Markovian Flow Matching: Accelerating MCMC with Continuous Normalizing Flows

Alberto Cabezas*
Department of Mathematics and Statistics
Lancaster University, UK
a.cabezasgonzalez@lancaster.ac.uk

Louis Sharrock*
Department of Mathematics and Statistics
Lancaster University, UK
l.sharrock@lancaster.ac.uk

Christopher Nemeth
Department of Mathematics and Statistics
Lancaster University, UK
c.nemeth@lancaster.ac.uk

Abstract

Continuous normalizing flows (CNFs) learn the probability path between a reference distribution and a target distribution by modeling the vector field generating said path using neural networks. Recently, Lipman et al. [45] introduced a simple and inexpensive method for training CNFs in generative modeling, termed flow matching (FM). In this paper, we repurpose this method for probabilistic inference by incorporating Markovian sampling methods in evaluating the FM objective, and using the learned CNF to improve Monte Carlo sampling. Specifically, we propose an adaptive Markov chain Monte Carlo (MCMC) algorithm, which combines a local Markov transition kernel with a non-local, flow-informed transition kernel, defined using a CNF. This CNF is adapted on-the-fly using samples from the Markov chain, which are used to specify the probability path for the FM objective. Our method also includes an adaptive tempering mechanism that allows the discovery of multiple modes in the target distribution. Under mild assumptions, we establish convergence of our method to a local optimum of the FM objective. We then benchmark our approach on several synthetic and real-world examples, achieving similar performance to other state-of-the-art methods, but often at a significantly lower computational cost.

1 Introduction

The task of sampling from a probability distribution known only up to a normalization constant is a fundamental problem arising in a wide variety of fields, including statistical physics [51], Bayesian inference [25], and molecular dynamics [43]. In particular, let $\pi(\mathrm{d}x)$ be a target probability distribution on $\mathbb{R}^d$ with density $\pi(x)$ with respect to the Lebesgue measure of the form[1]

$$\pi(x) = \frac{\hat{\pi}(x)}{Z}, \tag{1}$$

where $\hat{\pi} : \mathbb{R}^d \to \mathbb{R}_+$ is a continuously differentiable function which can be evaluated pointwise, and $Z = \int_{\mathbb{R}^d} \hat{\pi}(x)\mathrm{d}x$ is an unknown normalizing constant. We are interested in generating samples from the target distribution π in order to approximate integrals of the form $\pi[f] = \mathbb{E}_\pi[f(x)]$, where $f : \mathbb{R}^d \to \mathbb{R}$.

*Equal contribution

[1] In a slight abuse of notation, we use π to denote both the target distribution and its density.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-3316

A standard solution to this problem is Markov chain Monte Carlo (MCMC) [12, 64], which relies on the construction of a Markov process which admits the target π as its invariant distribution. One of the most broadly applicable and widely studied MCMC methods is the Metropolis-Hastings (MH) algorithm [32], which proceeds in two steps. First, given a current sample x, a new sample y is proposed according to some proposal distribution $q(\cdot|x)$. Then, this sample is accepted with probability $\alpha(x, y) = \min\left\{1, \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}\right\}$. This strategy generates a Markov chain with the desired stationary distribution and, under mild conditions on the proposal and the target, also ensures that the Markov chain is ergodic [65]. However, for high-dimensional, multi-modal settings, such methods can easily get stuck in local modes, and suffer from very slow mixing times [e.g., 48].

Naturally, the choice of proposal distribution $q(\cdot|x)$ is critical to ensuring that MH MCMC algorithms explore the target distribution within a reasonable number of iterations. A key goal is to obtain proposal distributions with fast mixing times, which can be applied generically to any target distribution. This is particularly challenging in the face of complex, multi-modal (or metastable) distributions, which commonly arise in applications such as genetics [38], protein folding [41], astrophysics [22], and sensor network localization [37]. On the one hand, local proposals, such as those employed in the Metropolis-Adjusted Langevin Algorithm (MALA) [66] or Hamiltonian Monte Carlo (HMC) [20, 56] struggle to transition between regions of high-probability, resulting in very long decorrelation times and few effective independent samples [e.g., 49]. On the other hand, global proposal distributions must be very carefully designed in order to avoid high rejection rates, particularly in high dimensions [17, 47].

Another popular approach to sampling is variational inference (VI) [10, 34, 61, 79], which obtains a parametric approximation $\pi_{\theta^*}(x) \approx \pi(x)$ to the target by minimising the Kullback-Leibler (KL) divergence to the target over a parameterized family of distributions $\mathcal{D}_\theta = \{\pi_\theta : \theta \in \Theta\}$. State-of-the-art VI methods use normalizing flows (NFs), which consist of a sequence of invertible transformations between a reference and a target distribution, to define a flexible variational family [62]. There has also been growing interest in the use of continuous normalizing flows (CNFs), which define a path between distributions using ordinary differential equations [15, 27, 45]. CNFs avoid the need for strong constraints on the flow but, until recently, have been hampered by expensive maximum likelihood training.

In recent years, several works have sought hybrid methods which utilize NFs to enhance the performance of MCMC algorithms; see, e.g., [28] for a recent review. For example, NFs have been successfully used to precondition complex Bayesian posteriors, significantly improving the performance of existing MCMC methods [e.g., 33, 39, 59, 68]. The synergy between local MCMC proposals and global, flow-informed proposals has also been explored, leading to enhanced mixing rates and effective estimation of multimodal targets [e.g., 24, 67].

Our contributions In this paper, we continue this promising line of work, introducing a new probabilistic inference scheme which integrates CNFs with MCMC sampling techniques. Our approach utilizes flow matching (FM), a scalable, simulation-free training objective for CNFs recently introduced by Lipman et al. [45]. This enables, for the first time, the incorporation of CNFs into an adaptive MCMC algorithm. Concretely, our approach augments a local, gradient-based Markov transition kernel with a non-local, flow-informed transition kernel, defined using a CNF. This CNF, and the corresponding transition kernel, are adapted on-the-fly using samples from the chain, which are used to define the probability path for the FM objective. Our scheme also includes an adaptive tempering mechanism, which is essential for discovering multiple modes in complex target distributions. Under mild assumptions, we establish that the flow-network parameters output by our method converge to a local optimum of the FM objective. We then demonstrate empirically the performance of our approach on several synthetic and real-world examples, illustrating comparable or superior performance to other state-of-the-art sampling methods.

2 Preliminaries

Continuous Normalizing Flows A continuous normalizing flow (CNF) is a continuous-time generative model which is trained to map samples from a base distribution p_0 to a given target distribution [15]. Let v_t be a time-dependent vector field that runs continuously in the unit interval. Under mild conditions, this vector field can be used to construct a time-dependent diffeomorphic map

called a flow $\phi : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$, defined via the ordinary differential equation (ODE):

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_t(x) = v_t(\phi_t(x)), \quad \phi_0(x) = x. \tag{2}$$

Given a reference density $p_0 : \mathbb{R}^d \to \mathbb{R}_+$, and the flow ϕ, we can generate a probability density path $p : [0,1] \times \mathbb{R}^d \to \mathbb{R}_+$ as the pushforward of p_0 under ϕ, viz $p_t := [\phi_t]_\sharp p_0$, for $t \in [0,1]$. This yields, via the instantaneous change-of-variables formula [e.g., 14]

$$\log p_t(x_t) = \log p_0(x) - \int_0^t \nabla \cdot v_s(x_s)\mathrm{d}s, \tag{3}$$

where $x_s := \phi_s(x)$, and where ∇ is the divergence operator, i.e. the trace of the Jacobian matrix. In modern applications, the vector field v_t is often parameterized using a neural network v_t^θ, in which case the ODE in (2) is referred to as a neural ODE [15]. In turn, this yields a deep parametric model ϕ_t^θ for the flow ϕ_t, known as a CNF [27].

Flow Matching One would typically like to learn a CNF which maps between a given reference density p_0 and a target density π. Given samples from the target, one approach is to maximize the log-likelihood $\mathbb{E}_{x\sim\pi}\left[\log p_1^\theta(x)\right]$. In practice, however, maximum likelihood training is very slow as both sampling and likelihood evaluation require multiple network passes to solve the ODE in (2).

Flow Matching (FM) provides an alternative, simulation-free method for training CNFs [45]. Let $p_t(x)$ be a target probability density path such that $p_0 = p$ is a simple reference distribution, and $p_1 \approx \pi$ is approximately equal to the target distribution. Let $v_t(x)$ be a vector field which generates this $p_t(x)$. Then the FM objective for the CNF vector field $v_t^\theta(x)$ is defined as

$$\mathcal{L}(\theta;\pi) = \mathbb{E}_{t\sim\mathcal{U}(0,1)}\mathbb{E}_{x\sim p_t}\left[\|v_t^\theta(x) - v_t(x)\|_2^2\right]. \tag{4}$$

In practice, we do not have direct access to the target vector field, $v_t(x)$, and so we cannot minimize (4) directly. However, as shown in Lipman et al. [45, Theorem 2], it is equivalent to minimize the conditional flow-matching (CFM) loss

$$\mathcal{J}(\theta;\pi) = \mathbb{E}_{t\sim\mathcal{U}(0,1)}\mathbb{E}_{x_1\sim\pi}\mathbb{E}_{x\sim p_t(\cdot|x_1)}\left[||v_t^\theta(x) - v_t(x|x_1)||_2^2\right], \tag{5}$$

where $p_t(\cdot|x_1)$ is a conditional probability density path satisfying $p_0(x|x_1) = p_0$ and $p_1(x|x_1) \approx \delta_{x_1}$, and $v_t(\cdot|x_1) : \mathbb{R}^d \to \mathbb{R}^d$ is a conditional vector field that generates $p_t(\cdot|x_1)$. There are various choices for $p_t(\cdot|x_1)$ and $v_t(\cdot|x_1)$. For simplicity, we here assume that the conditional probability path is Gaussian, viz $p_t(x|x_1) = \mathcal{N}(x|m_t(x_1), s_t(x_1)^2\mathbb{I}_d)$, where $m : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ denotes a time-dependent mean, and $s : [0,1] \times \mathbb{R} \to \mathbb{R}_+$ a time-dependent scalar standard deviation. For our experiments, we further adopt the optimal transport conditional probability path introduced in [45], setting $m_t(x_1) = tx_1$ and $s_t(x_1) = 1 - (1-\sigma_{\min})t$ for some $\sigma_{\min} \ll 1$. In this case, the conditional vector field assumes the particularly simple form $v_t(x|x_1) = \frac{x_1 - (1-\sigma_{\min})x}{1-(1-\sigma_{\min})t}$.

