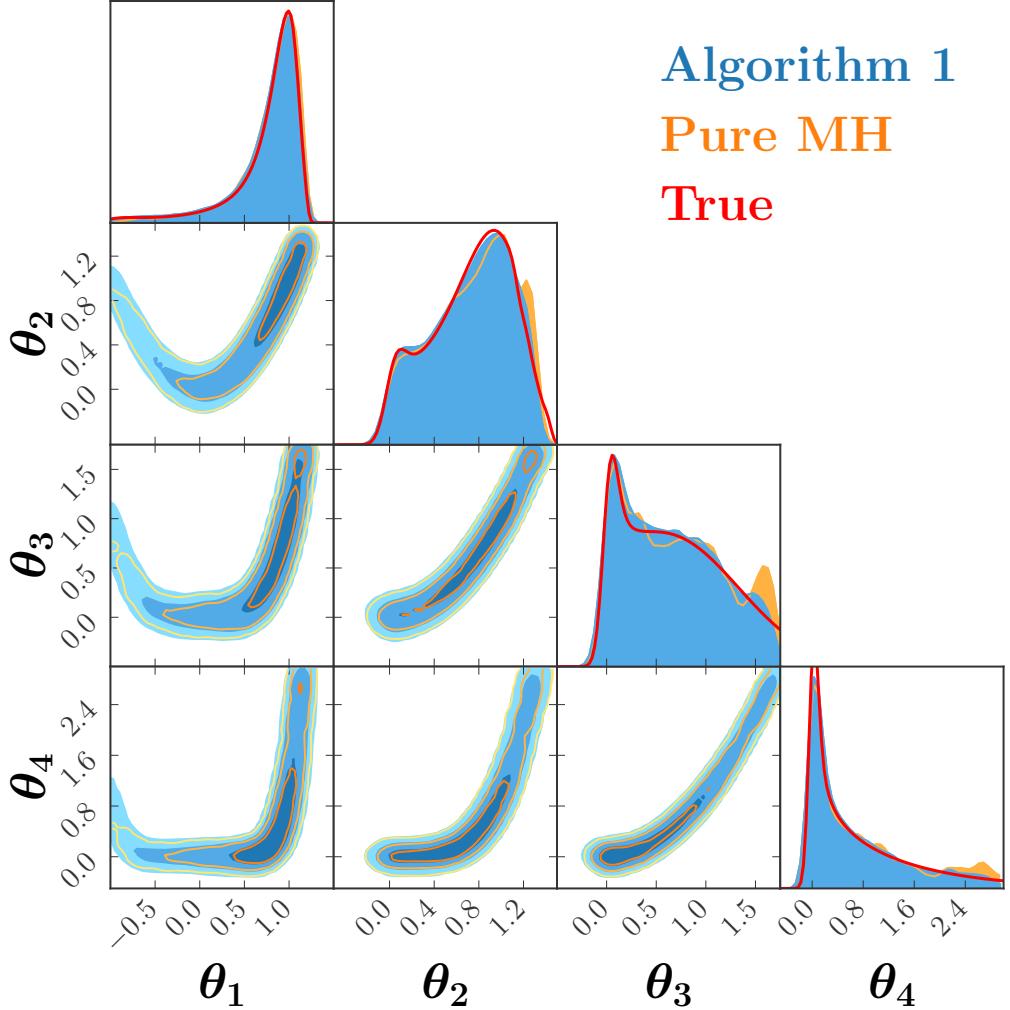


Figure 7: Comparison of Algorithm 1 and pure MH for the 4D Rosenbrock function (0.5M samples). The true marginalized posteriors are shown in red, coloured contours represent sample density.

B 4D Rosenbrock function

The Rosenbrock function is given by

$$f(\boldsymbol{\theta}) = \sum_{i=1}^{D-1} [100 \times (\theta_{i+1} - \theta_i)^2 + (1 - \theta_i)^2], \quad (\text{B.1})$$

where D is the number of dimensions. This banana-shaped function consists of a single mode located within a long, flat bump. In 4D, it is particularly difficult to find this mode while also correctly mapping the curved shape of the surrounding area. Figure 7 shows a comparison between pure MH and Algorithm 1 for the 4D Rosenbrock function over 0.5M MCMC samples. Pure MH produces a false secondary mode, which is not present once the diffusion model is added in. Twice as many samples were required for pure MH to match

