NESTED SLICE SAMPLING

David Yallup*, **Will Handley**
Institute of Astronomy and Kavli Institute for Cosmology Cambridge
University of Cambridge
{dy297, wh260}@cam.ac.uk

Namu Kroupa
Department of Engineering and Cavendish Laboratory
University of Cambridge
nk544@cam.ac.uk

ABSTRACT

Nested Sampling is a powerful meta Bayesian inference algorithm known for its ability to estimate the marginal likelihood of a model and perform parameter inference, even for complex multimodal distributions. In this paper, we refine the formulation of Nested Sampling in the context of slice sampling, leading to a novel vectorized version of the algorithm that leverages GPU acceleration for improved efficiency in machine learning applications. We demonstrate that this vectorized Nested Slice Sampling algorithm can exploit parallelization opportunities to substantially reduce runtime while maintaining sampling accuracy. The performance of the approach is evaluated on a range of challenging benchmark problems, showing significant improvements in sampling efficiency and scalability to high dimensions. The proposed vectorized Nested Sampling algorithm opens up new possibilities for applying Nested Sampling to large-scale machine learning problems where efficient Bayesian inference is critical. We provide an open-source implementation of our method to facilitate adoption and reproducibility.

1 INTRODUCTION

Sampling from unnormalized probability distributions is a foundational task in Machine Learning. This probability distribution can be defined,

$$P(x) = \frac{\exp\left(-\beta E(x)\right)\Pi(x)}{Z}, \quad \beta \propto 1/T, \tag{1}$$

Where x are the parameters of interest, E is the scalar energy function and Π some reference probability density. The normalizing constant Z is the partition function of the Boltzmann distribution, and is typically described as the intractable normalizing constant in the machine learning literature. The inverse temperature T, defined as β, is introduced that can smoothly modulate between some reference distribution Π (often implicitly taken as an unbound uniform probability density and omitted) at $\beta = 0$ and the target density at $\beta = 1$. This constant is defined as the integral of the Boltzmann factor $\exp(-\beta E(x))$ over the microstates x of the system,

$$Z = \int \exp\left(-\beta E(x)\right)\Pi(x)dx\,. \tag{2}$$

When the Boltzmann factor is identified as a likelihood function – the probability of observing some data given parameters of a model – and the reference distribution is identified as a prior distribution of model parameters, the normalizing constant can be identified as the marginal likelihood in statistical inference (Murphy, 2022). Numerical estimation of this quantity is a central task when performing Bayesian Model comparison (MacKay, 2002). This numerical estimation is widely performed with Markov Chain Monte Carlo (MCMC) (Geyer, 1992) methods, although most standard MCMC

*Corresponding author

algorithms do not directly estimate Z. Advances in such algorithms is of broad interest across Machine Learning (ML) driven fields.

Nested Sampling was first introduced by Skilling as a generic meta algorithm targeting the estimation of the marginal likelihood (Skilling, 2006). At a high level it belongs to the same school as particle Monte Carlo methods, namely Sequential Monte Carlo (Doucet et al., 2001), evolving a population of particles through a series of intermediate distributions that bridge between the reference uninformed state and the informed target. It saw immediate adoption in the physical sciences (Ashton et al., 2022; Pártay et al., 2014), where it was noted for its robustness in the face of sampling from complex, multimodal distributions. However, despite this popularity, a single, canonical implementation strategy has not emerged, leading to a variety of named implementations with differing internal mechanics. This presents a methodological gap compared to more standardized algorithms like Hamiltonian Monte Carlo. Various aspects of its specific implementation have been extensively explored in the intervening years, with some common successful patterns emerging which we summarize in this work. Contemporary work on incorporating gradients of the energy surface has seen some promising results (Feroz & Skilling, 2013; Cai et al., 2022; Lemos et al., 2023), and some implementations in modern autodiff (Baydin et al., 2018) compatible frameworks (Albert, 2020; Anau Montel et al., 2024) have appeared in recent years.

In this work, building on successful patterns in NS development, we look to address this methodological gap and provide a clear, efficient, and modern implementation suitable for both physical science and ML practitioners. We firstly seek to refine the formalism of Nested Sampling in Section 2, focussing on slice sampling as a robust and efficient inner kernel, arguing for its foundational role in performant NS implementations. We distill this insight into a practical implementation, Nested Slice Sampling (NSS), detailed in Section 3, that incorporates several key advantages over existing approaches (discussed in Section D), most notably enabling efficient vectorization for parallel hardware (GPUs), analogous to recent efforts vectorizing other MCMC methods (Hoffman & Sountsov, 2022; Hoffman et al., 2021). Our implementation is designed to be composable within modern probabilistic programming ecosystems like `jax`. Subsequently, in Section 4, we highlight the state-of-the-art performance achievable with this implementation on challenging black-box sampling problems, demonstrating its ability to handle multimodality and estimate normalizing constants accurately across a range of dimensions.

2 Theoretical Framework and Related Methodologies

In this section we review the generic framework of Nested Sampling, and it's connections to other sampling methodologies. We also introduce the Slice Sampling algorithm, and demonstrate how it uniquely suited as an inner kernel for Nested Sampling.

2.1 Nested Sampling

An illustration of the NS algorithm is shown in the top panel of Figure 1, where the reference Gaussian density in orange is weighted by a Boltzmann factor illustrated in solid blue, with this sketch of the algorithm running from left to right. The target log density we wish to sample from is shown in the bottom panel in black, and exhibits two distinct modes of density. A shortcoming of many MCMC single chains is that probabilities to transition between these modes rapidly approaches zero when realized in higher dimensions (Łatuszyński et al., 2025). A common theme of algorithm design to alleviate this is to construct a path of interpolating distributions between the known reference and the desired target, where *annealing* – increasing β smoothly from zero to one – is perhaps the dominant paradigm. NS constructs these interpolating targets by iterative compression of the reference density, using a decreasing energy criterion (dashed blue) to drive particles towards the low energy (high likelihood) regions of the reference density (Skilling, 2006). This path of intermediate distributions (shown in solid shades of orange) is unique to NS, formed of constrained regions of the reference density.

We form a generic *outer kernel* that defines the transition between a population of particles from one energy level to the next in Algorithm 1 (defined in Appendix B). Leaving aside the issue of the constrained propagation step for now, we review how this outer kernel yields a set of samples that can reconstruct the partition function. Repeated action of this kernel yields a series of monotonically

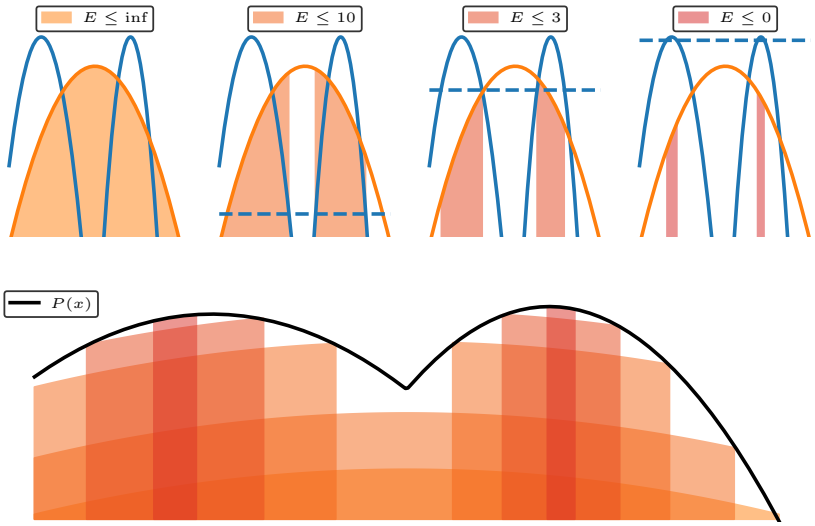

Figure 1: Illustration of the Nested Sampling algorithm [top], and how it constructs the partition function [bottom].

decreasing ordinates in energy[1],

$$E_1 > E_2 > \ldots > E_n \,, \tag{3}$$

and k deleted (*dead*) particles that estimate the fractional decrease in volume (X) by a geometric factor dependent on the size of the (*live*) population, m, as,

$$\Delta \ln X_i = -k/m \,. \tag{4}$$

Starting from a known reference volume of $X_0 = 1$, the partition function can be constructed as a sum weighted by the Boltzmann factor associated with each energy ordinate as,

$$Z = \sum_{i=1}^{n} \exp(-\beta E_i) \Delta X_i \,. \tag{5}$$

Noting that this possesses the unique property of being *athermal* i.e. amenable to reweighting to any temperature level. This estimation follows geometric arguments on the expectation of the volume contraction, and the accumulated effect this has on the partition function has been well studied in the literature (Chopin & Robert, 2010; Keeton, 2011; Fowlie et al., 2023). The energy levels being set automatically renders the algorithm inherently *adaptive*, and it is typical to provide some *termination criteria* exiting once the remaining *live* term in the sum is sufficiently smaller than the accumulated estimate. We revisit some finer details of these arguments in Section C.

The base form of NS comprises of a single particle $k = 1$ transition, but this can be set to some higher fraction of the total particle count m allowing vectorized versions we will exploit in this article. This parallelization strategy, detailed further in Section D.3, is key to leveraging modern hardware. This scheme has already been exploited to some level with MPI parallelization for CPU usage (Handley et al., 2015). The resulting estimate in Equation (5) is a Lebesgue integral of *vertical* division of the energy levels (Llorente et al., 2023). Where the choice of vertical steps is automatically determined by the algorithm and optimally set for the target of marginal likelihood estimation (Polson & Scott, 2015).

2.2 Slice Sampling as Nested Sampling

Efficiently sampling from the constrained reference distribution is the core challenge of Nested Sampling (NS). At each iteration, NS requires drawing samples from the reference distribution $\Pi(x)$ re-

[1]In probabilistic inference terms, the energy is the negative-log likelihood, hence decreasing energy is increasing likelihood.

stricted to the region where the energy is less than a certain threshold $E_{\min}$ (i.e., $E(x) < E_{\min}$) (Buchner, 2021a). We propose two arguments in this work, firstly that it *requires a minimal augmentation to Slice Sampling (Neal, 2003) to realize NS*, and secondly that *all efficient implementations of NS inherently resemble some form of Slice Sampling*.

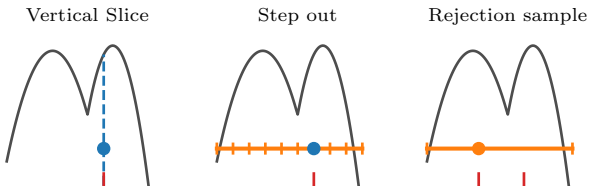

Figure 2: Illustration of the Slice Sampling procedure. First a vertical slice of the target density is taken, followed by an iterative stepping out procedure to set the interval of the horizontal slice, finally followed by a rejection sampling from this segment.

In moderate-dimensional spaces, rejection sampling can be employed, utilizing the current set of live particles to inform a proposal distribution (Feroz & Hobson, 2008; Buchner, 2021b). However, rejection sampling becomes inefficient in high-dimensional spaces due to the curse of dimensionality. Proposal regions for rejection can be made increasingly expressive – using the *live* populations of particles to inform this choice – with a variety of ML approaches investigated (Lange, 2023; Williams et al., 2021; Torrado et al., 2023).