3 Markovian Flow Matching

In this section, we present our main contribution, an adaptive MCMC algorithm which combines a non-local, flow-informed transition kernel trained via FM; a local, gradient-based Markov transition kernel; and an adaptive annealing schedule. We begin by describing how CNFs can be used within a MH MCMC algorithm.

3.1 MCMC with Flow Matching

Suppose, for now, that we have access to a CNF $(\phi_t^\theta)_{t\in[0,1]}$, trained (e.g.) via flow-matching, with corresponding vector field $(v_t^\theta)_{t\in[0,1]}$, which generates a probability path $(p_t^\theta)_{t\in[0,1]}$ between a reference density p_0 and an approximation of the target density π. Given a point $x_0 \in \mathbb{R}^d$ on the reference space, we can evaluate the log-density of the pullback of the target distribution π as

$$\log[\phi_1^\theta]^\sharp\pi(x_0) = \log\pi(\phi_1^\theta(x_0)) - \int_1^0 \nabla \cdot v_t^\theta(\phi_t^\theta(x_0))\mathrm{d}t. \tag{6}$$

https://doi.org/10.52202/079017-3316

Under the assumption that the CNF approximately transports samples from p_0 to π, we expect that $[\phi_1^\theta]_\sharp p_0 \approx \pi$ in the target space, and that $[\phi_1^\theta]^\sharp \pi \approx p_0$ in the reference space. Given that the reference distribution p_0 is chosen such that it is easy to sample from, this suggests the following strategy, sometimes referred to as *neural trasport MCMC* or *neutraMCMC* [28, 33, 44, 59]. First, transform initial positions x_1 from the target space to the reference space by solving

$$\begin{bmatrix} x_0 \\ \log p_1^\theta(x_1) - \log p_0(x_0) \end{bmatrix} = \begin{bmatrix} x_1 \\ 0 \end{bmatrix} + \int_1^0 \begin{bmatrix} v_t^\theta(x_t) \\ -\nabla \cdot v_t^\theta(x_t) \end{bmatrix} \mathrm{d}t, \tag{7}$$

which integrates the combined dynamics of x_t and the log-density of the sample backwards in time. Then, generate MCMC proposals y_0 in the reference space using any standard MCMC scheme which targets the pullback of the target distribution, as defined in (6). Finally, transform accepted proposals back to target space using the forward dynamics, viz

$$\begin{bmatrix} y_1 \\ \log p_1^\theta(y_1) - \log p_0(y_0) \end{bmatrix} = \begin{bmatrix} y_0 \\ 0 \end{bmatrix} + \int_0^1 \begin{bmatrix} v_t^\theta(y_t) \\ -\nabla \cdot v_t^\theta(y_t) \end{bmatrix} \mathrm{d}t. \tag{8}$$

This corresponds to using a transformation-informed proposal in a Markov transition step, an approach which has been successfully applied using (discrete) normalizing flows [28, 33, 44, 59].

There are various possible choices for the proposal distribution on the reference space (see Appendix A). For example, [23] consider an independent MH (IMH) proposal, where i.i.d. samples are drawn from the reference distribution. Here we focus on a flow-informed random-walk, motivated largely by its superior empirical performance in numerical experiments. This proposal performs particularly well on high-dimensional problems, where overfitting of the CNF can be corrected with stochastic steps, while exacerbated by independent proposals [39]. Concretely, our flow-informed random-walk transition kernel, summarized in Algorithm 2 (see Appendix A), can be written as

$$P(x, \mathrm{d}y; \pi, \theta) = \alpha(x, y)\rho_\theta(\mathrm{d}y|x) + (1 - b(x))\delta_x(\mathrm{d}y), \tag{9}$$

where $\rho_\theta(\mathrm{d}y|x)$ is the distribution defined by the transition

$$x_0 = x + \int_1^0 v_t^\theta(\phi_t^\theta(x))\mathrm{d}t, \quad y_0 \sim \mathcal{N}(x_0, \sigma^2_{\mathrm{opt}}), \quad y = y_0 + \int_0^1 v_t^\theta(\phi_t^\theta(y_0))\mathrm{d}t, \tag{10}$$

and $\alpha(x, y) = \min\left\{1, \frac{\pi(y)\rho_\theta(x|y)}{\pi(x)\rho_\theta(y|x)}\right\}$ and $b(x) = \int_{\mathbb{R}^d} \alpha(x, y)\rho_\theta(\mathrm{d}y|x)$.

Training the CNF Thus far, we have assumed that it is possible to train a CNF which maps samples from the reference distribution p_0 to (an approximation of) the target distribution π. Clearly, however, the CFM objective is not immediately applicable in our setting, since we do not have access to samples from the target π.

Tong et al. [72] propose two alternatives in this case: (i) use an importance sampling reweighted objective function, or (ii) use samples from a long-run MCMC algorithm (e.g., MALA) as approximate target samples. Both of these approaches, however, have limitations. The former is unlikely to succeed when the proposal distribution differs significantly from the target distribution, while the latter will only perform well when the chosen MCMC method mixes well.

In this paper, we adopt a different approach, updating the parameters of a CNF based on a dynamic estimate of the CFM objective obtained via an adaptive MCMC algorithm. This is similar in spirit to other recent flow-informed MCMC algorithms [24, 35, 67], and the *Markovian score climbing* algorithm in [54].

3.2 Adaptive MCMC with Flow Matching

Overview Our adaptive MCMC scheme combines a non-local, flow-informed transition kernel (e.g., a flow informed random-walk) and a local transition kernel (e.g., MALA), which generate new samples from a sequence of annealed target distributions. These new samples are used to define a new estimate of the CFM objective in (5), which is optimized to define a new CNF. These steps are repeated until the samples converge in distribution to the target π, and the flow-network parameters converge to a local minima of the flow matching objective (see Proposition 3.1). This scheme, which we refer to as *Markovian Flow Matching* (MFM), is summarized in Algorithm 1.

Sampling There is significant freedom regarding the choice of both the local and the non-local MCMC algorithms. In our experiments, we adopt the Metropolis-Adjusted Langevin Algorithm (MALA) as the local algorithm. Thus, the local Markov kernel Q is given by

$$Q(x, \mathrm{d}y; \pi) = \alpha(x, y)q(\mathrm{d}y|x) + (1 - b(x))\delta_x(\mathrm{d}y), \tag{11}$$

where $q(\mathrm{d}y|x)$ is given by

$$q(\mathrm{d}y|x) \propto \exp\left(-\frac{1}{4\tau}\|y - x - \tau\nabla \log \pi(x)\|^2\right) \mathrm{d}y, \tag{12}$$

and where, as elsewhere, $\alpha(x, y) = \min\left\{1, \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}\right\}$ and $b(x) = \int_{\mathbb{R}^d} \alpha(x, y)q(\mathrm{d}y|x)$. In principle, however, other choices such as HMC could also be used.

Meanwhile, for the non-local MCMC algorithm, we adopt the flow-informed random walk with non-local Markov kernel P defined in (9). Together, assuming alternate local and non-local steps, these two kernels define a Markov chain with Markov transition kernel $R := P \circ Q$, given explicitly by $R(x, \mathrm{d}y; \pi, \theta) = \int_{\mathcal{Z}} Q(x, \mathrm{d}z; \pi)P(z, \mathrm{d}y; \pi, \theta)$. In practice, the balance between local and non-local moves is controlled by the hyperparameter k_Q, which sets the number of local steps before a global step.