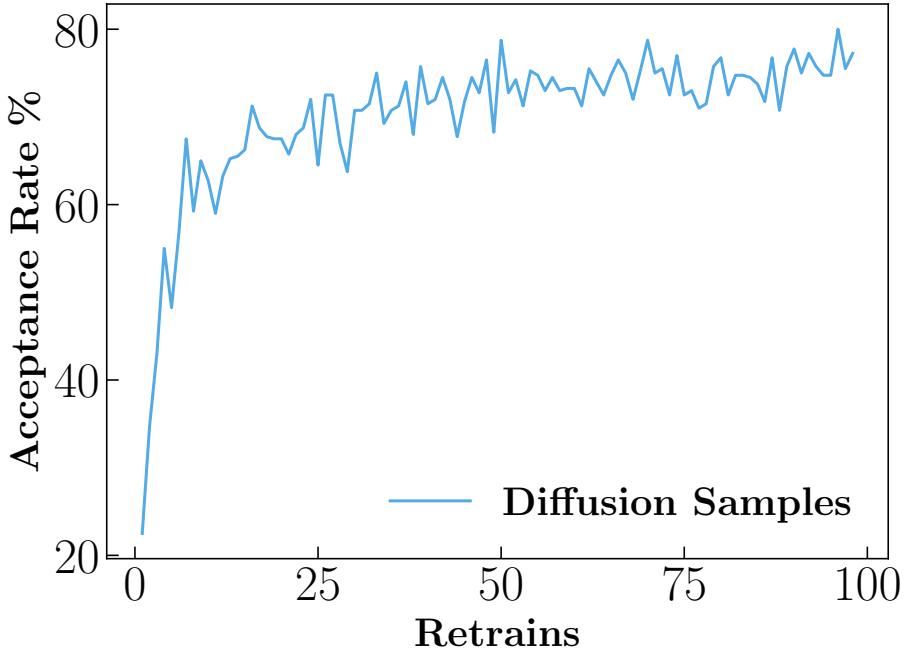


Figure 8: Increase in diffusion sample acceptance rate with number of retrains for the 4D Rosenbrock function (0.5M samples).

Algorithm 1 for this test function. One advantage the diffusion model has over pure MH is that it can make nonlocal jumps to key regions of interest and avoid getting stuck in the tails of the distribution, which explains why Algorithm 1 is able to converge to the truth in fewer function evaluations.

We also show in Fig. 8 a plot demonstrating asymptotic exactness for the Rosenbrock test function using Algorithm 1. As the MCMC chain progresses and the diffusion model is repeatedly retrained, the acceptance rate increases well beyond 70%. This indicates that the proposal diffusion model is closely approximating the target posterior, as explained in Sec. 3.

References

- [1] DARKMACHINES HIGH DIMENSIONAL SAMPLING GROUP collaboration, *A comparison of optimisation algorithms for high-dimensional particle and astrophysics applications*, *JHEP* **05** (2021) 108 [[2101.04525](#)].
- [2] A. Kvellestad, P. Scott and M. White, *GAMBIT and its Application in the Search for Physics Beyond the Standard Model*, [1912.04079](#).
- [3] G.M. Martin, D.T. Frazier and C.P. Robert, *Computing bayes: Bayesian computation from 1763 to the 21st century*, 2020. [DOI](#).
- [4] J. Liu, *Monte Carlo strategies in scientific computing*, Springer Verlag, New York, Berlin, Heidelberg (2008).