To address this challenge, successful implementations for high-dimensional problems often incorporate a Slice Sampling (SS) inner kernel within NS (Handley et al., 2015; Speagle, 2020; Albert, 2020). SS is an MCMC method designed to sample from a target distribution that is *in principle* free of tuning (Neal, 2003). SS constructs a Markov transition kernel that alternates between *vertical* slices of a target distribution by some random uniform auxiliary variable $u \sim \mathcal{U}(0, 1)$, allowing a new state x' to be sampled from the *horizontal* slice of the target at height $y = u \times \Pi(x)$. Specifically, we consider the *Hit-and-Run Slice Sampling* algorithm, which simulates from the horizontal slice by an iterative stepping out and rejection sampling procedure to efficiently sample from high-dimensional constrained distributions (Neal, 2003). A figurative diagram of this procedure is shown in Figure 2. This realization of SS has favorable scaling properties and practical implementation advantages in high-dimensional settings (Power et al., 2024; Rudolf & Ullrich, 2018). The hit-and-run (Smith, 1984) portion of the procedure is powered by constructing some proposal distribution, and using this to generate new directions to perform the horizontal slice step (Buchner, 2023). We consider a simple Gaussian direction proposal distribution in practice, characterized by some conditioning matrix, M, used to sample Mahalanobis normalized directions for slicing. There is much potential for improvement in building more complex proposals (Moss, 2020), but a fast, robust baseline will prove highly competitive.

In Algorithm 2 (Appendix B) we detail SS as an inner kernel complement to the NS outer kernel detailed in Algorithm 1. We observe that if the auxiliary energy variable constraint is removed (i.e., $E_{\min} \to \infty$), and the target is the unnormalized posterior density, we recover standard SS. Where the auxiliary energy level criteria does appear, it appears in conjunction with the existing stepping out procedure – vitally SS is already simulating from a region with compact (and potentially complex) support. By maintaining a sample of *live* particles defining the auxiliary energy level $E_{\min}$, SS can be amenably tuned throughout the runtime of the algorithm. NS benefits immensely from sampling methods adept at handling *compact support* – regions constrained by hard boundaries – contrary to many MCMC methods that perform best with unbounded support (a comparison demonstrating the inefficiency of constrained random walks is shown in Section F). While there are some notable exceptions (Murray et al., 2005; Brewer & Foreman-Mackey, 2016), Hit and Run SS has been the dominant paradigm for NS for a reason. Potential improvements to this core design invariably have many similarities to SS (Lee & Vempala, 2016; Chen et al., 2019).

2.3 Connections to Sequential Monte Carlo

The connection between Sequential Monte Carlo (SMC) (Chopin, 2002; Del Moral et al., 2006) and Nested sampling has been formalized in (Salomone et al., 2024), where NS can be styled as a particular

form of SMC. SMC is widely used in the statistical community for model comparison (Zhou et al., 2015), is impressively parallelizable (Murray et al., 2015) and is noted for robustness in multimodal sampling problems (Paulin et al., 2018). These are all features we highlight of NS in this work, hence it forms a natural choice for a baseline in our empirical studies. Similarly to Nested Sampling, sequential Monte Carlo is a meta algorithm admitting many realizations in practice. We consider in this an SMC implementation that is *tempered* – sequentially stepping in temperature between reference and target – and *adaptive* – choosing these temperature steps on the fly (Fearnhead & Taylor, 2010). A generic outer kernel of SMC is given in Algorithm 3 (Appendix B), where the core components are propagation, weighting and resampling steps. The tempered SMC kernel takes a monotonically increasing sequence of (inverse) temperatures $\{\beta_t\}_{t=0}^{t=T}$ that smoothly bridge from the known, sampleable, distribution at $\beta_0 = 0$ to the target at $\beta_T = 1$. The sequence of temperatures could be specified in advance, however a popular adaptive scheme is to calculate the effective sample size, ESS=$(\sum w_i^2)/(\sum w_i)^2$, allowing the choice of β^{N+1} to be set by solving for a desired ESS to be maintained in each outer step.

The choice of inner kernel to perform the propagation is an open design choice, with Metropolis Hastings Random Walks (Robert & Casella, 2005; Andrieu et al., 2010; Chopin, 2002) or Hamiltonian Monte Carlo (Duane et al., 1987; Buchholz et al., 2020) being popular choices. There are an array of other methods employed in the statistics literature to estimate normalizing constants (Llorente et al., 2023), however SMC's state-of-the-art performance on many problems and notable commonalities to NS make it a logical baseline. We note that where comparison between NS and SMC has been made in the past (Grosse et al., 2015; Salomone et al., 2024), a suboptimal choice of MH inner kernels for NS has been used (we explore this empirically in Section F). While SMC admits a highly flexible choice of inner kernel, our argument in Section 2.2 is that performant NS implementations are often best served by kernels adept at handling hard constraints, such as slice sampling.

3 Algorithm and implementation

The algorithm has been implemented in the `blackjax` framework (Cabezas et al., 2024), which is a library for sampling algorithms in the `jax` ecosystem (Bradbury et al., 2018). This choice is well motivated by the `jax` ecosystems rapid adoption in the scientific community, where the goal of composable and differentiable functions allows complex forward models used in scientific applications to be greatly accelerated. Additionally, the presence of state of the art, well tested SMC implementations (Chopin & Papaspiliopoulos, 2020) within `blackjax` affords a direct comparison (crucially, comparisons in prior work often lacked optimized NS implementations, see Section F).

3.1 Nested Slice Sampling

The combination of Algorithm 1 and Algorithm 2 (defined in Appendix B) define the form of Nested Sampling we dub here NSS. Whilst our implementation accommodates flexible declaration of proposal distributions, following the simplest definition in Algorithm 2, we take the proposal to be a normalized direction drawn from a multivariate Gaussian with online tuning using the covariance of the set of *live* particles. This tuning is run every outer kernel step but offers minimal overhead to the algorithm run time, more expressive tuning could be run less frequency to balance any incurred computational cost. An initial population is drawn from the reference density for the problem at hand. Following the steps of *deleting* live particles based on energy levels, and *resampling* from the remaining set, a short SS chain is run to construct new valid decorrelated states from the k duplicated start points. We configure the NSS algorithm with the following key hyperparameters, motivated by ablation studies (Section E) and typical usage. We maintain a population of $m = 1000$ particles, initialized from the prior distribution $\Pi(x)$. At each iteration, we employ the vectorized update described in Section D.3, removing the $k = 100$ particles with the highest energy values and generating replacements in parallel. To ensure the new particles are sufficiently decorrelated from their parents (selected from the surviving population), each replacement involves running the inner HRSS kernel (Algorithm 2) for p steps, where we set $p = 3 \times d$ proportional to the problem dimension d. Finally, the overall process terminates based on a standard NS evidence convergence criterion (Skilling, 2006): the algorithm stops when the estimated remaining contribution to the marginal likelihood integral (Equation (5)) from the current live particle set falls below a tolerance

of 0.001 relative to the evidence accumulated from deleted particles. While flexible termination conditions are possible, this default ensures the integration captures the vast majority of the evidence.

To utilize hardware acceleration, we apply the short SS chains in parallel, by vectorizing the inner kernel and applying it to the set of chosen parent particles for propagation. As demonstrated in Section F, HRSS benefits from a relatively low variance in the number of evaluations per step compared to constrained random walks, meaning whilst the vectorization isn't perfectly efficient (see Section D.3), it is still very worthwhile. Notably the SMC methods used as benchmarks do not require successful steps in their short chains. This is a key difference we revisit in the discussion.

3.2 SMC BENCHMARKS

To provide a benchmark, that similarly enables estimation of the normalizing constant with adaptive tuning, we use two variants of SMC with different choices of inner kernel. The overall adaptive scheme for SMC is detailed in (Fearnhead & Taylor, 2010), and similarly to our NSS implementation we consider a form that enables adaptive tuning of the inner kernel using the covariance of the live particles of the sampler. Specifically we use the `inner_kernel_tuning` algorithm as implemented in `blackjax`. For the inner kernel we consider two choices, Random Walk Metropolis Hastings (RW) and Independent Rosenbluth Metropolis Hastings (IRMH) (Robert & Casella, 2005). Both inner kernels involve a proposal, which is chosen to a multivariate normal, with covariance estimated from the live particles at each iteration. The static proposal of IRMH is a common choice (South et al., 2019), to implement an Importance Sampling like scheme in a particle Monte Carlo (Andrieu et al., 2010), this should perform similarly to Annealed Importance Sampling (Neal, 1998). The same proposal is used to form the RW inner kernel, however in this case its covariance is additionally scaled by a factor proportional to the dimension of the parameter space d, $2.38^2/d$. This scaling targets optimal acceptance rates for RW MH algorithms (Geyer, 1992).

To ensure a fair comparison with NSS, we configured the SMC benchmarks with analogous settings where possible. Both SMC variants utilized a population of 1000 particles, matching the NSS configuration. The MCMC propagation step within each SMC iteration involved running the respective inner kernel (RW or IRMH) for $p = 3 \times d$ attempted steps, directly mirroring the chain length factor used in NSS, where d is the problem dimension. Furthermore, we employed the adaptive tempering scheme described in (Fearnhead & Taylor, 2010), where the sequence of inverse temperatures β is determined automatically by targeting an Effective Sample Size (ESS) of 0.9 after each weighting step. This target ESS value controls the adaptivity, managing the trade-off between making progress towards the target distribution and maintaining particle diversity. While further problem-specific tuning might yield marginal improvements, we kept these parameters fixed across experiments to provide a consistent baseline comparison against NSS.

The effective sample size controls the size of the steps in the temperature β that are taken. There is some leeway to optimize these parameters for problem specific performance, but we have chosen to keep them fixed for the purposes of this study. Additionally, it would perhaps be expected to include an inner kernel of Hamiltonian Monte Carlo to be adaptively tuned in this setup (Buchholz et al., 2020), or include a more advanced proposal for the MCMC steps composed of a flow based neural network (Matthews et al., 2023). Both of these represent opportunities to further augment the basic setup of NSS, so we omit these in favor of a more like for like comparison. Parallel tempering is another "classical" method that has seen some recent development (Syed et al., 2021), which could have been selected as an alternative baseline. We omit propagation of errors on the normalizing constant estimates of SMC, which we take to form unbiased estimators with minimal counting error (Beskos et al., 2011).

4 EXPERIMENTS

In this section we review the performance of the proposed algorithm on a variety of benchmark problems. We consider a range of synthetic benchmarks, as well as a selection of problems from the Inference Gym (Sountsov et al., 2020). We evaluate the quality of the samples provided by each algorithm using the Maximum Mean Discrepancy (MMD) (Gretton et al., 2012), where the ground truth for comparison is composed of samples drawn from either analytically known results, or from a computationally exhaustive MCMC run. To understand the efficiency of each algorithm we report

the Effective Sample Size (ESS) of the resulting posterior approximation, amortized for runtime and number of energy evaluations required to reach the result. Details on this calculation are included in Section C.2. All experiments are run on either an M1 Max Studio CPU, or an NVIDIA T4 GPU. Details on uncertainty and ESS calculation are included in Section C.