Training Following each MCMC step, the parameters of the flow-informed Markov transition kernel P are updated based on a new estimate of $\mathcal{J}(\theta; \pi)$. To be precise, suppose we write $\mu_t := \mu_0 R^k(\cdot, \cdot; \pi, \theta)$ for the distribution of the Markov chain with kernel $R(\cdot, \cdot; \pi, \theta)$ after $k \in \mathbb{N}$ steps, starting from initialization μ_0, where $R^k = R \circ R \cdots \circ R$. Our objective function is then given by

$$\mathcal{J}(\theta; \mu_k) = \mathbb{E}_{t\sim\mathcal{U}(0,1)}\mathbb{E}_{x_1\sim\mu_k}\mathbb{E}_{x\sim p_t(\cdot|x_1)}\left[||v_t^\theta(x) - v_t(x|x_1)||_2^2\right]. \tag{13}$$

For our choice of conditional probability path (i.e., the optimal transport path), we can in fact rewrite this objective as [45, Section 4.1]

$$\mathcal{J}(\theta; \mu_k, \sigma_{\min}) = \mathbb{E}_{t\sim\mathcal{U}(0,1)}\mathbb{E}_{x_1\sim\mu_k}\mathbb{E}_{x_0\sim p_0}\left[\left|\left|v_t^\theta(\phi_t(x_0|x_1)) - v_t(\phi_t(x_0|x_1)|x_1)\right|\right|_2^2\right], \tag{14}$$

where $v_t(x|x_1) = \frac{x_1 - (1-\sigma_{\min})x}{1-(1-\sigma_{\min})t}$ and $\phi_t(x|x_1) = (1 - (1 - \sigma_{\min}t)x + tx_1$. In practice, we will optimize a Monte Carlo estimate of this objective, namely,

$$\mathcal{J}(\theta; \{x^i(k)\}_{i=1}^N, \sigma_{\min}) = \frac{1}{N}\sum_{i=1}^N \left|\left|v_{t_i}^\theta(\phi_{t_i}(x_0^i|x^i(k))) - v_{t_i}(\phi_{t_i}(x_0^i|x^i(k))|x^i(k))\right|\right|_2^2, \tag{15}$$

where $\{x^i(k)\}_{i=1}^N$ are the samples from N chains of our MCMC algorithm after $k \in \mathbb{N}$ iterations, $x_0^i \overset{\text{i.i.d.}}{\sim} p_0$, and $t_i \sim \mathcal{U}(0, 1)$. The use of N particles allow the state's mutation to N computing cores running in parallel at each iteration. Sampling steps can be run in parallel using modern vector-oriented libraries, before each particle is used to approximate the loss and update the parameters. Thus, the speedup gained by using more than one core scales linearly with the number of cores as long as there are as many cores as there are particles.

Annealing For complex (e.g., multimodal) target distributions, it can be challenging to learn a CNF that successfully maps between the reference p_0 and the target π. For example, if the locations of the modes of the target are not known a priori, and the MCMC chains are initialized far from one or more of the modes, it is unlikely that the local MCMC kernel, and therefore the trained flow, will ever discover these modes [e.g., 24, Section IV.C]. To alleviate this problem, one approach is to iteratively target a sequence of annealed densities $\{\pi_k(x)\}_{k=0:K}$, which smoothly interpolate between a simple base distribution $\pi_0(x)$ (e.g., a standard Gaussian), and the target distribution $\pi_K(x) := \pi(x)$. This idea is central to other Monte Carlo sampling methods such as Sequential Monte Carlo (SMC) [18] and Annealed Importance Sampling (AIS) [55], as well as sampling methods used in score-based generative modelling [e.g., 70]. In our case, the annealed targets act as intermediary steps within the flow-informed MCMC scheme.

A standard way in which to construct the sequence $\{\pi_k(x)\}_{k=0:K}$ is to use a geometric interpolation, defining

$$\pi_k(x) = \pi_K(x)^{\beta_k}\pi_0(x)^{1-\beta_k}, \tag{16}$$

https://doi.org/10.52202/079017-3316

Algorithm 1 Markovian Flow Matching

Input: target π, base π_0, vector field v_t^θ, reference p_0, concentration σ_{min}, initial parameters θ_0, target ESS fraction α, iterations K, number of particles N, local kernel Q, flow-informed kernel P, MCMC steps per resampling step k_Q, step sizes $\varepsilon_{1:K}$.
Output: flow-network parameters θ_K
Sample $x_0^i \sim \pi_0$ for $i = 1, \ldots, N$ (*initialize samples*)
for $k = 1 : K$ **do**
  if $\beta_{k-1} < 1$ **then**
    $\beta_k =$ solve (17) for $\beta_{k-1} < \beta \leq 1$ (*update annealing temperature*)
    $\pi_k =$ solve (16) (*update annealing density*)
  end if
  if $k \mod k_Q + 1 = 0$ **then**
    $x_k^i \sim P(x_{k-1}^i, \cdot; \pi_k, \theta_{k-1})$ for $i = 1, \ldots, N$ (*flow-informed Markov transition*).
  else
    $x_k^i \sim Q(x_{k-1}^i, \cdot; \pi_k)$ for $i = 1, \ldots, N$ (*local Markov transition*).
  end if
  $\theta_k = \theta_{k-1} + \varepsilon_k \nabla_\theta \mathcal{J}(\theta_{k-1} | \{x_k^i\}_{i=1}^N, \sigma_{\min})$ (*update flow-network parameters*)
end for

where $\beta_{0:K}$ is a sequence of temperatures which satisfies $0 = \beta_0 < \beta_1 < \cdots < \beta_K = 1$ [e.g., 55]. In practice, it can be difficult to choose a good sequence of temperatures that provides a smooth transition between densities. One heuristic for adaptively setting this sequence is based on the effective sample size (ESS). In particular, by setting the ESS to a user-specified percentage α of the number of particles N, the next temperature β_k in the schedule can be determined by solving the recursive equation [9]

$$\beta_k = \inf \left\{ \beta_{k-1} < \beta \leq 1 : \frac{\left[\frac{1}{N} \sum_{i=1}^N w_i^{\beta_{k-1}}(\beta) \right]^2}{\frac{1}{N} \sum_{i=1}^N w_i^{\beta_{k-1}}(\beta)^2} = \alpha \right\}, \tag{17}$$

where $w_i^{\beta_{k-1}}(\beta) = \left[\pi_K(x^i)^\beta \pi_0(x^i)^{1-\beta} \right] / \left[\pi_K(x^i)^{\beta_{k-1}} \pi_0(x^i)^{1-\beta_{k-1}} \right] = \left[\pi_K(x^i) / \pi_0(x^i) \right]^{\beta - \beta_{k-1}}$ are new importance weights given the current temperature β_{k-1}. In practice, we find that the inclusion of this adaptive tempering scheme is essential in the presence of highly multimodal target distributions, enabling the discovery of modes which are not known *a priori*.

Convergence The output of Algorithm 1 is a vector of parameters θ_K which defines a CNF $(\phi_t^{\theta_K})_{t \in [0,1]}$. Under the assumption that the parameter estimate converges, that is, $\theta_K \to \theta^*_{\text{global}}$ as $K \to \infty$, where $\theta^*_{\text{global}} = \arg\min_{\theta \in \Theta} \mathcal{J}(\theta; \pi)$ is the global minimizer of the CFM objective $\mathcal{J}(\theta; \pi)$ in (5), this CNF is guaranteed to generate a probability path $(p_t^\theta)_{t \in [0,1]}$ which transports samples from the reference p_0 to the true target π [e.g., 45].

In practice, the objective $\mathcal{J}(\theta; \pi)$ is highly non-convex, and thus it is not possible to establish a convergence result of this type without imposing unreasonably strong assumptions on the vector field $(v_t^\theta)_{t \in [0,1]}$. This being said, it is reasonable to ask whether θ_K converges to a local optimum of the CFM objective. We now answer this question in the affirmative. In particular, under mild regularity conditions, Proposition 3.1 guarantees that $\theta_K \to \theta^*$ almost surely as $K \to \infty$, where θ^* denotes a local minimum of the CFM objective. This proposition closely mirrors [54, Proposition 1]. Its proof, which relies on a classical result in [6, Theorem 3.17], is provided in Appendix B.

Proposition 3.1. *Assume that Assumptions B.1 - B.6 hold (see Appendix B). Assume also that $(\theta_K)_{K \in \mathbb{N}}$ is a bounded sequence, which almost surely visits a compact subset of the domain of attraction of θ^* infinitely often. Then $\theta_K \to \theta^*$ almost surely.*

4 Related work

In recent years, a number of works have proposed algorithms which combine MCMC techniques with NFs; see, e.g., [2, 28] for recent surveys. Broadly speaking, these algorithms fall into two distinct categories. *NeutraMCMC* methods leverage NFs as reparameterization maps which simplify

the geometry of the target distribution, before running (local) MCMC samplers in the latent space. This technique was pioneered in the seminal paper [59], and since been investigated in a number of different works [e.g., 13, 33, 44, 54, 58, 68, 82, 84]. *Flow MCMC* methods, meanwhile, utilize the pushforward of the base distribution through the NF as an (independent) proposal within an MCMC scheme. This approach was first studied by [3], and further extended in [4, 31, 57].

More recently, [24, 67] have introduced adaptive MCMC schemes which combine local MCMC samplers (e.g., MALA or HMC), with a non-local, flow-informed proposal (IMH or i-SIR); see also [35]. Our algorithm combines aspects of both *neutraMCMC* and *flow MCMC* methods and, unlike any existing approach, make use of a CNF (as opposed to a discrete NF), by leveraging the conditional flow matching objective. The use of NFs within other Monte Carlo algorithms has also been the subject of recent interest. For example, [5, 50] consider augmenting SMC with NFs, while [19, 52] use NFs (or diffusion models) within AIS.

Although less directly comparable to our own approach, several other recent works have proposed to use (controlled) diffusion processes to sample from unnormalized probability distributions. Such works include Zhang and Chen [85], who introduce the *path integral sampler*, Vargas et al. [75], who propose the *denoising diffusion sampler*, and Zhang et al. [83], who introduce *generative flow samplers*. Some other relevant contributions in this direction include [1, 8, 16, 60, 63, 69, 73, 74, 75, 76, 77, 78].