- [5] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller, *Equation of state calculations by fast computing machines*, *The Journal of Chemical Physics* **21** (1953) 1087.
- [6] W.K. Hastings, *Monte carlo sampling methods using markov chains and their applications*, *Biometrika* **57** (1970) 97
[\[http://biomet.oxfordjournals.org/cgi/reprint/57/1/97.pdf\]](http://biomet.oxfordjournals.org/cgi/reprint/57/1/97.pdf).
- [7] D.J.C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Copyright Cambridge University Press (2003).
- [8] J. Sohl-Dickstein, E.A. Weiss, N. Maheswaranathan and S. Ganguli, *Deep unsupervised learning using nonequilibrium thermodynamics*, *CoRR* **abs/1503.03585** (2015) [1503.03585].
- [9] Y. Song and S. Ermon, *Generative modeling by estimating gradients of the data distribution*, 2019. [DOI](#).
- [10] J. Ho, A. Jain and P. Abbeel, *Denoising diffusion probabilistic models*, 2020. [DOI](#).
- [11] Y. Song and S. Ermon, *Improved techniques for training score-based generative models*, 2020. [DOI](#).
- [12] A. Nichol and P. Dhariwal, *Improved denoising diffusion probabilistic models*, *CoRR* **abs/2102.09672** (2021) [2102.09672].
- [13] P. Dhariwal and A. Nichol, *Diffusion models beat gans on image synthesis*, 2021. [DOI](#).
- [14] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew et al., *Glide: Towards photorealistic image generation and editing with text-guided diffusion models*, 2021. [DOI](#).
- [15] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu and M. Chen, *Hierarchical text-conditional image generation with clip latents*, 2022. [DOI](#).
- [16] A. Montanari and Y. Wu, *Posterior sampling from the spiked models via diffusion processes*, 2023. [2304.11449].
- [17] M.S. Albergo, G. Kanwar and P.E. Shanahan, *Flow-based generative models for Markov chain Monte Carlo in lattice field theory*, *Phys. Rev. D* **100** (2019) 034515 [1904.12072].
- [18] D. Boyda, G. Kanwar, S. Racanière, D.J. Rezende, M.S. Albergo, K. Cranmer et al., *Sampling using $SU(N)$ gauge equivariant flows*, *Phys. Rev. D* **103** (2021) 074504 [2008.05456].
- [19] M.S. Albergo, D. Boyda, D.C. Hackett, G. Kanwar, K. Cranmer, S. Racanière et al., *Introduction to Normalizing Flows for Lattice Field Theory*, 2101.08176.
- [20] M. Gabrié, G.M. Rotskoff and E. Vanden-Eijnden, *Efficient bayesian sampling using normalizing flows to assist markov chain monte carlo methods*, 2021. [DOI](#).
- [21] M. Gabrié, G.M. Rotskoff and E. Vanden-Eijnden, *Adaptive Monte Carlo augmented with normalizing flows*, *Proc. Nat. Acad. Sci.* **119** (2022) e2109420119 [2105.12603].
- [22] Q. Zhang and Y. Chen, *Diffusion normalizing flow*, 2021. [2110.07579].
- [23] C. Balázs et al., *Cosmological constraints on decaying axion-like particles: a global analysis*, *JCAP* **12** (2022) 027 [2205.13549].
- [24] GAMBIT collaboration, *Collider constraints on electroweakinos in the presence of a light gravitino*, *Eur. Phys. J. C* **83** (2023) 493 [2303.09082].

- [25] C. Chang, P. Scott, T.E. Gonzalo, F. Kahlhoefer and M. White, *Global fits of simplified models for dark matter with GAMBIT II. Vector dark matter with an s-channel vector mediator*, [2303.08351](#).
- [26] R. Cornish, A.L. Caterini, G. Deligiannidis and A. Doucet, *Relaxing bijectivity constraints with continuously indexed normalising flows*, 2021. [[1909.13833](#)].
- [27] W. Feller, *On the theory of stochastic processes, with particular reference to applications*, 1949.
- [28] B.C. Allanach and C.G. Lester, *Sampling using a ‘bank’ of clues*, *Comput. Phys. Commun.* **179** (2008) 256 [[0705.0486](#)].
- [29] L. Tierney, *Markov Chains for Exploring Posterior Distributions*, *The Annals of Statistics* **22** (1994) 1701 .
- [30] D. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press (2003).
- [31] S. Samsonov, E. Lagutin, M. Gabré, A. Durmus, A. Naumov and E. Moulines, *Local-global mcmc kernels: the best of both worlds*, in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh, eds., vol. 35, pp. 5178–5193, Curran Associates, Inc., 2022,
https://proceedings.neurips.cc/paper_files/paper/2022/file/21c86d5b10cdc28664ccdadf0a29065a-Paper-Conference.pdf.
- [32] D. Foreman-Mackey, D.W. Hogg, D. Lang and J. Goodman, *emcee: The MCMC hammer*, *Publications of the Astronomical Society of the Pacific* **125** (2013) 306.
- [33] N.T. Hunt-Smith, A. Accardi, W. Melnitchouk, N. Sato, A.W. Thomas and M.J. White, *Determination of uncertainties in parton densities*, *Phys. Rev. D* **106** (2022) 036003 [[2206.10782](#)].
- [34] C. Andrieu and J. Thoms, *A tutorial on adaptive mcmc*, *Statistics and Computing* **18**, issue 4 (2008) 343 .