4.1 Mixture of bivariate Gaussians

We consider a pedagogical example of a challenging, yet visualizable, 2D multimodal problem. Comprised of a mixture of 40 bivariate normal distributions on a uniform reference density, as introduced in (Midgley et al., 2023). It is noteworthy that this problem seems challenging for the many Normalizing Flow (Papamakarios et al., 2021b) based Neural Network approaches being explored for sampling. We display target samples obtained from running the various algorithms considered in Figure 3, and the results are encapsulated in Table 1. We include an example of Hamiltonian Monte Carlo with No-U-Turn sampling (Hoffman & Gelman, 2014) as a further control due to its ubiquity. However, as HMC does not estimate the normalizing constant Z, we exclude it from the main numerical comparisons in subsequent sections, though its struggles with this multimodal target (even in 2D) highlight the challenges faced by standard gradient-based methods. We find that both NSS and SMC trivially recover the target with competitive accuracy, indicated by the retrieval of $\ln Z$ and the MMD. Whilst using more energy evaluations to reach this state, when judged by ESS per evaluation or per second, NSS displays an edge in efficiency.

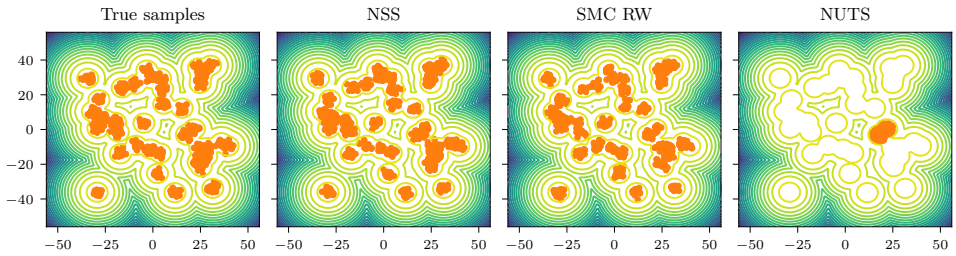

Figure 3: Mixture of 40 bivariate Gaussian distributions on a bounded uniform distribution.

4.2 Synthetic benchmarks

Next we introduce challenging synthetic benchmarks designed to probe sampler performance on specific pathological features common in inference problems, such as high dimensionality, strong correlations (Rosenbrock), hierarchical structures (Neal's funnel), and pronounced multimodality (Gaussian Mixture), where accurate posterior sampling and evidence estimation are difficult. We list the specific details of each benchmark in the following subsections, providing a visualization of the results in a 2D slice of parameter space in Figure 4 (with ground truth samples as larger blue points, and algorithm results in smaller orange points), alongside performance details in Table 2. Across these three experiments we see that NSS is consistently providing the highest quality of samples, as well as leading in efficiency of sample generation.

4.2.1 Mixture of Gaussians

An extension of the basic 2D example to a more challenging 10D problem, with a mixture of five 10-dimensional Gaussian distributions with randomized means and covariance.

4.2.2 Neal's Funnel

A ten dimension example of a funnel distribution, as defined in (Neal, 2003). The ground truth is found by running an HMC chain for many iterations and using a non-centered parameterization (Gorinova et al., 2019). For the algorithms under test we use the more challenging centered parameterization. The target distribution is defined as,

$$P(y, \mathbf{x}) = \mathcal{N}(y|0, 3) \prod_{n=1}^{9} \mathcal{N}(x_n|0, \exp(y/2)) \,, \tag{6}$$

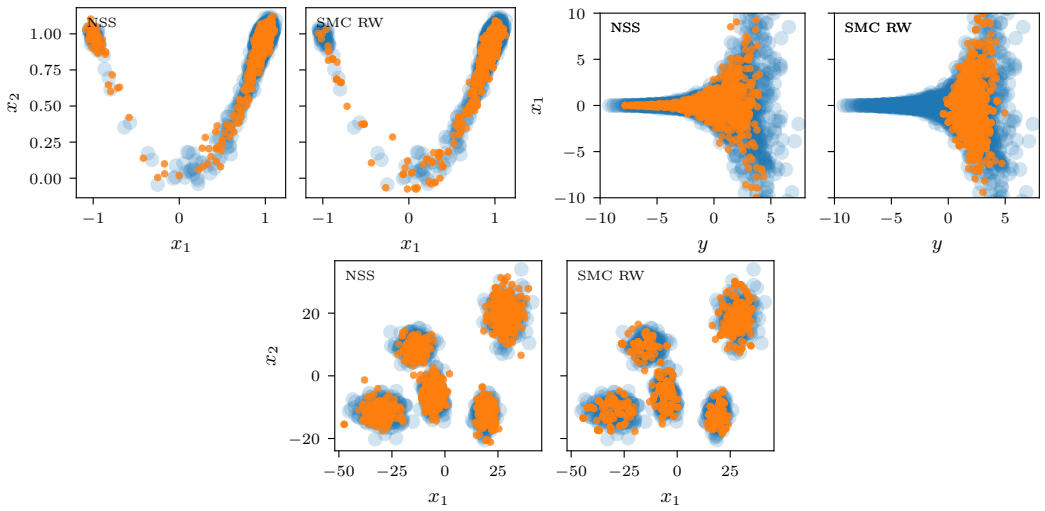

Figure 4: Marginal distributions of the first two parameters of the synthetic benchmarks. Rosenbrock [top left], Neal's funnel [top right], and Mixture of Gaussians [bottom].

With a uniform prior distribution on all parameters.

4.2.3 Rosenbrock Function

A ten-dimensional example of the Rosenbrock function on a uniform prior. The Rosenbrock function, is known for having a challenging sweeping degeneracy, is a common test problem for optimization algorithms. In its generalized n-dimensional form, its energy is defined as,

$$E(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right]. \tag{7}$$

4.3 Inference Gym

To complement the results on synthetic benchmarks, we also run the set of algorithms on some common intermediate dimension parameter space problems included in the inference gym (Sountsov et al., 2020) repository. These examples typically include data, and are realizations of Bayesian probabilistic models, where the energy function is the negative log likelihood of the model. We include the results in Table 3, noting the dimension of the parameter space of the problem as $[d]$. This package includes ground truth derived from exhaustive and selectively tuned HMC runs, performing parameter inference in high dimensional Bayesian inference models would typically be the domain of gradient assisted sampling methods.

An important quantity in Bayesian inference that is used to perform model selection is the Bayes factor, defined as the ratio of marginal likelihoods of two models, $\ln \text{BF}_{12} = \ln Z_1 - \ln Z_2$, where the integral in Equation (5) is now understood as the integral of the likelihood function over the prior on the parameters. Reference values for marginal likelihoods are hard to come by on such problems, HMC and other popular scalable methods explicitly do not estimate this, and the algorithms presented in this work are the state of the art on this task.

NSS still appears to consistently provide the most accurate posterior inference, whilst being competitive in runtime metrics, although the inefficiency of vectorization relative to the perfectly vectorized SMC is likely the cause here. We note that for the largest problem, based on a S&P500 volatility model, the prior contains a sizable set of invalid likelihood regions. Even after attempting to mask these as low likelihood, neither SMC algorithm is able to make any progress, but NSS sees invalid regions as a simple extension of its already constrained sampling set (a natural consequence of the HRSS kernel, Algorithm 2) and progresses fine.

5 DISCUSSION

In both SMC and NSS, the hyperparameter that is not comfortably set is the length of the short Markov Chains used to mutate the particles, p. This needs to be set sufficiently high to decorrelate the particle that is being duplicated, but overestimation of this length causes poor sampling efficiency. The version of SMC employed uses an 'embarrassingly' parallel update that attempts a fixed number of steps across all particles, Nested Sampling in its base form is at the extreme other end attempting to only update one particle at a time. The NSS approach detailed falls somewhere between this, and bears resemblance to *waste-free* variants of SMC (Dau & Chopin, 2021). The numerical experiments in Section 4.3 show that across the board NSS is able to be more efficient in terms of number of evaluations, but loses out on total runtime. This suggests that the variable number of evaluations required in the slice sampling construction (Figure 10) is somewhat wasteful when vectorized under strict synchronization (discussed in Section D.3), and an outstanding issue for Nested Sampling approaches is the ability to tune this length on the fly based on some measure of how decorrelated samples are across many short chains (Margossian et al., 2024). Similar discussions in the NS literature have focussed on a *dynamic* form of NS (Higson et al., 2018), dynamically tuning numbers of live particles, however this is badly suited for a static memory vectorized implementation, so we prefer to highlight tuning the length of the short chains (p) as a more tractable problem for future work.

Conceptually, the comparison highlights different adaptive strategies. Tempered SMC adapts its path through intermediate distributions by managing particle diversity (via ESS targets) along a predefined annealing coordinate (β). NSS, conversely, adapts its path by discovering iso-likelihood (iso-energy) contours dictated by the data itself, making its progression inherently tied to the likelihood structure. While this makes NSS potentially more direct for evidence estimation, it also imposes stricter requirements on the inner kernel's ability to handle the resulting hard constraints, unlike the greater kernel flexibility typically afforded in SMC.

On synthetic benchmarks, with complex geometries, NSS is notably strong across all metrics with minimal tuning. Although more detailed tuning of benchmark SMC algorithms may close this gap, the robustness and accuracy on a variety of $\mathcal{O}(10)$ dimensional problems explains and aligns with many usages in Bayesian inference for the physical sciences. Providing a vectorized implementation of NS is vital and has immense benefit for the physical sciences, where vectorized forward models (Wong et al., 2023b) and Neural Network emulators (Spurio Mancini et al., 2022) are increasingly commonplace. Tentative applications of the core NS procedure to Machine Learning applications such as marginalization of Gaussian Process hyperparameters (Simpson et al., 2021; Kroupa et al., 2024), has thus far used legacy MPI parallelized code and would benefit immensely from this vectorized implementation. We expect this implementation to be suitable for $\mathcal{O}(100)$ dimension problems, although more efficient gradient based short chains (an open challenge discussed in Section F.1) will be needed to progress much beyond this (Lemos et al., 2023).

6 CONCLUSION

Nested Sampling and Sequential Monte Carlo are state-of-the-art algorithms for providing unbiased estimates of the normalizing constant of a target distribution. Whilst there are other options, estimation of this normalizing constant, particularly in the presence of complex features present in many real world distributions mean this remains a notoriously difficult task. Nested Sampling is a technique that has become the workhorse for model comparison in many fields in the physical sciences, particularly those with a Bayesian preference such as astrophysics (Trotta, 2008), but has remained relatively obscure in the machine learning community. This paper addresses this by providing a clear, modern implementation of Nested Sampling with some important algorithmic refinements centered on slice sampling (Section 2.2) and key simplifications compared to legacy codes (e.g., direct prior sampling, no clustering, see Section D). We formulate a static memory version of the algorithm that compiles on modern GPU hardware in the `jax` ecosystem. This implementation is compatible with single digit precision, and allows highly parallelized, vectorized computation (Section D.3). We remove some of the more cumbersome details of existing NS algorithms, whilst retaining superior performance on many metrics, and this is demonstrated on a variety of challenging benchmarks in this work. We provide an open source and general framework for NS, and outline a number of open problems

and future research directions that this work enables, including improved adaptation via NDEs (Section F.2, Section F.3) and incorporation of gradient information (Section F.1).