5 Experiments

In this section, we evaluate the performance of MFM (Algorithm 1) on two synthetic and two real data examples. Our method is benchmarked against four relevant methods. The Denoising Diffusion Sampler [DDS; 75] is a VI method which approximates the reversed diffusion process from a reference distribution to an extended target distribution by minimizing the KL divergence. Adaptive Monte Carlo with Normalizing Flows [NF-MCMC; 24] is an augmented MCMC scheme which uses a mixture of MALA and adaptive transition kernels learned using discrete NFs. Flow Annealed Importance Sampling Bootstrap [FAB; 52] is an augmented AIS scheme minimizing the mass-covering α-divergence with $\alpha = 2$. Finally, Adaptive Tempered SMC (AT-SMC), i.e. the SMC algorithm described in [18] using a MALA transition kernel and a sequence of annealed distributions chosen adaptively by solving (17).

For each experiment, all MALA kernels use the same step size, targeting an acceptance rate of close to 1 since we estimate expectations, e.g. in (14), using the current ensemble of particles, rather than a single long chain. Following [85], we parameterize the vector field as

$$\mathrm{NN}^*(t;\theta_3)v_t^\theta(x) = \mathrm{NN}(x,t;\theta_1) + \mathrm{NN}(t;\theta_2) \times \nabla \log \pi(x), \tag{18}$$

where the neural networks are standard MLPs with 2 hidden layers, using a Fourier feature augmentation for t [71], and where NN^* outputs a real value that reweights the vector field output using the time component. This architecture is also used by DDS [75, Section 4]. Meanwhile, FAB and NF-MCMC use rational quadratic splines [21]. Flows are trained using Adam [40] with a linear decay schedule terminating at $\varepsilon_K = 0$. We report results for all methods averaged over 10 independent runs with varying random seeds. Code to reproduce the experiments is provided at https://github.com/albcab/mfm.

5.1 4-mode Gaussian mixture

Our first example is a mixture of four Gaussians, evenly spaced and equally weighted, in two-dimensional space. The four mixture components have means $(8, 8)$, $(-8, 8)$, $(8, -8)$, $(-8, -8)$, and all have identity covariance. This ensures that the modes are sufficiently separated to mean that jumping between modes requires trajectories over sets with close to null probability. Given the synthetic nature of the problem, we can measure approximation quality using the Maximum Mean Discrepancy (MMD) [e.g., 29]; see Appendix C.1 for details. We can also include, as a benchmark, the results for an approximation learned using FM with true target samples. Diagnostics for all models are presented in Table 1, and learned flow samples in Figure 1. Further algorithmic details and results are provided in Appendix C.2.

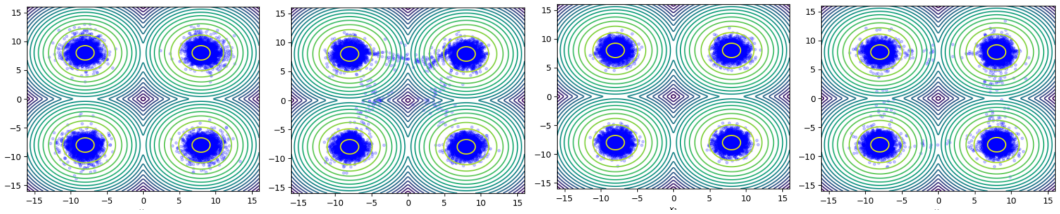
In this experiment, only our method (Figure 1a) and DDS (Figure 1c) learn the fully separated modes, reflecting the greater expressivity of CNFs in comparison to the discrete NFs used in, e.g., NF-MCMC

 https://doi.org/10.52202/079017-3316

	4-mode		16-mode	
	MMD	seconds	MMD	seconds
FM w/ π samples	3.69e-4±1.84e-4	22.3 ± 0.64	1.35e-3±6.66e-4	22.4 ± 1.01
MFM $k_Q = K$	2.37e-3±2.29e-3	27.9 ± 1.27	1.88e-2±3.67e-3	28.2 ± 2.84
MFM $k_Q = 10$	8.13e-4±4.41e-4	$117. \pm 5.65$	2.87e-3±9.67e-4	89.6 ± 5.19
DDS	1.76e-4±2.32e-4	$114. \pm 0.68$	1.02e-1±4.10e-2	$115. \pm 0.64$
NF-MCMC	5.85e-3±3.91e-3	72.0 ± 11.7	8.05e-3±1.42e-2	67.0 ± 12.3
FAB	2.69e-4±2.06e-4	$101. \pm 3.24$	1.51e-3±1.06e-3	$102. \pm 4.32$
AT-SMC	3.95e-2±2.90e-2	2.18 ± 0.26	1.73e-2±5.30e-3	2.19 ± 0.21

Table 1: Diagnostics for the two synthetic examples. MMD is the Maximum Mean Discrepancy between real samples from the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

(Figure 1d). It is worth noting that DDS provides a closer approximation to the real target than MFM and, notably, even FM trained using true target samples (top row). Given that both methods use the same network architecture but a different learning objective, this suggests a potential limitation with the FM objective, at least when using this network architecture. This being said, MFM is notably more efficient than DDS (as well as the other methods) in terms of total computation time. While this is not a critical consideration in this synthetic, low-dimensional setting, it is a significant advantage of MFM in higher-dimensional settings involving real data (e.g., Section 5.3 and Section 5.4).



(a) MFM $k_Q = 10^2$ (b) FAB (c) DDS (d) NF-MCMC

Figure 1: Comparison between MFM, FAB, DDS, and NF-MCMC. Samples from the target density for the 4-mode Gaussian mixture example.

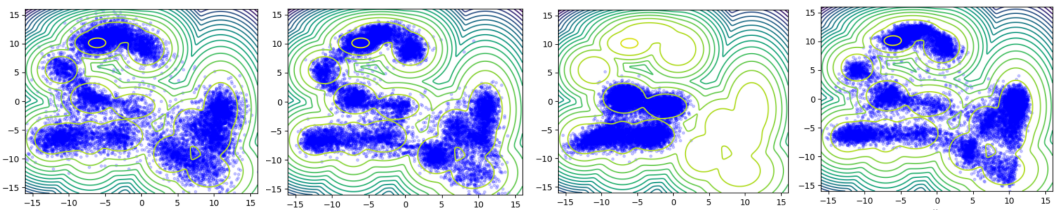
5.2 16-mode Gaussian mixture

The second experiment is a mixture of bivariate Gaussians with 16 mixture components. This is a modification of the 4-mode example, with contrasting qualities that illustrate other characteristics of each of the presented methods. In this case, the modes are evenly distributed on $[-16, 16]^2$, with random log-normal variances. The number of modes reduces the size of sets of (near) null probability between the modes, making jumping between them easier. To increase the difficulty of this model, all methods are initialized on a concentrated region of the sampling space. Diagnostics are presented in Table 1 and learned flow samples in Figure 2. Further details are provided in Appendix C.3.

In this example, DDS collapses to the modes closest to the initial positions while our method captures the whole target. Since the modes are no longer separated by areas of near-zero probability, the discrete NF methods are now able to accurately capture the target density. In this case, FAB marginally outperforms MFM as measured by the MMD, but this slight improvement in performance comes at the cost of a much higher run-time.

5.3 Field system

Our first real-world example considers the stochastic Allen–Cahn model [7], used as a benchmark in [24], and described in Appendix C.5. This fundamental reaction-diffusion equation is central to the study of phase transitions in condensed matter systems. Incorporating random forcing terms or thermal fluctuations allows for a stochastic treatment of the dynamics, capturing the inherent randomness and uncertainties in physical systems. This model leads to a discretized target density



(a) MFM $k_Q = 10^2$ (b) FAB (c) DDS (d) NF-MCMC

Figure 2: Comparison between MFM, FAB, DDS, and NF-MCMC. Samples from the target density for the 16-mode Gaussian mixture example.

which takes the form

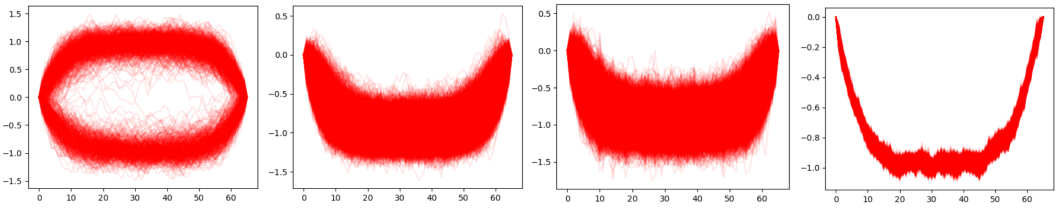
$$\log \pi(x) = -\beta\left(\frac{a}{2\Delta s}\sum_{i=1}^{d+1}(x_i - x_{i-1})^2 + \frac{b\Delta s}{4}\sum_{i=1}^{d}(1 - x_i^2)^2\right), \tag{19}$$

with $\Delta s = \frac{1}{d}$, and boundary conditions $x_0 = x_{d+1} = 0$. In our experiments, we take $d = 64$. Meanwhile, following [24], other parameter values are chosen to ensure bimodality at $x = \pm 1$: $a = 0.1$, $b = 1/a = 10$, and $\beta = 20$. The bimodality induced by the two global minima complicates mixing when using traditional MCMC updates. Learning the global geometry of the target and using that information to propose transitions facilitates movement between modes. Unlike previous work [e.g., 24], we deliberately choose not to employ an informed base measure. Instead, we opt for a standard Gaussian with no additional information, making the problem significantly more challenging. This choice illustrates the robustness of our approach.

Numerical diagnostics for each method are presented in Table 2. In this case, we use the Kernelized Stein Discrepancy (KSD) as a measure of sample quality [e.g., 26, 46]; see Appendix C.1 for details. While this is not a perfect metric, it does allow us to qualitatively compare the different methods considered.