Software and Data

The core algorithm implementation has been prepared for submission to a widely used open source library. On publication, explicit experiment example code will similarly be made available. All data is composed of standard benchmark problems, which are open source and online.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. We target a widely used method in the physical sciences and provide a modern implementation that is compatible with hardware acceleration. We believe this has the potential to enable a new class of problems to be solved in the physical sciences, and potential for innovative usage in the machine learning community. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Joshua G. Albert. Jaxns: a high-performance nested sampling package based on jax, 2020. URL https://arxiv.org/abs/2012.15286.

Justin Alsing and Will Handley. Nested sampling with any prior you like. *Monthly Notices of the Royal Astronomical Society: Letters*, 505(1):L95–L99, June 2021. ISSN 1745-3933. doi: 10.1093/mnrasl/slab057. URL http://dx.doi.org/10.1093/mnrasl/slab057.

Noemi Anau Montel, James Alvey, and Christoph Weniger. Scalable inference with autoregressive neural ratio estimation. *Mon. Not. Roy. Astron. Soc.*, 530(4):4107–4124, 2024. doi: 10.1093/mnras/stae1130.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 72(3):269–342, 05 2010. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2009.00736.x. URL https://doi.org/10.1111/j.1467-9868.2009.00736.x.

Greg Ashton et al. Nested sampling for physical scientists. *Nature*, 2, 2022. doi: 10.1038/s43586-022-00121-x.

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey, 2018. URL https://arxiv.org/abs/1502.05767.

Alexandros Beskos, Dan Crisan, Ajay Jasra, and Nick Whiteley. Error bounds and normalizing constants for sequential monte carlo in high dimensions, 2011. URL https://arxiv.org/abs/1112.1544.

Denis Blessing, Julius Berner, Lorenz Richter, and Gerhard Neumann. Underdamped diffusion bridges with applications to sampling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Q1QTxFm0Is.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Brendon J. Brewer and Daniel Foreman-Mackey. DNest4: Diffusive Nested Sampling in C++ and Python. 6 2016.

Alexander Buchholz, Nicolas Chopin, and Pierre E. Jacob. Adaptive tuning of hamiltonian monte carlo within sequential monte carlo, 2020. URL https://arxiv.org/abs/1808.07730.

Johannes Buchner. Nested sampling methods. 1 2021a. doi: 10.1214/23-ss144.

Johannes Buchner. Ultranest – a robust, general purpose bayesian inference engine, 2021b. URL https://arxiv.org/abs/2101.09604.

Johannes Buchner. Comparison of step samplers for nested sampling. In *MaxEnt 2022*, MaxEnt 2022, pp. 46. MDPI, February 2023. doi: 10.3390/psf2022005046. URL http://dx.doi.org/10.3390/psf2022005046.

Alberto Cabezas, Adrien Corenflos, Junpeng Lao, and Rémi Louf. Blackjax: Composable Bayesian inference in JAX, 2024.

Xiaohao Cai, Jason D. McEwen, and Marcelo Pereyra. Proximal nested sampling for high-dimensional bayesian model selection. *Statistics and Computing*, 32(5), October 2022. ISSN 1573-1375. doi: 10.1007/s11222-022-10152-9. URL http://dx.doi.org/10.1007/s11222-022-10152-9.

Junhua Chen, Lorenz Richter, Julius Berner, Denis Blessing, Gerhard Neumann, and Anima Anandkumar. Sequential controlled langevin diffusions. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=dImD2sgy86.

Yuansi Chen, Raaz Dwivedi, Martin J. Wainwright, and Bin Yu. Fast mcmc sampling algorithms on polytopes, 2019. URL https://arxiv.org/abs/1710.08165.

N. Chopin and O. Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer Series in Statistics. Springer International Publishing, 2020. ISBN 9783030478452. URL https://books.google.co.uk/books?id=ZZEAEAAAQBAJ.

N. Chopin and C. P. Robert. Properties of nested sampling. *Biometrika*, 97(3):741–755, June 2010. ISSN 1464-3510. doi: 10.1093/biomet/asq021. URL http://dx.doi.org/10.1093/biomet/asq021.

Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–551, 2002. ISSN 00063444. URL http://www.jstor.org/stable/4140600.

Hai-Dang Dau and Nicolas Chopin. Waste-free sequential monte carlo, 2021. URL https://arxiv.org/abs/2011.02328.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001.

Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows, 2019. URL https://arxiv.org/abs/1906.04032.

Paul Fearnhead and Benjamin M. Taylor. An adaptive sequential monte carlo sampler, 2010. URL https://arxiv.org/abs/1005.1193.

Farhan Feroz and M. P. Hobson. Multimodal nested sampling: an efficient and robust alternative to MCMC methods for astronomical data analysis. *Mon. Not. Roy. Astron. Soc.*, 384:449, 2008. doi: 10.1111/j.1365-2966.2007.12353.x.

Farhan Feroz and John Skilling. Exploring multi-modal distributions with nested sampling. In *AIP Conference Proceedings*. AIP, 2013. doi: 10.1063/1.4819989. URL http://dx.doi.org/10.1063/1.4819989.

Andrew Fowlie, Qiao Li, Huifang Lv, Yecheng Sun, Jia Zhang, and Le Zheng. Nested sampling statistical errors. *Monthly Notices of the Royal Astronomical Society*, 521(3):4100–4108, March 2023. ISSN 1365-2966. doi: 10.1093/mnras/stad751. URL http://dx.doi.org/10.1093/mnras/stad751.

Marylou Gabrié, Grant M. Rotskoff, and Eric Vanden-Eijnden. Adaptive Monte Carlo augmented with normalizing flows. *Proc. Nat. Acad. Sci.*, 119(10):e2109420119, 2022. doi: 10.1073/pnas.2109420119.

Charles J. Geyer. Practical Markov Chain Monte Carlo. *Statistical Science*, 7(4):473 – 483, 1992.

Maria I. Gorinova, Dave Moore, and Matthew D. Hoffman. Automatic reparameterisation of probabilistic programs, 2019. URL https://arxiv.org/abs/1906.03028.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

Roger B. Grosse, Zoubin Ghahramani, and Ryan P. Adams. Sandwiching the marginal likelihood using bidirectional monte carlo, 2015. URL https://arxiv.org/abs/1511.02543.

W. J. Handley, M. P. Hobson, and A. N. Lasenby. polychord: next-generation nested sampling. *Mon. Not. Roy. Astron. Soc.*, 453(4):4385–4399, 2015. doi: 10.1093/mnras/stv1911.

Edward Higson, Will Handley, Michael Hobson, and Anthony Lasenby. Dynamic nested sampling: an improved algorithm for parameter estimation and evidence calculation. *Statistics and Computing*, 29(5):891–913, December 2018. ISSN 1573-1375. doi: 10.1007/s11222-018-9844-0. URL http://dx.doi.org/10.1007/s11222-018-9844-0.

Edward Higson, Will Handley, Mike Hobson, and Anthony Lasenby. nestcheck: diagnostic tests for nested sampling calculations. *Monthly Notices of the Royal Astronomical Society*, 483(2):2044–2056, 2019. doi: 10.1093/mnras/sty3090. URL http://doi.org/10.1093/mnras/sty3090.

Matthew Hoffman, Alexey Radul, and Pavel Sountsov. An adaptive-mcmc scheme for setting trajectory lengths in hamiltonian monte carlo. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3907–3915. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/hoffman21a.html.

Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(47):1593–1623, 2014. URL http://jmlr.org/papers/v15/hoffman14a.html.

Matthew D. Hoffman and Pavel Sountsov. Tuning-free generalized hamiltonian monte carlo. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 7799–7813. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/hoffman22a.html.

Charles R. Keeton. On statistical uncertainty in nested sampling: Statistical uncertainty in nested sampling. *Monthly Notices of the Royal Astronomical Society*, 414(2):1418–1426, April 2011. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2011.18474.x. URL http://dx.doi.org/10.1111/j.1365-2966.2011.18474.x.

Namu Kroupa, David Yallup, Will Handley, and Michael Hobson. Kernel-, mean-, and noise-marginalized gaussian processes for exoplanet transits and h 0 inference. *Monthly Notices of the Royal Astronomical Society*, 528(2):1232–1248, 2024.

Johannes U. Lange. nautilus: boosting Bayesian importance nested sampling with deep learning. *Mon. Not. Roy. Astron. Soc.*, 525(2):3181–3194, 2023. doi: 10.1093/mnras/stad2441.

Yin Tat Lee and Santosh S. Vempala. Geodesic walks on polytopes. *CoRR*, abs/1606.04696, 2016. URL http://arxiv.org/abs/1606.04696.

Pablo Lemos, Nikolay Malkin, Will Handley, Yoshua Bengio, Yashar Hezaveh, and Laurence Perreault-Levasseur. Improving gradient-guided nested sampling for posterior inference, 2023. URL https://arxiv.org/abs/2312.03911.

F. Llorente, L. Martino, D. Delgado, and J. López-Santiago. Marginal likelihood computation for model selection and hypothesis testing: An extensive review. *SIAM Review*, 65(1):3–58, 2023.

David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, USA, 2002. ISBN 0521642981.

Charles C Margossian, Matthew D Hoffman, Pavel Sountsov, Lionel Riou-Durand, Aki Vehtari, and Andrew Gelman. Nested ˆr: Assessing the convergence of markov chain monte carlo when running many short chains. *Bayesian Analysis*, 1(1):1–28, 2024.

Alexander G. D. G. Matthews, Michael Arbel, Danilo J. Rezende, and Arnaud Doucet. Continual repeated annealed flow transport monte carlo, 2023. URL https://arxiv.org/abs/2201.13117.

Laurence Illing Midgley, Vincent Stimper, Gregor N. C. Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap, 2023. URL https://arxiv.org/abs/2208.01893.

Adam Moss. Accelerated Bayesian inference using deep learning. *Mon. Not. Roy. Astron. Soc.*, 496 (1):328–338, 2020. doi: 10.1093/mnras/staa1469.

Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL http://probml.github.io/book1.

Iain Murray, David MacKay, Zoubin Ghahramani, and John Skilling. Nested sampling for potts models. In Y. Weiss, B. Schölkopf, and J. Platt (eds.), *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005. URL https://proceedings.neurips.cc/paper_files/paper/2005/file/9dc372713683fd865d366d5d9ee810ba-Paper.pdf.

Lawrence M. Murray, Anthony Lee, and Pierre E. Jacob. Parallel resampling in the particle filter, 2015. URL https://arxiv.org/abs/1301.4019.

Radford M. Neal. Annealed importance sampling, 1998. URL https://arxiv.org/abs/physics/9803008.

Radford M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705 – 767, 2003. doi: 10.1214/aos/1056562461. URL https://doi.org/10.1214/aos/1056562461.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021a.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference, 2021b. URL https://arxiv.org/abs/1912.02762.