In this case, the tempering mechanism of our method is crucial for ensuring that the learned flow does not collapse on one of the modes and instead explores both global minima. This is confirmed when plotting the samples generated in the grid in Figure 3. This experiment demonstrates the ability of our method to capture complex multi-modal densities, even without an informed base measure, at a *significantly* lower computational cost (e.g., 10-25x faster) than competing methods. Indeed, while FAB was the best performing method in this experiment as measured by the KSD, it failed to capture both of the modes in the target distribution, and required a much greater total computation time (see Table 2).

It is worth noting that MFM (and the other two benchmarks, DDS and FAB) significantly outperformed NF-MCMC in this example, despite the similarities between MFM and NF-MCMC. While we tested various hyperparameter configurations for NF-MCMC, we were not able to find a setting that achieved comparable results in the absence of an informed base measure.



(a) MFM $k_Q = 10^2$ (b) FAB (c) DDS (d) NF-MCMC

Figure 3: Comparison between MFM, FAB, DDS, and NF-MCMC. Representative samples from the target density for the Field system example.

https://doi.org/10.52202/079017-3316

5.4 Log-Gaussian Cox process

Bayesian inference for high-dimensional spatial models is known to be challenging. One such model is the log-Gaussian Cox process (LGCP) introduced in [53], which is used to model the locations of 126 Scots pine saplings in a natural forest in Finland. See Appendix C.6 for full details. The target space is discretized to a $M = 40 \times 40$ regular grid, rendering the target dimension $d = 1600$. In Table 2, we report diagnostics for each algorithm.

In this case, the lack of multimodality in the target makes it a good fit for non-tempered schemes. Similar to the previous example, NF-MCMC is unable to obtain an accurate approximation to the target distribution. We suspect that this may be a result of non-convergence: due to memory issues, it was not possible to run NF-MCMC (or FAB) for more than $K = 10^3$ iterations. This also explains the (relatively) smaller run times of these algorithms in this example. By a small margin, DDS provides the best approximation of the target, slightly outperforming MFM and FAB. Meanwhile, MFM provides a good approximation to the target at a lower computational cost with respect to its competitors.

	Field system			Log-Gaussian Cox		
$K = 10^4$	KSD U-stat.	KSD V-stat.	seconds	KSD U-stat.	KSD V-stat.	seconds
MFM $k_Q = K$	2.61 ± 2.00	20.9 ± 2.49	52.3 ± 1.23	1.13e-1$\pm$.05	28.1 ± 0.24	117 ± 4.19
MFM $k_Q = 10^3$	2.67 ± 2.16	21.0 ± 2.66	53.6 ± 1.33	1.12e-1$\pm$.04	28.1 ± 0.23	143 ± 14.5
DDS	15.2 ± 35.9	18.0 ± 36.9	2400 ± 8.65	7.59e-2$\pm$.02	24.7 ± 0.08	3260 ± 8.41
NF-MCMC	548 ± 325	549 ± 325	2000 ± 15.6	11.8 ± 7.55	89.0 ± 238	215 ± 46.4
FAB	0.14 ± 0.42	1.78 ± 0.42	3880 ± 7.19	1.55e-1$\pm$.06	52.3 ± 2.02	1040 ± 2.78
AT-SMC	1.61 ± 2.33	18.4 ± 2.35	4.13 ± 0.30	1.39e-2$\pm$.01	25.0 ± 0.12	6.11 ± 0.44

Table 2: Diagnostics for the two real data examples. KSD U-stat and V-stat are the Kernel Stein Discrepancy U- and V-statistics between the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

6 Conclusion

Summary. In this paper, we introduced Markovian Flow Matching, a new approach to sampling from unnormalized probability distributions that augments MCMC with CNFs. Our method combines a local Markov kernel with a non-local, flow-informed Markov kernel, which is adaptively learned during sampling using FM. It also incorporates an adaptive tempering mechanism, which allows for the discovery of multiple target modes. Under mild assumptions, we established convergence of the flow network parameters output by our algorithm to a local optimum of the FM objective. We also benchmarked the performance of our algorithm on several examples, illustrating comparable performance to other state-of-the-art methods, often at a fraction of the computational cost.

Limitations and Future Work. We highlight three limitations of our work. First, our theoretical result established convergence of the flow network parameters obtained via MFM to a *local* minimum of the FM objective. Further work is required to understand how well these local minima generalize, in order to accurately quantify how accurately the corresponding CNF captures the target posterior. Second, we did not establish non-asymptotic convergence rates for our method. Finally, since it was not the main focus of this work, we did not explore in great detail other choices of architecture for the flow network. We expect that, for certain targets, this could have a significant impact on the performance of MFM. Indeed, a promising avenue for further research lies in developing tailored CNFs designed for particular posterior distributions. This approach would go beyond the current practice of including the gradient of the log-posterior and instead exploit unique characteristics intrinsic to each model when constructing the flow.

Acknowledgments and Disclosure of Funding

LS and CN were supported by the Engineering and Physical Sciences Research Council (EPSRC), grant number EP/V022636/1. CN acknowledges further support from the EPSRC, grant numbers EP/S00159X/1 and EP/Y028783/1.

References

[1] T. Akhound-Sadegh, J. Rector-Brooks, A. J. Bose, S. Mittal, P. Lemos, C.-H. Liu, M. Sendera, S. Ravanbakhsh, G. Gidel, Y. Bengio, et al. Iterated denoising energy matching for sampling from Boltzmann densities. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. 7

[2] M. S. Albergo and E. Vanden-Eijnden. Learning to sample better. *arXiv preprint arXiv:2310.11232*, 2023. 6

[3] M. S. Albergo, G. Kanwar, and P. E. Shanahan. Flow-based generative models for Markov chain Monte Carlo in lattice field theory. *Physical Review D*, 100(3):034515, 2019. 7

[4] M. S. Albergo, G. Kanwar, S. Racanière, D. J. Rezende, J. M. Urban, D. Boyda, K. Cranmer, D. C. Hackett, and P. E. Shanahan. Flow-based sampling for fermionic lattice field theories. *Physical Review D*, 104:114507, 2021. doi: 10.1103/PhysRevD.104.114507. 7

[5] M. Arbel, A. Matthews, and A. Doucet. Annealed flow transport Monte Carlo. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. 7

[6] A. Benveniste, M. Metivier, and P. Priouret. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, Berlin, Heidelberg, 1990. doi: 10.1007/978-3-642-75894-2. 6

[7] N. Berglund, G. D. Gesù, and H. Weber. An Eyring–Kramers law for the stochastic Allen–Cahn equation in dimension two. *Electronic Journal of Probability*, 22:1 – 27, 2017. doi: 10.1214/17-EJP60. 8

[8] J. Berner, L. Richter, and K. Ullrich. An optimal control perspective on diffusion-based generative modeling. *Transaction on Machine Learning Research (TMLR)*, 2024. 7

[9] A. Beskos, A. Jasra, N. Kantas, and A. Thiery. On the convergence of adaptive sequential Monte Carlo methods. *Annals of Applied Probability*, 26(2):1111–1146, 2016. doi: 10.1214/15-AAP1113. 6

[10] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: a review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773. 2

[11] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`. 18

[12] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 1st edition, 2011. doi: 10.1201/b10905. 2

[13] A. Cabezas and C. Nemeth. Transport elliptical slice sampling. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023. 7

[14] R. Chen and Y. Lipman. Flow matching on general geometries. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024. 3

[15] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018. 2, 3

[16] V. De Bortoli, M. Hutchinson, P. Wirnsberger, and A. Doucet. Target score matching. *arXiv preprint arXiv:2402.08667*, 2024. 7

[17] L. Del Debbio, J. Marsh Rossney, and M. Wilson. Efficient modeling of trivializing maps for lattice ϕ 4 theory using normalizing flows: a first look at scalability. *Physical Review D*, 104(9):094507, 2021. doi: 10.1103/PhysRevD.104.094507. 2

[18] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006. 5, 7

 https://doi.org/10.52202/079017-3316

[19] A. Doucet, W. Grathwohl, A. G. Matthews, and H. Strathmann. Score-based diffusion meets annealed importance sampling. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 7

[20] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987. doi: 10.1016/0370-2693(87)91197-X. 2

[21] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 7

[22] F. Feroz, M. Hobson, and M. Bridges. MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics. *Monthly Notices of the Royal Astronomical Society*, 398(4):1601–1614, 2009. 2

[23] M. Gabrié, G. M. Rotskoff, and E. Vanden-Eijnden. Efficient Bayesian sampling using normalizing flows to assist Markov chain Monte Carlo methods. In *Proceedings of the 38th International Conference on Machine Learning (ICML): 3rd Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021. 4

[24] M. Gabrié, G. M. Rotskoff, and E. Vanden-Eijnden. Adaptive Monte Carlo augmented with normalizing flows. *Proceedings of the National Academy of Sciences*, 119(10):e2109420119, 2022. doi: 10.1073/pnas.2109420119. 2, 4, 5, 7, 8, 9, 20

[25] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995. 1

[26] J. Gorham and L. Mackey. Measuring sample quality with kernels. In *Proceedings of the Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. 9, 18

[27] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2018. 2, 3, 19, 21

[28] L. Grenioux, A. Durmus, É. Moulines, and M. Gabrié. On sampling with approximate transport maps. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. 2, 4, 6

[29] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schöolkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, 13:723–773, 2012. 7, 18