Lívia B Pártay, Albert P Bartók, and Gábor Csányi. Nested sampling for materials: The case of hard spheres. *Physical Review E*, 89(2):022302, 2014.

Daniel Paulin, Ajay Jasra, and Alexandre Thiery. Error bounds for sequential monte carlo samplers for multimodal distributions, 2018. URL https://arxiv.org/abs/1509.08775.

Nicholas G. Polson and James G. Scott. Vertical-likelihood monte carlo, 2015. URL https://arxiv.org/abs/1409.3601.

Sam Power, Daniel Rudolf, Björn Sprungk, and Andi Q. Wang. Weak poincaré inequality comparisons for ideal and hybrid slice sampling, 2024. URL https://arxiv.org/abs/2402.13678.

Metha Prathaban and Will Handley. Costless correction of chain based nested sampling parameter estimation in gravitational wave data and beyond, 2024. URL https://arxiv.org/abs/2404.16428.

Lorenz Richter and Julius Berner. Improved sampling via learned diffusions. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=h4pNROsO06.

Lionel Riou-Durand, Pavel Sountsov, Jure Vogrinc, Charles C. Margossian, and Sam Power. Adaptive tuning for metropolis adjusted langevin trajectories, 2023. URL https://arxiv.org/abs/2210.12200.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 0387212396.

Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341 – 363, 1996.

Daniel Rudolf and Mario Ullrich. Comparison of hit-and-run, slice sampler and random walk metropolis. *Journal of Applied Probability*, 55(4):1186–1202, December 2018. ISSN 1475-6072. doi: 10.1017/jpr.2018.78. URL http://dx.doi.org/10.1017/jpr.2018.78.

Robert Salomone, Leah F. South, Adam M. Johansen, Christopher Drovandi, and Dirk P. Kroese. Unbiased and consistent nested sampling via sequential monte carlo, 2024. URL https://arxiv.org/abs/1805.03924.

Fergus Simpson, Vidhi Lalchand, and Carl Edward Rasmussen. Marginalised gaussian processes with nested sampling, 2021. URL https://arxiv.org/abs/2010.16344.

John Skilling. Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1(4):833 – 859, 2006. doi: 10.1214/06-BA127. URL https://doi.org/10.1214/06-BA127.

Robert L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/170949.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *CoRR*, abs/2011.13456, 2020. URL https://arxiv.org/abs/2011.13456.

Pavel Sountsov, Alexey Radul, and contributors. Inference gym, 2020. URL https://pypi.org/project/inference_gym.

L. F. South, A. N. Pettitt, and C. C. Drovandi. Sequential Monte Carlo Samplers with Independent Markov Chain Monte Carlo Proposals. *Bayesian Analysis*, 14(3):753 – 776, 2019.

Joshua S Speagle. dynesty: a dynamic nested sampling package for estimating bayesian posteriors and evidences. *Monthly Notices of the Royal Astronomical Society*, 493(3):3132–3158, February 2020. ISSN 1365-2966. doi: 10.1093/mnras/staa278. URL http://dx.doi.org/10.1093/mnras/staa278.

Alessio Spurio Mancini, Davide Piras, Justin Alsing, Benjamin Joachimi, and Michael P Hobson. Cosmopower: emulating cosmological power spectra for accelerated bayesian inference from next-generation surveys. *Monthly Notices of the Royal Astronomical Society*, 511(2):1771–1788, January 2022. ISSN 1365-2966. doi: 10.1093/mnras/stac064. URL http://dx.doi.org/10.1093/mnras/stac064.

Saifuddin Syed, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Non-reversible parallel tempering: A scalable highly parallel mcmc scheme. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(2):321–350, December 2021. ISSN 1467-9868. doi: 10.1111/rssb.12464. URL http://dx.doi.org/10.1111/rssb.12464.

Jesús Torrado, Nils Schöneberg, and Jonas El Gammal. Parallelized Acquisition for Active Learning using Monte Carlo Sampling. 5 2023.

Roberto Trotta. Bayes in the sky: Bayesian inference and model selection in cosmology. *Contemporary Physics*, 49(2):71–104, March 2008. ISSN 1366-5812. doi: 10.1080/00107510802066753. URL http://dx.doi.org/10.1080/00107510802066753.

Michael J. Williams, John Veitch, and Chris Messenger. Nested sampling with normalizing flows for gravitational-wave inference. *Phys. Rev. D*, 103(10):103006, 2021. doi: 10.1103/PhysRevD.103.103006.

Kaze W. k. Wong, Marylou Gabrié, and Daniel Foreman-Mackey. flowMC: Normalizing flow enhanced sampling package for probabilistic inference in JAX. *J. Open Source Softw.*, 8(83):5021, 2023a. doi: 10.21105/joss.05021.

Kaze W. K. Wong, Maximiliano Isi, and Thomas D. P. Edwards. Fast Gravitational-wave Parameter Estimation without Compromises. *Astrophys. J.*, 958(2):129, 2023b. doi: 10.3847/1538-4357/acf5cd.

Yan Zhou, Adam M Johansen, and John A D Aston. Towards automatic model comparison: An adaptive sequential monte carlo approach, 2015. URL https://arxiv.org/abs/1303.3123.

Krzysztof Łatuszyński, Matthew T. Moores, and Timothée Stumpf-Fétizon. Mcmc for multi-modal distributions, 2025. URL https://arxiv.org/abs/2501.05908.

A Results Tables

Here we include tables detailing the results of the numerical experiments run in the main text.

Table 1: Algorithm performance details for the pedagogical mixture of bivariate Gaussians problem.

Algorithm	Energy evals	ESS	Time (s)	MMD	$\ln Z$	ESS /1k eval	ESS /second
Ground Truth	-	-	-	-	-9.21	-	-
NSS	455628	3743	1.9	9.57×10^{-3}	-9.18 ± 0.05	8.2	2×10^{3}
SMC RW	190000	989	0.59	7.46×10^{-3}	-9.24	5.2	1.7×10^{3}
SMC IRMH	190000	991	0.6	4.80×10^{-3}	-9.21	5.2	1.6×10^{3}

Table 2: Algorithm performance details for synthetic benchmarks.

Algorithm	Energy evals	ESS	Time (s)	MMD	$\ln Z$	ESS /1k eval	ESS /second
Funnel							
NSS	12276194	37086	6.7	0.052	-30.03 ± 0.13	3	5.6×10^{3}
SMC RW	2350000	985	1.6	0.39	-31.37	0.42	6.2×10^{2}
SMC IRMH	2300000	960	1.6	0.43	-31.38	0.42	6.1×10^{2}
Gaussian Mixture							
Ground Truth	-	-	-	-	-46.05	-	-
NSS	3469821	8737	5.1	0.062	-46.40 ± 0.13	2.5	1.7×10^{3}
SMC RW	1170000	954	1.3	0.19	-45.92	0.82	7.3×10^{2}
SMC IRMH	1170000	989	1.4	1.5	-47.33	0.85	7.1×10^{2}
Rosenbrock							
NSS	6856062	9937	5.4	0.22	-50.14 ± 0.22	1.4	1.8×10^{3}
SMC RW	3180000	999	2.3	0.79	-50.78	0.31	4.4×10^{2}
SMC IRMH	3150000	998	2.4	2	-51.66	0.32	4.1×10^{2}

Table 3: Results on benchmarks from the inference gym set of problems.

Algorithm	Energy evals	ESS	Time (s)	MMD	$\ln Z$	ESS /1k eval	ESS /second
Eight Schools $[d] = 10$							
NSS	1534238	5180	4	6.03×10^{-3}	-36.19 ± 0.08	3.4	1.3×10^{3}
SMC RW	1320000	990	1.9	8.28×10^{-3}	-36.23	0.75	5.2×10^{2}
SMC IRMH	1350000	989	2.2	8.49×10^{-3}	-36.12	0.73	4.6×10^{2}
Credit Logistic $[d] = 25$							
NSS	20723029	11671	2.1×10^{2}	7.57×10^{-3}	-529.16 ± 0.26	0.56	56
SMC RW	4875000	942	12	9.17×10^{-3}	-528.98	0.19	80
SMC IRMH	4875000	960	13	0.012	-529.14	0.2	76
Radon Contextual $[d] = 97$							
NSS	128345045	15353	9.1×10^{2}	7.78×10^{-3}	-2590.43 ± 0.28	0.12	17
SMC RW	27936000	999	55	0.011	-2601.00	0.036	18
SMC IRMH	28227000	960	71	0.012	-2610.48	0.034	14
S&P500 Small $[d] = 103$							
NSS	40684489	11670	1.9×10^{3}	0.036	-688.23 ± 9.92	0.29	6.2
SMC RW	-	-	-	-	-	-	-
SMC IRMH	-	-	-	-	-	-	-

B ALGORITHM DETAILS

Here to complement the description of the algorithm in the main text, we include the pseudo-code for the main algorithms discussed.

Algorithm 1 Nested Sampling outer Kernel

Input: Particles $\{x_i\}_0^m$, deletion number k
Sort $\{x_i\}_0^m$ by $E(x_i)$, finding top-k energy levels
Define $E_{\min}$ as min of top-k energy levels
Weight by boolean $w_i = (E(x_i) < E_{\min})$
Resample k parents from $\{x_i\}_0^m$ with replacement by w_i
Propagate k duplicated particles with $\{x_i\}_{i=1}^k \leftarrow \{x_i'\}_{i=1}^k$, $E(x_i') < E_{\min}$ with some inner kernel for p steps
Output: $\{x_i\}_0^m$, $E_{\min}$

Algorithm 2 Inner kernel Hit and Run Slice Sampling for Nested Sampling

Input: Current state x, Target distribution $\Pi(x)$, Conditioning matrix M, Energy function $E(x)$, Auxiliary energy variable $E_{\min}$
Vertical slice $y \leftarrow u \times \Pi(x), u \sim \mathcal{U}(0,1)$
Sample direction $d \sim \mathcal{N}(0, M)$
Normalize $d \leftarrow d/\|d\|$
Draw initial slice direction $w \sim \mathcal{U}(0,1)$
Initialize $l \leftarrow x - wd, r \leftarrow x + (1-w)d$
{*Stepping-out phase: Find interval boundaries satisfying both constraints*}
while $\Pi(l) \geq y$ **and** $E(l) < E_{\min}$ **do**
  $l \leftarrow l - d$
end while
while $\Pi(r) \geq y$ **and** $E(r) < E_{\min}$ **do**
  $r \leftarrow r + d$
end while
{*Rejection phase: Sample uniformly from interval until valid point found*}
loop
  Propose $x' \sim \text{Uniform}(l, r)$
  if $\Pi(x') \geq y$ **and** $E(x') < E_{\min}$ **then**
    break {*Accept valid sample x' and exit loop*}
  else
    {*Shrink interval based on invalid proposal x'*}
    if $x' < x$ **then**
      $l \leftarrow x'$
    else
      $r \leftarrow x'$
    end if
  end if
end loop
Output: x'

Algorithm 3 Tempered Sequential Monte Carlo

Input: Particles $\{x_i\}_0^m$, weight w_i^N, temperature β^N
Resample $\{x_i\}_0^m$ with replacement with probability w_i^N
Propagate $\{x_i\}_0^m \to \{x_i'\}_0^m$ with some inner kernel for p steps.
Weight by β^{N+1}, $w_i^{N+1} \leftarrow \exp(-\beta^{N+1} - \beta^N E(x_i'))$
Output: Particles $\{x_i'\}_0^m$, weight w_i^{N+1}, temperature β^{N+1}

C Probabilistic Volume Estimation and Uncertainty Quantification

Nested Sampling (NS) estimates the normalizing constant (Equation (5)) by summing contributions from particles deleted at increasing energy levels. The weight of each contribution depends on the estimated prior volume shrinkage ΔX_i associated with deleting particle i. Following Section 2.1, this section details how to accurately estimate these volumes and propagate the associated uncertainty, particularly for the vectorized ($k > 1$) version of NS.