[30] M. G. Gu and F. H. Kong. A stochastic approximation algorithm with Markov chain Monte-Carlo method for incomplete data estimation problems. *Proceedings of the National Academy of Sciences*, 95(13):7270–7274, 1998. doi: 10.1073/pnas.95.13.7270. 17

[31] D. C. Hackett, C.-C. Hsieh, M. S. Albergo, D. Boyda, J.-W. Chen, K.-F. Chen, K. Cranmer, G. Kanwar, and P. E. Shanahan. Flow-based sampling for multimodal distributions in lattice field theory. *arXiv preprint arXiv:2107.00734*, 2021. 7

[32] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. doi: 10.1093/biomet/57.1.97. 2

[33] M. Hoffman, P. Sountsov, J. V. Dillon, I. Langmore, D. Tran, and S. Vasudevan. Neutra-lizing bad geometry in Hamiltonian Monte Carlo using neural transport. In *Proceedings of the 1st Symposium on Advances in Approximate Bayesian Inference (AABI)*, 2019. 2, 4, 7

[34] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013. 2

[35] N. T. Hunt-Smith, W. Melnitchouk, F. Ringer, N. Sato, A. W. Thomas, and M. J. White. Accelerating Markov chain Monte Carlo sampling with diffusion models. *Computer Physics Communications*, 296:109059, 2024. doi: 10.1016/j.cpc.2023.109059. 4, 7

[36] M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989. 19, 21

[37] A. Ihler, J. Fisher III, R. Moses, and A. Willsky. Nonparametric belief propagation for self-localization of sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4), 2005. doi: 10.1109/JSAC.2005.843548. 2

[38] S. T. Jensen, X. S. Liu, Q. Zhou, and J. S. Liu. Computational discovery of gene regulatory binding motifs: a Bayesian perspective. *Statistical Science*, 19(1):188–204, 2004. doi: 10.1214/088342304000000107. 2

[39] M. Karamanis, F. Beutler, J. A. Peacock, D. Nabergoj, and U. Seljak. Accelerating astronomical and cosmological inference with preconditioned Monte Carlo. *Monthly Notices of the Royal Astronomical Society*, 516(2):1644–1653, 2022. doi: 10.1093/mnras/stac2272. 2, 4

[40] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015. 7

[41] S. Kou, J. Oh, and W. H. Wong. A study of density of states and ground states in hydrophobic-hydrophilic protein folding models by equi-energy sampling. *The Journal of Chemical Physics*, 124(24), 2006. doi: 10.1063/1.2208607. 2

[42] A. Lee. *U-Statistics: Theory and Practice*. Taylor & Francis, 1st edition, 1990. doi: 10.1201/9780203734520. 18

[43] B. Leimkuhler and C. Matthews. Molecular dynamics. *Interdisciplinary Applied Mathematics*, 39:443, 2015. doi: 10.1007/978-3-319-16375-8. 1

[44] S.-H. Li and L. Wang. Neural network renormalization group. *Physical Review Letters*, 121: 260601, 2018. doi: 10.1103/PhysRevLett.121.260601. 4, 7

[45] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. 1, 2, 3, 5, 6

[46] Q. Liu, J. Lee, and M. Jordan. A kernelized Stein discrepancy for goodness-of-fit tests. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016. 9, 18

[47] A. H. Mahmoud, M. Masters, S. J. Lee, and M. A. Lill. Accurate sampling of macromolecular conformations using adaptive deep learning and coarse-grained representation. *Journal of Chemical Information and Modeling*, 62(7):1602–1617, 2022. doi: 10.1021/acs.jcim.1c01438. 2

[48] O. Mangoubi, N. S. Pillai, and A. Smith. Does Hamiltonian Monte Carlo mix faster than a random walk on multimodal densities? *arXiv preprint arXiv:1808.03230*, 2018. 2

[49] O. Mangoubi, N. Pillai, and A. Smith. Simple conditions for metastability of continuous Markov chains. *Journal of Applied Probability*, 58(1):83–105, 2021. doi: 10.1017/jpr.2020.83. 2

[50] A. Matthews, M. Arbel, D. J. Rezende, and A. Doucet. Continual repeated annealed flow transport Monte Carlo. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022. 7

[51] 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(6): 1087–1092, 1953. 1

[52] L. I. Midgley, V. Stimper, G. N. Simm, B. Schölkopf, and J. M. Hernández-Lobato. Flow annealed importance sampling bootstrap. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. 7, 18, 20

[53] J. Møller, A. R. Syversveen, and R. P. Waagepetersen. Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998. doi: 10.1111/1467-9469.00115. 10, 22

[54] C. Naesseth, F. Lindsten, and D. Blei. Markovian score climbing: variational inference with KL(p||q). *Proceedings of the 34th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 4, 6, 7, 17

 https://doi.org/10.52202/079017-3316

[55] R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11:125–139, 2001. doi: 10.1023/A:1008923215028. 5, 6

[56] R. M. Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2(11):2, 2011. 2

[57] K. A. Nicoli, S. Nakajima, N. Strodthoff, W. Samek, K.-R. Müller, and P. Kessel. Asymptotically unbiased estimation of physical observables with neural samplers. *Physical Review E*, 101(2): 023304, 2020. doi: 10.1103/PhysRevE.101.023304. 7

[58] F. Noé, S. Olsson, J. Köhler, and H. Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457), 2019. doi: 10.1126/science.aaw1147. 7, 20

[59] M. D. Parno and Y. M. Marzouk. Transport map accelerated Markov chain Monte Carlo. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):645–682, 2018. doi: 10.1137/17M1134640. 2, 4, 7

[60] A. Phillips, H.-D. Dau, M. J. Hutchinson, V. D. Bortoli, G. Deligiannidis, and A. Doucet. Particle denoising diffusion sampler. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. 7

[61] R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014. 2

[62] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015. 2

[63] L. Richter, J. Berner, and G.-H. Liu. Improved sampling via learned diffusions. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024. 7

[64] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, New York, 2nd edition, 2004. doi: 10.1007/978-1-4757-4145-22. 2

[65] G. O. Roberts and J. S. Rosenthal. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20 – 71, 2004. doi: 10.1214/154957804100000024. 2

[66] G. O. Roberts and R. L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996. 2

[67] S. Samsonov, E. Lagutin, M. Gabrié, A. Durmus, A. Naumov, and E. Moulines. Local-global MCMC kernels: the best of both worlds. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 2, 4, 7

[68] C. Schönle and M. Gabrié. Optimizing Markov chain Monte Carlo convergence with normalizing flows and Gibbs sampling. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems (NeurIPS): AI for Science Workshop*, 2023. 2, 7

[69] M. Sendera, M. Kim, S. Mittal, P. Lemos, L. Scimeca, J. Rector-Brooks, A. Adam, Y. Bengio, and N. Malkin. Improved off-policy training of diffusion samplers. *arXiv preprint arXiv:2402.05098*, 2024. 7

[70] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 5

[71] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 7

[72] A. Tong, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, K. Fatras, G. Wolf, and Y. Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research (TMLR)*, 2024. 4

[73] B. Tzen and M. Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Proceedings of the 32nd Annual Conference on Learning Theory (COLT)*, 2019. 7

[74] F. Vargas and N. Nüsken. Transport, variational inference and diffusions: with applications to annealed flows and Schrödinger bridges. *Proceedings of the 40th International Conference on Machine Learning (ICML): Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023. 7

[75] F. Vargas, W. Grathwohl, and A. Doucet. Denoising diffusion samplers. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. 7

[76] F. Vargas, A. Ovsianas, D. Fernandes, M. Girolami, N. D. Lawrence, and N. Nüsken. Bayesian learning via neural Schrödinger–Föllmer flows. *Statistics and Computing*, 33(1):3, 2023. doi: 10.1007/s11222-022-10172-5. 7

[77] F. Vargas, T. Reu, and A. Kerekes. Expressiveness remarks for denoising diffusion models and samplers. In *Proceedings of the 5th Symposium on Advances in Approximate Bayesian Inference (AABI)*, 2023. 7

[78] F. Vargas, S. Padhy, D. Blessing, and N. Nüsken. Transport meets variational inference: controlled Monte Carlo diffusions. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024. 7

[79] M. J. Wainwright, M. I. Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008. doi: 10.1561/2200000001. 2

[80] K. W. K. Wong, M. Gabrié, and D. Foreman-Mackey. flowMC: Normalizing flow enhanced sampling package for probabilistic inference in JAX. *Journal of Open Source Software*, 8(83): 5021, 2023. 18

[81] H. Wu, J. Köhler, and F. Noé. Stochastic normalizing flows. *Advances in Neural Information Processing Systems*, 33:5933–5944, 2020. 20

[82] B. J. Zhang, Y. M. Marzouk, and K. Spiliopoulos. Transport map unadjusted Langevin algorithms. *arXiv preprint arXiv:2302.07227*, 2023. 7

[83] D. Zhang, R. T. Q. Chen, C.-H. Liu, A. Courville, and Y. Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024. 7

[84] L. Zhang, D. M. Blei, and C. A. Naesseth. Transport score climbing: variational inference using forward kl and adaptive neural transport. *Transactions on Machine Learning Research (TMLR)*, 2023. 7

[85] Q. Zhang and Y. Chen. Path integral sampler: a stochastic control approach for sampling. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022. 7

 https://doi.org/10.52202/079017-3316

A Flow-Informed Markov Chain Monte Carlo Methods

Algorithm 2 Flow-informed Random Walk Metropolis Hastings

Input: initial x, target π, vector field v_t^θ, flow parameters θ
Output: x'
$\sigma_{opt} \leftarrow 2.38/\sqrt{d}$
$\phi_1(x) = x_{t=1} \leftarrow x$
$\begin{bmatrix} x_0 \\ \Delta \log p(x_0) \end{bmatrix} \leftarrow \begin{bmatrix} x \\ 0 \end{bmatrix} + \int_1^0 \begin{bmatrix} v_t^\theta(\phi_t(x)) \\ -\nabla \cdot v_t^\theta(\phi_t(x)) \end{bmatrix} \mathrm{d}t$
$y_0 \sim \mathcal{N}(\cdot|x_0, \sigma_{opt}^2)$
$\begin{bmatrix} y_1 \\ \Delta \log p(y_1) \end{bmatrix} \leftarrow \begin{bmatrix} y_0 \\ 0 \end{bmatrix} + \int_0^1 \begin{bmatrix} v_t^\theta(\phi_t(y)) \\ -\nabla \cdot v_t^\theta(\phi_t(y)) \end{bmatrix} \mathrm{d}t$
$\alpha \leftarrow \min\left\{1, \frac{\pi(y_1)\exp(-\Delta \log p(y_1))}{\pi(x_1)\exp(\Delta \log p(x_0))}\right\}$
With probability α make $x' \leftarrow y_1$ else $x' \leftarrow x$

Algorithm 3 Flow-informed Independent Metropolis Hastings

Input: initial x, target density π, vector field v_t^θ, reference density p_0, flow parameters θ.
Output: x'
$\phi_1(u) = u_{t=1} \leftarrow x$
$\begin{bmatrix} u_0 \\ \Delta \log p(u_0) \end{bmatrix} \leftarrow \begin{bmatrix} u_1 \\ 0 \end{bmatrix} + \int_1^0 \begin{bmatrix} v_t^\theta(\phi_t(u)) \\ -\nabla \cdot v_t^\theta(\phi_t(u)) \end{bmatrix} dt$
$\phi_0(x) = x_{t=0} \sim p_0$
$\begin{bmatrix} x_1 \\ \log p(x_1) \end{bmatrix} \leftarrow \begin{bmatrix} x_0 \\ \log p_0(x_0) \end{bmatrix} + \int_0^1 \begin{bmatrix} v_t^\theta(\phi_t(x)) \\ -\nabla \cdot v_t^\theta(\phi_t(x)) \end{bmatrix} dt$
$\alpha \leftarrow \min\left\{1, \frac{\pi(x_1)p_0(u_0)\exp(-\Delta \log p(u_0))}{\exp(\log p(x_1))\pi(u_1)}\right\}$
With probability α make $x' \leftarrow x_1$ else $x' \leftarrow x$

Algorithm 4 Flow-informed Conditional Importance Sampling

Input: initial x, target density π, vector field v_t^θ, reference density q_0, flow parameters θ, number of importance samples K.
Output: x'
$\phi_1(u) = u_{t=1} \leftarrow x$
$\begin{bmatrix} u_0 \\ \Delta \log p(u_0) \end{bmatrix} \leftarrow \begin{bmatrix} u_1 \\ 0 \end{bmatrix} + \int_1^0 \begin{bmatrix} v_t^\theta(\phi_t(u)) \\ -\nabla \cdot v_t^\theta(\phi_t(u)) \end{bmatrix} dt$
$w_0 \leftarrow \frac{\pi(u_1)}{p_0(u_0)\exp(-\Delta \log p(u_0))}$
$x^{(0)} \leftarrow x$
for $k = 1 : K$ **do**
    $\phi_0(x) = x_{t=0} \sim q_0$
    $\begin{bmatrix} x_1 \\ \log p(x_1) \end{bmatrix} \leftarrow \begin{bmatrix} x_0 \\ \log p_0(x_0) \end{bmatrix} + \int_0^1 \begin{bmatrix} v_t^\theta(\phi_t(x)) \\ -\nabla \cdot v_t^\theta(\phi_t(x)) \end{bmatrix} dt$
    $w_k \leftarrow \frac{\pi(x_1)}{\exp(\log p(x_1))}$
    $x^{(k)} \leftarrow x_1$
end for
Choose k' with probability $P(k' = k) \propto w_k$, then make $x' \leftarrow x^{(k')}$

B Proof of Proposition 3.1

Our proof follows closely the proof of [54, Proposition 1]. Let θ^* be a minimizer of the CFM objective in (4), which we recall is given by

$$\mathcal{J}(\theta;\pi) = \mathbb{E}_{x_1\sim\pi}\mathbb{E}_{t\sim\mathcal{U}(0,1)}\mathbb{E}_{x\sim p_t(\cdot|x_1)}\left[||v_t^\theta(x) - v_t(x|x_1)||^2\right] := \mathbb{E}_{x_1\sim\pi}\left[j(\theta,x_1)\right] \tag{20}$$

where we have defined $j(\theta,x_1) = \mathbb{E}_{t\sim\mathcal{U}(0,1)}\mathbb{E}_{x\sim p_t(\cdot|x_1)}\left[||v_t^\theta(x) - v_t(x|x_1)||^2\right]$. Now, consider the ordinary differential equation (ODE) given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\theta(t) = \nabla\mathcal{J}_\theta(\theta(t);\pi), \quad \theta(0) = \theta_0, \quad t \geq 0. \tag{21}$$

We say that $\hat{\theta}$ is stability point of this ODE if, given the initial condition $\theta(0) = \hat{\theta}$, the ODE admits the unique solution $\theta(t) = \hat{\theta}$ for all $t \geq 0$. Naturally, the minimizer θ^* is a stability point of this ODE, since $\nabla_\theta\mathcal{J}(\theta;\pi)|_{\theta=\theta^*} = 0$. Meanwhile, we call Θ the domain of attraction of θ^* if, given the initial condition $\theta(0) \in \Theta$, the solution $\theta(t) \in \Theta$ for all $t \geq 0$, and $\theta(t)$ converges to θ^* as $t \to \infty$.

Let $x_k \in \mathbb{R}^{d_x}$, and $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ be an open subset of $\mathbb{R}^{d_x}$. Let $\Theta \subseteq \mathbb{R}^{d_\theta}$ be an open set in $\mathbb{R}^{d_\theta}$, and $\Theta_c \subseteq \Theta$ be a compact subset of Θ. Consider the Markov transition kernel $M := P \circ Q^k$ given by a cycle of k_Q repeated transitions of a MALA transition kernel and a flow-informed RWMH transition kernel, viz

$$M_{\pi,\theta}(x,\mathrm{d}y) = \int\cdots\int Q(x,\mathrm{d}x_1;\theta)Q(x_1,\mathrm{d}x_2)\ldots Q(x_{k_Q-1},\mathrm{d}x_{k_Q};\theta)P(x_{k_Q},\mathrm{d}y;\pi,\theta). \tag{22}$$

This transition kernel is π-invariant since both P and Q are π-invariant. In addition, let $M^k_{\pi,\theta}(x,\mathrm{d}y)$ be the repeated application of this Markov transition kernel, namely,

$$M^k_{\pi,\theta}(x,\mathrm{d}y) = \int\cdots\int M_{\pi,\theta}(x,\mathrm{d}x_1)M_{\pi,\theta}(x_1,\mathrm{d}x_2)\cdots M_{\pi,\theta}(x_{k-2},\mathrm{d}x_{k-1})M_{\pi,\theta}(x_{k-1},\mathrm{d}y). \tag{23}$$

Following [30, 54], we impose the following assumptions, for some sufficiently large positive real number $q > 1$.

Assumption B.1 (Robbins-Monro Condition)**.** The step size sequence $(\varepsilon_k)_{k=1}^\infty$ satisfies the following requirements:

$$\sum_{k=1}^\infty \varepsilon_k = \infty, \quad \sum_{k=1}^\infty \varepsilon_k^2 < \infty. \tag{24}$$

Assumption B.2 (Integrability)**.** There exists a constant $C_1 > 0$ such that for, any $\theta \in \Theta$, $x \in \mathcal{X}$ and $k \geq 1$,

$$\int(1+|y|^q)M^k_{\pi,\theta}(x,\mathrm{d}y) \leq C_1(1+|x|^q). \tag{25}$$

Assumption B.3 (Convergence of the Markov Chain)**.** For each $\theta \in \Theta$, it holds that

$$\lim_{k\to\infty}\sup_{x\in\mathcal{X}}\frac{1}{1+|x|^q}\int(1+|y|^q)|M^k_{\pi,\theta}(x,\mathrm{d}y) - \pi(\mathrm{d}y)| = 0. \tag{26}$$

Assumption B.4 (Continuity in θ)**.** There exists a constant C_2 such that for all $\theta,\theta' \in \Theta_c$,

$$\left|\int(1+|y|^q)(M^k_{\pi,\theta}(x,\mathrm{d}y) - M^k_{\pi,\theta'}(x,\mathrm{d}y))\right| \leq C_2|\theta-\theta'|(1+|x|^q). \tag{27}$$

Assumption B.5 (Continuity in x)**.** There exists a constant C_3 such that for all $x_1,x_2 \in \mathcal{X}$,

$$\sup_{\theta\in\Theta}\left|\int(1+|y|^{q+1})(M^k_{\pi,\theta}(x_1,\mathrm{d}y) - M^k_{\pi,\theta}(x_2,\mathrm{d}y))\right| \leq C_3|x_1-x_2|(1+|x_1|^q+|x_2|^q). \tag{28}$$