C.1 Tracking the Effective Number of Live Particles

Our vectorized implementation (Algorithm 1 with $k > 1$) simultaneously removes the k particles with the highest energies. Let these particles have energies $E_0 > E_1 > \cdots > E_{k-1}$, where E_{k-1} is the energy threshold $E_{\min}$ for the batch removal.

While the particles are removed together, we can analyze the expected volume shrinkage by conceptualizing k sequential single-particle removals ordered by these energies. The expected log-volume shrinkage associated with the removal of the $(j+1)$-th particle in this ordered subsequence (the one with energy E_j) reflects the reduction in live particle count from $m - j$ to $m - j - 1$:

$$\Delta \ln X_{i,j} = -\frac{1}{m-j}, \quad \text{for } j \in \{0, \ldots, k-1\}. \tag{8}$$

The total expected log-shrinkage for removing the entire batch is the sum of these individual expected contributions: $\sum_{j=0}^{k-1}(-\frac{1}{m-j})$. Noting that i still indexes the running outer sum in Equation (5). This provides a more accurate point estimate for the expected volume contraction at an outer step i than the simple approximation $-k/m$. Crucially, this calculation provides the expectation of the shrinkage, and we can deal with this uncertainty as detailed in Section C.2. In simple terms this means rather than jumping k energy levels, we jump k times with a smaller geometric sample each time, before replenishing back to m total particles.

C.2 Stochastic Estimation of Prior Volumes and Importance Weights

While Section C.1 discussed the *expected* volume shrinkage, robust uncertainty quantification in Nested Sampling relies on simulating the *stochastic* nature of the volume compression process post-hoc (Skilling, 2006; Keeton, 2011). This is the strategy used in our experiments.

To simplify the notation, we can drop the second index introduced in Equation (8) and absorb the changing volume estimate into an effective number of live particles at iteration i, as m_i^*. At each energy level transition, provided we have drawn a new *decorrelated* sample from the level set, the next energy level is sampled uniformly within the level set. The stochastic change in log-volume associated with the i-th particle deletion is then simulated by sampling $u_i \sim \mathcal{U}(0, 1)$:

$$\Delta \ln X_i = \frac{\ln u_i}{m_i^*}. \tag{9}$$

Using this simulated volume element ΔX_i, the importance weight for particle i (at inverse temperature β) is:

$$\log w_i(\beta) = \ln(\Delta X_i) - \beta E_i. \tag{10}$$

An ensemble of weights for each particle can be calculated by resampling the stochastic volume estimates in Equation (9). This ensemble of weights is then used to compute expectations and uncertainties on the normalizing constant estimate and any other related derived quantity.

This simulation captures the primary volume uncertainty. Other potential error sources may require different diagnostics, explored in the literature (Prathaban & Handley, 2024; Higson et al., 2019).

C.3 Demonstration

To demonstrate the practical application of the stochastic volume estimation, we replicate the synthetic 10D mixture of Gaussian experiment from Section 4.2. We perform multiple runs of both SMC RW and NSS, each initiated with different random seeds.

For **NSS**, the procedure detailed in Section C.2 is used on the output of each run (or the combined output of all runs) to derive a standard deviation for the log-normalizing constant estimate ($\ln Z$). This standard deviation quantifies the uncertainty originating from the stochastic volume compression inherent in the NS method.

For **SMC**, such an internal error estimate is not directly available from the standard algorithm. Therefore, we assess its performance variability by calculating the empirical mean and standard deviation of the $\ln Z$ estimates across the multiple independent runs.

An additional benefit of the NSS post-hoc analysis is that particle lists from multiple runs can be concatenated before applying the volume simulation procedure. This allows for a single, combined $\ln Z$ estimate and uncertainty derived from the full set of generated samples, potentially offering improved precision compared to analyzing runs individually.

The results are visualized in Figure 5. The plot compares the distribution of $\ln Z$ estimates from multiple SMC runs against the NSS estimates. On this challenging multimodal problem, the figure illustrates the utility and accuracy of the internally generated NSS error bars compared to the SMC variant.

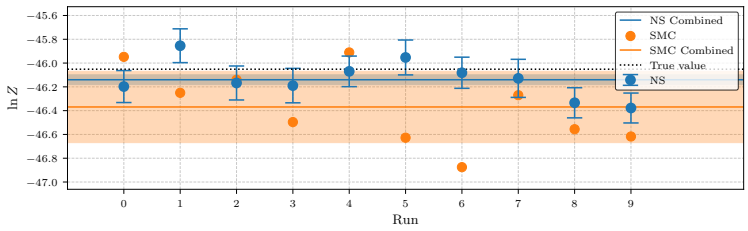

Figure 5: Log-normalizing constant estimate across 10 reseeded runs for a 10D Gaussian Mixture target.

D Advantages versus existing Nested Sampling Implementations

In this section we consider some important differences that distinguish the algorithm proposed in this work from existing work, particularly focussing on aspects not covered in the main text.

D.1 Encoding a reference (prior) density

Standard implementations of NS typically use a transform from the unit hypercube to encode a prior density. This is convenient and for most proper priors a largely moot point on paper. However, in practice it offers a number of distinct disadvantages that we lift by designing an algorithm to work directly with prior densities.

- **Numerical stability:** The transform from the unit hypercube to the prior density can be numerically unstable, this is compounded for higher dimensional problems, and particularly harshly visible when using single precision numerics. For ML and hardware accelerated version of the algorithm this is a strong motivation.
- **Flexibility:** For some applications such as online Bayesian updating this would require representing previous results with a learnt transform from the unit hypercube, and whilst feasible (Alsing & Handley, 2021), it is restrictive in certain use cases.
- **Geometry:** The simplest probabilistic inference case of a Gaussian Prior and Gaussian likelihood, and hence Gaussian posterior, has the simplest geometry one could hope for, a level set comprised of a hyperellipsoid. If one is reliant on a transform to the unit hypercube to perform sampling in that space, this geometry is naturally deformed, taking a simple problem and making it harder. We demonstrate this for a simple bivariate normal distribution in Figure 6.

In summary the insistence on a representation of the prior as a transform from the unit hypercube is largely a historical artifact derived from initial usage of rejection sampling, where arbitrary truncation

of non-uniform priors to make sampling efficient are largely impossible otherwise. Leaving this behind brings the algorithm in much closer alignment with modern probabilistic programming languages, and the ability to use the algorithm in a wider range of applications.

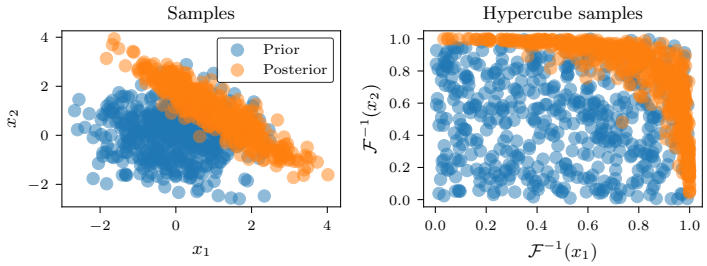

Figure 6: Samples drawn in a space defined on the real metric of the prior [left], and a representation of the same samples after the inverse CDF of the prior density $\mathcal{F}^{-1}$ is used to map these to the unit hypercube [right]. The target (posterior) geometry is warped and even a simple Gaussian problem becomes non-linear.

D.2 Clustering and density estimation

Another common feature of existing Nested Sampling implementations is the use of clustering algorithms on the live particle set $\{x_i\}_0^m$. By identifying distinct groups and often fitting simple geometric shapes (e.g., ellipsoids based on local covariance) to them, these methods perform a form of rudimentary density estimation. This is used to create locally adapted proposals, a necessity for scaling early rejection sampling based implementations (Feroz & Hobson, 2008) which struggle with disparate modes using a single proposal distribution.

In contrast, our Markov Chain inner kernel approach (NSS) makes explicit clustering unnecessary. NSS primarily adapts **globally** using information from the entire live set (e.g., updating the conditioning matrix M based on the global covariance) combined with the **inherent local adaptation** of the slice sampling steps themselves (finding boundaries via stepping-out). It does not build explicit local density models for separate modes. This offers significant advantages in terms of algorithmic **simplicity** (removing the need for a complex clustering module) and **efficiency** (avoiding clustering overhead and potentially facilitating better hardware vectorization).

While effective, the global adaptation in NSS could be enhanced. There would be clear benefit in using more modern techniques to construct better inner kernel proposals, potentially replacing the simple global covariance M with explicit **Neural Density Estimators** (NDEs). We discuss opportunities for Neural Network proposals further in Section F.2. NDEs offer potential advantages over legacy clustering implementations due to better scalability to high dimensions and their suitability for parallel GPU computation compared to often sequential or tree-based clustering algorithms. This mirrors improvements in modern SMC algorithms, where issues like sample impoverishment and mode collapse are mitigated by incorporating adaptive proposals, using NDEs to better capture multimodal targets and guide particle propagation (Matthews et al., 2023).

We consider the same 10D mixture of Gaussians problem from Section 4.2. The first two dimensions of the resulting posterior distribution are shown in Figure 7, and the results are encapsulated in Table 4. Despite lacking clustering, NSS recovers the posterior distribution accurately (lowest MMD) and efficiently, offering a drastic improvement in walltime (over an order of magnitude on this benchmark) compared to PolyChord (Handley et al., 2015) or UltraNest (Buchner, 2021b). While all algorithms provide good solutions, this demonstrates the viability and performance benefits of the non-clustering approach.

D.3 Vectorization for Parallel Acceleration

With the increased proliferation of Neural Network surrogates, neural Simulation-Based Inference, and the general utilization of GPU acceleration in scientific problems, sampling algorithms that

Table 4: Comparison between popular legacy implementations and the current work.

Algorithm	MMD	$\ln Z$	Energy evals
Ground truth	-	-46.05	-
NSS	0.018	-46.1±0.2	2910702
PolyChord	0.055	-46.1±0.2	3309636
UltraNest	0.057	-45.7±0.32	3143172

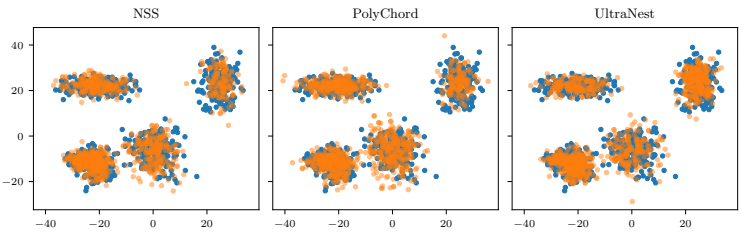

Figure 7: Comparison of first two dimensions of a 10D mixture model, comparing different legacy implementations with the current work.

efficiently leverage parallel hardware have become essential. However, implementing typically control-flow-heavy MCMC methods effectively on architectures like GPUs presents challenges (Hoffman & Sountsov, 2022; Hoffman et al., 2021; Riou-Durand et al., 2023).

Our approach accelerates Nested Sampling by exploiting the flexibility in choosing the number of particles, k, deleted per outer iteration (Algorithm 1). The standard implementation uses $k = 1$, proceeding sequentially. Choosing $k > 1$ allows k particles to be deleted simultaneously, and crucially, allows the generation of k replacement particles (typically by duplicating k survivors and evolving them with the inner MCMC kernel) to be performed in parallel.

The efficiency of this parallelization is limited by the inner MCMC step, here Hit-and-Run Slice Sampling (Algorithm 2). Slice sampling involves a variable number of likelihood evaluations per call (e.g., during stepping-out). When executing k chains in parallel (e.g., via `jax.vmap`), the time for the collective step is determined by the chain requiring the maximum number of evaluations. Therefore, the speedup is not perfectly linear in k. However, despite this inherent variability, significant wall-clock time reductions are achievable, especially on highly parallel hardware like GPUs.

To demonstrate the efficiency and trade-offs, we consider a Gaussian linear model ($y \sim \mathcal{N}(A\theta, C)$) with Gaussian priors on θ, where A is a random matrix. Defining a data generating process for y_i, subject to Gaussian noise C, as,

$$A_{ij} \leftarrow \mathcal{N}(0, I) \tag{11}$$
$$\theta_j \leftarrow \mathcal{N}(0, I) \tag{12}$$
$$y_i \leftarrow \mathcal{N}(A_{ij}\theta_j, C)\,. \tag{13}$$

By simulating a *true* value for the parameters θ_j, a data observation y_i is generated. Such a model has an analytic Gaussian expression for the distribution of θ conditioned on the simulated observation, and allows scaling to large parameter spaces by increasing j, and large data vectors by scaling i. To test the efficiency of vectorization we consider an example where $i = 100$, $j = 10$ and $C = I$.

Using NSS to infer θ, we start with $m = 500$ live particles and vary the deletion number k from 1 to 400. Figure 8 shows the effect on total runtime and the error in the estimated log marginal likelihood ($\ln Z$). Vectorization provides substantial runtime benefits, with diminishing returns as k (and thus the fraction k/m) increases when run on CPU hardware, however exhibiting no diminishing returns when run on a GPU. This speedup comes at some cost in accuracy: the error in $\ln Z$ increases, appreciably increasing when deleting over half of the particles. This increased error arises because larger k leads to larger jumps between the energy contours where particles are deleted. This results in a coarser discretization of the evidence integral (Equation (5)), increasing the variance of the numerical integration.

The shape of the runtime curve depends on the problem specifics (e.g., cost of likelihood evaluation relative to MCMC overhead) and hardware. The explicit comparison between CPU and GPU runs in Figure 8 (normalized) highlights the vastly superior scalability achievable with GPU acceleration for this vectorized approach.

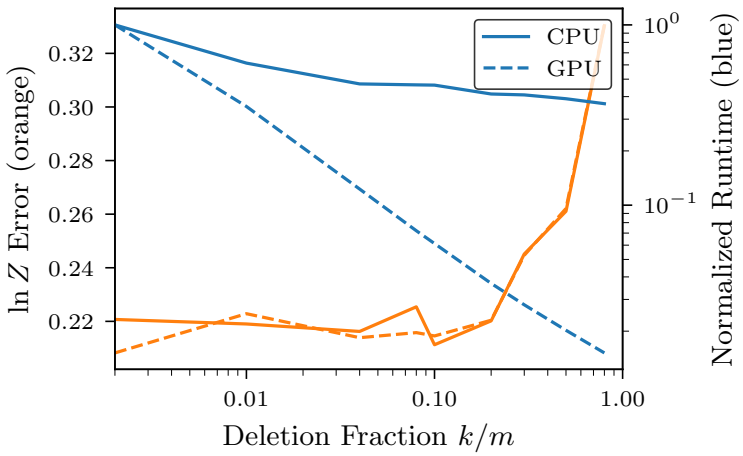

Figure 8: Ablation of the deletion fraction (k/m) as a hyperparameter of the NSS algorithm, effect on runtime is shown for a simple Gaussian Linear model with other hyperparameters fixed. Default value used elsewhere in this work corresponds to $k/m = 0.1$.

E Hyperparameters of Nested Slice Sampling

We identified the main hyperparameters of NSS in Section 3. In this section, we provide ablation studies on a controlled problem to motivate the default choices used in our empirical studies. We chose a $d = 20$ dimensional Gaussian inference problem: comprised of a multivariate Gaussian prior and energy (likelihood) function, an inference problem where the target distribution $P(x)$ is also Gaussian hence has analytic form and a known normalizing constant. The prior distribution is taken to be a unit multivariate normal distribution, and the energy (likelihood) function $E(x)$ is a d-dimensional Gaussian $E(x) = \mathcal{N}(x|\mu_E, \Sigma_E)$. The parameters of the energy function were randomly generated (but fixed across ablations) for this experiment as,

$$\mu_{E,i} \leftarrow \mathcal{N}(\mathbf{0}, I)\,, \tag{14}$$

$$C_{i,j} \leftarrow \mathcal{N}(0, 0.1^2)\,, \tag{15}$$

$$\Sigma_E = CC^T\,. \tag{16}$$

In Figure 9, we explore how varying key hyperparameters affects performance, measured by the error in the estimate of $\ln Z$ and the total number of likelihood/energy evaluations required. We analyze:

- **Population size (m):** (Left panel) As expected, increasing the number of live particles m improves the accuracy of the $\ln Z$ estimate (error scales roughly as $1/\sqrt{m}$), primarily by reducing the statistical variance in the stochastic volume estimations (Equation (9)). This comes at a roughly linear increase in the total number of energy evaluations.
- **MCMC Chain Length Factor (p):** (Middle panel) This factor determines the number of inner MCMC steps p taken to decorrelate duplicated particles. Initially, longer chains improve sample decorrelation, potentially reducing bias or variance in $\ln Z$, but beyond a certain point (e.g., $p \approx 2 - 5 \times d$ here), there are diminishing returns on accuracy while the computational cost continues to grow linearly.
- **Proposal Scale Factor (ϵ):** (Right panel) This factor scales the covariance matrix M used for proposing directions in the Hit-and-Run Slice Sampling kernel (proposal $\sim \mathcal{N}(0, \epsilon^2 M)$). Setting ϵ optimally balances the efficiency of the slice sampling steps; too small may lead to slow exploration, too large may lead to inefficient stepping-out or rejection within the slice.

The results suggest that $\epsilon = 1.0$ (using the unmodified covariance) performs well for this problem, but we are largely insensitive to this on this idealised problem.

The deletion fraction k/m was analyzed separately in Section D.3 due to its direct link to vectorization. These ablation studies illustrate the typical trade-offs between computational cost and accuracy inherent in tuning NS hyperparameters. The default values used in this work ($m = 1000$, $p = 3 \times d$, $\epsilon = 1.0$, and $k/m = 0.1$) represent a reasonable and practical balance for the benchmark problems considered.

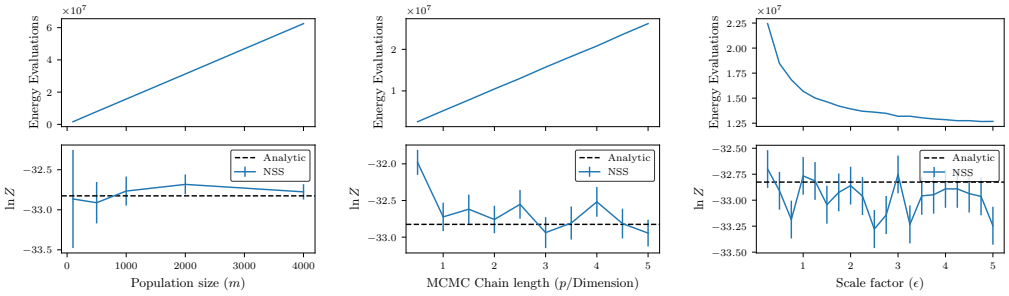

Figure 9: Ablation of 3 hyperparameters of the NSS algorithm on a randomized 20 dimension Gaussian Prior, Gaussian Energy/Likelihood problem. Scaling the population size (left), scaling the length of the short inner Markov chains (middle) and tuning a scale factor of the particle covariance slice proposal (right). Defaults used in this work are $m = 1000$, $p = 3 \times d$, and a scale factor of $\epsilon = 1.0$.

F Challenges of Constrained Prior sampling

A core challenge in Nested Sampling is efficiently drawing samples from the prior distribution $\Pi(x)$ restricted to the dynamically shrinking region where the energy $E(x)$ is less than the current threshold ($E(x) < E_{\min}$). This hard energy constraint poses difficulties for many standard MCMC techniques, often rendering them suboptimal due to high rejection rates near the boundary. Some expositions and comparisons of Nested Sampling have highlighted this by using inner kernels based on simple constrained random walks (RW) (e.g., Salomone et al. (2024)).

To illustrate the benefit of our chosen inner kernel, we contrast the performance of such a constrained RW sampler with our **Hit-and-Run Slice Sampling (HRSS) implementation detailed in Algorithm 2**. The RW implementation mirrors the adaptive approach used for SMC in Section 2.3: proposals are drawn from a multivariate Gaussian whose covariance is tuned based on the live particle set and scaled by $2.38^2/d$, but proposed steps are rejected if they fall outside the energy constraint $E(x') < E_{\min}$. We compare these two inner kernels on a 50D ill-conditioned Gaussian problem. Figure 10 displays the distribution of the number of likelihood/energy evaluations required per successful MCMC step for both methods.

The HRSS kernel (Algorithm 2) is naturally suited to this constrained sampling task. Its 'stepping-out' procedure explicitly seeks the boundaries of the valid slice (defined by both the prior slice level y and the energy constraint $E_{\min}$) along a given direction before drawing a sample from within that valid interval. This avoids the high rejection rates encountered by the RW kernel near the energy boundary. As shown in Figure 10, the RW approach frequently proposes steps that are rejected, leading to a long tail in the distribution of evaluations per step and wasted computations. Empirically, the NSS algorithm using HRSS significantly outperforms the RW variant, achieving lower runtimes (over an order of magnitude faster on this problem) and more accurate $\ln Z$ estimates. The poor and highly variable performance of the constrained RW (indicated by the long tail) seriously hinders efficient vectorization (Section D.3).

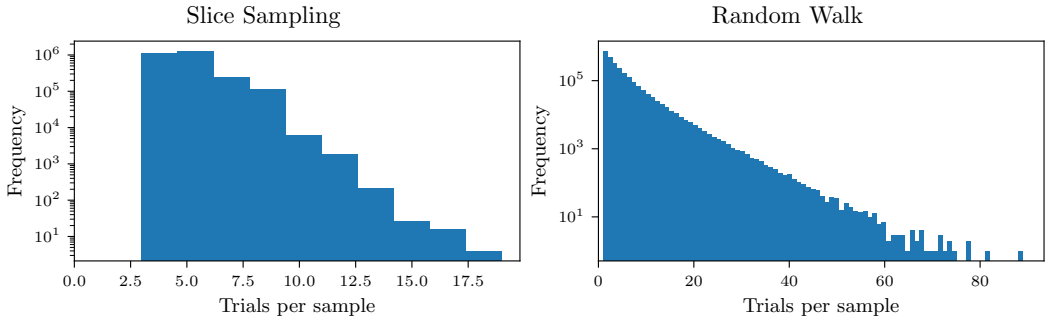

Figure 10: Comparison of efficiency of the inner kernel for Slice Sampling [left] and Random Walks [right], evaluated on a 50D ill-conditioned Gaussian problem under the NS energy constraint. Plotted is the histogram of energy evaluations per accepted step.

F.1 Gradient-Based Sampling

A topic of considerable interest is how to incorporate gradient information from the energy function, $\nabla E(x)$, (where available) to improve sampling efficiency within Nested Sampling, particularly in high-dimensional problems (Lemos et al., 2023; Feroz & Skilling, 2013). Developing a robust method to leverage gradients effectively would be transformative. While our modular implementation allows exploring different inner kernels, successfully integrating gradient information into the NS process remains an open challenge, arguably a key limitation compared to gradient-based methods applied to unconstrained posteriors.

The core difficulty stems from the NS sampling task itself (Figure 1). The target distribution within the constraint $E(x) < E_{\min}$ is the (often uninformative) prior $\Pi(x)$, yet the useful gradient $\nabla E(x)$ describes the landscape of the energy function defining the constraint boundary. Standard gradient-based MCMC proposals (like HMC or MALA trajectories) do not inherently respect this hard energy boundary.

Existing approaches often draw inspiration from Neal's suggestions for slice sampling (Neal, 2003), such as using reflections based on the energy gradient at the boundary defined by $E(x) = E_{\min}$. However, this construction faces practical hurdles: accurate reflection requires precise boundary identification and handling, which is computationally complex and difficult to implement robustly, especially for curved boundaries. Furthermore, hitting the boundary exactly is a rare occurrence in continuous space. A potentially more fruitful path might involve generalizing ideas from methods like (Cai et al., 2022), perhaps using proximal gradient terms or softened constraints within a Langevin-guided proposal mechanism designed to better navigate near the hard energy boundary. Currently, however, no clear, generally successful design pattern for gradient-based NS has emerged. Despite this, the performance of gradient-free NSS on challenging black-box problems remains noteworthy.

F.2 Comparison with NDE-Augmented MCMC Methods

The NSS method proposed here acquires information about the target distribution via evaluations of the prior $\Pi(x)$ and energy $E(x)$ during its slice sampling steps (Algorithm 2). Adaptation occurs through relatively simple information compression: using the global covariance of the live points $\{x_i\}^m$ to update the conditioning matrix M for the Gaussian direction proposal.

A prominent alternative approach involves augmenting Markov Chain Monte Carlo (MCMC) methods with modern Neural Density Estimators (NDEs) (Papamakarios et al., 2021a). We benchmark NSS against FlowMC (Wong et al., 2023a; Gabrié et al., 2022), which exemplifies this paradigm. FlowMC primarily *acquires information* about the target posterior distribution through its Metropolis-Adjusted Langevin Algorithm (MALA) (Roberts & Tweedie, 1996) steps. These steps explore the local probability landscape using evaluations of the posterior density and its gradient, accepting or rejecting moves based on the Metropolis-Hastings criterion. The key role of the NDE (specifically, a Rational Quadratic Spline Normalizing Flow (Durkan et al., 2019)) in FlowMC is *information compression*

Table 5: Comparison with FlowMC on the 10D Gaussian mixture problem.

Algorithm	Energy evals	ESS	Time (s)	MMD	$\ln Z$	ESS /1k eval	ESS /second
Gaussian Mixture							
Ground Truth	-	-	-	-	-46.05	-	-
NSS	3837175	8260	3.6	0.077	-45.92 ± 0.16	2.2	2.3×10^3
flowMC $\epsilon = 0.1$	3200000	1520.14	20	0.52	-	0.48	75
flowMC $\epsilon = 1.0$	3200000	2701.55	17	0.051	-	0.84	1.6×10^2
flowMC $\epsilon = 10.0$	3200000	1768.62	17	0.57	-	0.55	1×10^2

and utilization: it learns an approximation of the target density based on the history of accepted MCMC samples. This learned NDE is then used to generate more efficient global proposals for subsequent MALA steps, aiming to accelerate the information acquisition process.

We compare NSS and FlowMC on the 10D Gaussian Mixture problem (Section 4.2), with results in Figure 11 and Table 5. FlowMC was run with mostly default settings, however we favored aligning the method closer to our NSS/SMC experiments, taking large numbers of short chains. To this end we used 1000 parallel chains with a length of $3 \times d$ as used in our NSS/SMC implementation. The MALA step size of FlowMC ϵ was varied. The reported FlowMC ESS reflects the effective number of independent posterior samples based on MCMC diagnostics.

Key observations from Table 5:

- **Evidence Estimation:** FlowMC, as an MCMC method, samples the normalized posterior but *does not estimate the marginal likelihood* $\ln Z$. NSS directly targets and estimates this quantity.
- **Posterior Sampling Quality (MMD):** Properly tuned FlowMC ($\epsilon = 1.0$) achieves posterior sample quality (MMD) comparable to NSS.
- **Efficiency:** Despite the NDE aiming to improve proposal efficiency, NSS demonstrates significantly higher ESS per evaluation and per second, and a much lower overall runtime on this benchmark. FlowMC incurs computational overhead for NDE training/evaluation and requires careful tuning of its MALA step size ϵ.

This comparison highlights different strategies: NSS uses a simpler adaptation mechanism but relies on an efficient, gradient-free inner kernel (HRSS) suitable for the NS task, directly yielding $\ln Z$. FlowMC uses sophisticated NDEs to accelerate gradient-based MCMC exploration but doesn't estimate $\ln Z$ and introduces overheads. While NSS performs strongly here, incorporating more advanced NDE proposals (beyond global covariance) into the NSS framework itself remains a promising direction for future work, potentially offering a different balance of information acquisition and compression tailored to the NS objective.

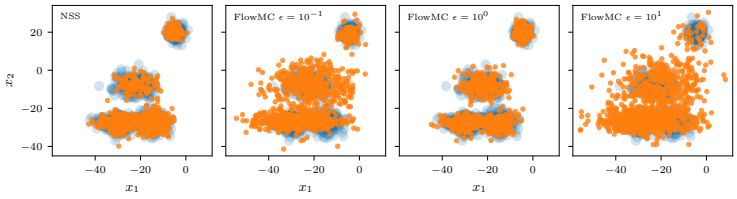

Figure 11: Comparison of different hyperparameter choices of an adaptive MCMC algorithm (augmented with Normalizing Flows) to NSS on a mixture Gaussian problem.

F.3 Comparison with Score-based Diffusion Samplers

Another avenue for incorporating NDE advances into inverse problem solving uses generative models, trained with variational methods, to learn to directly approximate the target distribution.

Table 6: Comparison with (underdamped) Diffusion Bridge Sampling (DBS) on the 10D Gaussian mixture problem. * Note this is not the fairest measure of ESS as one can drawn arbitrarily many samples from the diffusion process, but this is the number of samples used to estimate the MMD.

Algorithm	Energy evals	Time (s)	MMD	$\ln Z$
Gaussian Mixture				
Ground Truth	-	-	-	-46.05
NSS	3837175	3.5	0.077	-45.92 ± 0.16
DBS	65536000	255	0.28	-

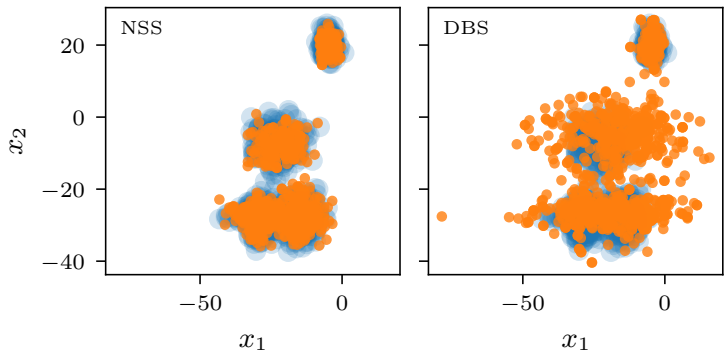

Figure 12: Comparison of NSS with Diffusion Bridge Sampling (DBS) on the 10D mixture Gaussian problem.

Contemporary approaches use methods common to score-based generative models, to train a network that can simulate approximate posterior samples, e.g. Controlled Monte Carlo Diffusions (Chen et al., 2025). The free parameters trained in a variational approach are the weights and biases of the network parameterizing the drift of the Stochastic Differential Equation (SDE) that generates the samples.

We compare NSS to Diffusion Bridge Sampling (DBS) (Richter & Berner, 2024) particularly using its underdamped realisation (Blessing et al., 2025). DBS aims to learn a stochastic differential equation that transports samples from a simple reference distribution to the potentially complicated target distribution $P(x)$. *Information acquisition* in DBS occurs during the training phase, requiring repeated evaluations of the *score function*, i.e., the gradient of the log-target density ($\nabla_x \log P(x)$), to guide the learned diffusion process. This gradient information, evaluated potentially many times across simulated trajectories, informs the model about the shape and high-density regions of the target. The acquired information is naturally *compressed* into the parameters of the Neural Network that approximates the score or drift function of the diffusion process. Once trained, generating samples involves simulating the learned SDE.

We applied DBS to the same 10D randomized Gaussian Mixture problem from Section 4.2, with results shown in Figure 12 and Table 6. We note that DBS and related methods can provide estimates of the normalizing constant, but this is typically relative to the reference (Gaussian) distribution, rather than the actual Prior as would be needed for Bayesian Model Comparison for example. The comparison (Table 6) highlights efficiency concerns for such approaches on this problem: DBS required significantly more computational effort (evaluations and walltime, imposed by the training cost), and questions remain regarding how favorably this cost scales with dimension. In this instance, the increased cost also resulted in a poorer posterior approximation (higher MMD) compared to NSS. While NDE-based generative methods can be powerful function approximators, their comparative performance against established methods like NSS appears problem-dependent, especially when considering both posterior accuracy and the primary goal of Bayesian evidence estimation. Although this comparison doesn't capture the utility of the trained approximate density provided by DBS, it would be similarly possible to use information acquired by NSS (the posterior samples) within a standard generative modelling framework to compress the learned representation, leaning on the established success of such generative methods (Song et al., 2020).