Assumption B.6 (Conditions on the Objective Function)**.** For any compact subset $\Theta_c \subset \Theta$, there exist positive constants p, K_1, K_2, K_3 and $v > 1/2$ such that for all $\theta,\theta' \in \Theta_c$ and $x, x_1, x_2 \in \mathcal{X}$,

$$|\nabla_\theta j(\theta,x_1)| \leq K_1(1+|x_1|^{p+1}), \tag{29}$$

$$|\nabla_\theta j(\theta,x_1) - \nabla_\theta j(\theta,x_1')| \leq K_2|x_1-x_1'|(1+|x_1|^p+|x_2|^p), \tag{30}$$

$$|\nabla_\theta j(\theta,x_1) - \nabla_\theta j(\theta',x_1)| \leq K_3|\theta-\theta'|^v(1+|x_1|^{p+1}). \tag{31}$$

With the above assumptions, the result follows from Theorem 1 of [30] by setting $x \to x_1$, $\Pi_\theta \to M_{\pi,\theta}$, $H(\theta,x) \to \nabla_\theta j(\theta,x_1)$.

https://doi.org/10.52202/079017-3316

C Additional Experimental Details

Code for the numerical experiments is written in Python with array computations handled by JAX [11]. The implementation of relevant methods for comparison is sourced from open source repositories: DDS using franciscovargas/denoising_diffusion_samplers, NF-MCMC using kazewong/flowMC [80], and FAB using lollcat/fab-jax [52]. All experiments are run on an NVIDIA V100 GPU with 32GB of memory. In the following subsections, we will give more details on the modelling and hyperparameter choices for each experiment, along with additional results.

C.1 Diagnostics

Let π and ν be two probability measures. Let $\mathcal{F}$ denote the unit ball in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, associated with the positive definite kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Then the maximum mean discrepancy (MMD) between π and ν is defined as [29, Section 2.2]

$$\mathrm{MMD}_k^2(\pi, \nu) = \|m_\pi - m_\nu\|_{\mathcal{F}}^2, \tag{32}$$

where m_π is the mean embedding of π, defined via $\mathbb{E}_\pi[f] = \langle f, m_\pi \rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$. Using standard properties of the RKHS, the squared MMD can be written as [29, Lemma 6]

$$\mathrm{MMD}_k^2(\pi, \nu) = \mathbb{E}_{x,x' \sim \pi}\left[k(x, x')\right] - 2\mathbb{E}_{x \sim \pi, y \sim \nu}\left[k(x, y)\right] + \mathbb{E}_{y \sim \nu, y' \sim \nu}\left[k(y, y')\right]. \tag{33}$$

Thus, given samples $(x_i)_{i=1}^m \sim \pi$ and $(y_i)_{i=1}^m \sim \nu$, an unbiased estimate of the squared MMD can be computed as

$$\begin{aligned}\widehat{\mathrm{MMD}}_k^2(\pi, \nu) = &\frac{1}{m(m-1)} \sum_{i=1}^m \sum_{i \neq j}^m k(x_i, x_j) - \frac{2}{m^2} \sum_{i=1}^m \sum_{j=1}^m k(x_i, y_j) \\ &+ \frac{1}{m(m-1)} \sum_{i=1}^m \sum_{j \neq i}^m k(y_i, y_j).\end{aligned} \tag{34}$$

For a kernel k, the kernel Stein discrepancy (KSD) between π and ν is defined as the MMD between π and ν, using the Stein kernel k_π associated with k, which is defined as

$$\begin{aligned}k_\pi(x, x') = &\nabla_x \cdot \nabla_{x'} k(x, x') + \nabla_x k(x, x') \cdot \nabla_{x'} \log \pi(x') \\ &+ \nabla_{x'} k(x, x') \cdot \nabla_x \log \pi(x) + k(x, x') \nabla_x \log \pi(x) \cdot \nabla_{x'} \log \pi(x),\end{aligned} \tag{35}$$

and satisfies the Stein identity $\mathbb{E}_\pi\left[k_\pi(x, \cdot)\right] = 0$. We thus have that

$$\begin{aligned}\mathrm{KSD}_k^2(\pi, \nu) &= \mathrm{MMD}_{k_\pi}^2(\pi, \nu) && (36)\\ &= \mathbb{E}_{x,x' \sim \pi}\left[k_\pi(x, x')\right] - 2\mathbb{E}_{x \sim \pi, y \sim \nu}\left[k_\pi(x, y)\right] + \mathbb{E}_{y \sim \nu, y' \sim \nu}\left[k_\pi(y, y')\right] && (37)\\ &= \mathbb{E}_{y \sim \nu, y' \sim \nu}\left[k_\pi(y, y')\right]. && (38)\end{aligned}$$

We can obtain estimates of the KSD by using U-statistics or V-statistics. In particular, an unbiased estimate of $\mathrm{KSD}_k^2(\pi, \nu)$ is given by the U-statistic [42]

$$\widehat{\mathrm{KSD}}_{k,U}^2(\pi, \nu) = \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{i \neq j}^n k_\pi(y_i, y_i'). \tag{39}$$

Alternatively, we can estimate $\mathrm{KSD}_k^2(\pi, \nu)$ using a biased (but non-negative) V-statistic of the form [46, Section 4]

$$\widehat{\mathrm{KSD}}_{k,V}^2(\pi, \nu) = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n k_\pi(y_i, y_i'). \tag{40}$$

In all of our numerical experiments, we calculate the U- and V- statistics using the inverse multi-quadratic kernel $k(x, x') = (1 + (x - x')^T (x - x'))^\beta$ due to its favourable convergence properties [26, Theorem 8], setting $\beta = -\frac{1}{2}$.

C.2 4-mode Gaussian mixture

For this experiment, all methods use $N = 128$ parallel chains for training and 128 hidden dimensions for all neural networks. Methods with a MALA kernel use a step size of 0.2, and methods with splines use 4 coupling layers with 8 bins and range limited to $[-16, 16]$.

In Table 3, we present results for $K = 10^3$ iterations. Since MFM is much more efficient than other methods, we also report results for a great number of total iterations. Table 4 contains results for $K = 5 \cdot 10^3$ iterations for MFM and AT-SMC. In the main text, we present results for $K = 5 \cdot 10^3$ learning iterations for MFM and AT-SMC, and $K = 10^3$ iterations for the other algorithms, since this renders the total computational cost of all algorithms somewhat comparable.

For both choices of K, we also present results using Hutchinson's trace estimator (HTE) [27, 36] to calculate the MH acceptance probability in the flow-informed Markov transition kernel. As expected, its effect on sample quality becomes more apparent as k_Q increases. However, its effect on computation time is less significant than in larger dimensional examples.

$K = 10^3$	$\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$	KSD U-stat.	KSD V-stat.	MMD	seconds
FM w/ π samples	-4.20 ± 0.08	6.85e-3±3.75e-3	7.16e-3±3.75e-3	1.90e-3±1.16e-3	6.46 ± 0.32
MFM $k_Q = K$	-4.55 ± 0.16	7.62e-3±7.46e-3	7.98e-3±7.45e-3	3.00e-3±2.20e-3	10.4 ± 0.66
MFM $k_Q = 10^2$	-4.50 ± 0.14	5.36e-3±3.18e-3	5.71e-3±3.19e-3	2.03e-3±2.12e-3	12.1 ± 0.65
– w/ HTE	-4.52 ± 0.15	4.77e-3±1.93e-3	5.12e-3±1.93e-3	2.27e-3±1.30e-3	13.9 ± 0.33
MFM $k_Q = 10$	-4.49 ± 0.08	7.01e-3±3.49e-3	7.36e-3±3.49e-3	1.62e-3±7.61e-4	24.8 ± 1.38
– w/ HTE	-4.53 ± 0.12	1.10e-2±6.80e-3	1.14e-2±6.81e-3	2.64e-3±1.03e-3	30.0 ± 1.70
DDS	-4.22 ± 0.03	9.89e-4±1.05e-3	1.30e-3±1.05e-3	1.76e-4±2.32e-4	$114. \pm 0.68$
NF-MCMC	-4.37 ± 0.21	1.80e-2±1.44e-2	1.83e-2±1.44e-2	5.85e-3±3.91e-3	72.0 ± 11.7
FAB	-4.67 ± 0.16	2.31e-3±1.19e-3	2.69e-3±1.21e-3	2.69e-4±2.06e-4	$101. \pm 3.24$
AT-SMC	-4.47 ± 0.04	3.95e-3±2.06e-3	4.30e-3±2.06e-3	2.98e-2±4.08e-2	1.38 ± 0.08

Table 3: Diagnostics for the 4-mode Gaussian mixture with $K = 10^3$. $\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$ is the Monte Carlo approximation of the log-target density using the learned flow to generate samples; KSD U-stat and V-stat are the Kernel Stein Discrepancy U- and V-statistics between the target and samples generated from the learned flow; MMD is the Maximum Mean Discrepancy between real samples from the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

$K = 5 \cdot 10^3$	$\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$	KSD U-stat.	KSD V-stat.	MMD	seconds
FM w/ π samples	-4.22 ± 0.04	1.50e-3±6.38e-4	1.81e-3±6.33e-4	3.69e-4±1.84e-4	22.3 ± 0.64
MFM $k_Q = K$	-4.47 ± 0.05	3.15e-3±2.10e-3	3.50e-3±2.10e-3	2.37e-3±2.29e-3	27.9 ± 1.27
MFM $k_Q = 10^2$	-4.45 ± 0.04	3.61e-3±2.07e-3	3.96e-3±2.07e-3	1.05e-3±8.90e-4	39.2 ± 1.74
– w/ HTE	-4.48 ± 0.09	3.50e-3±2.22e-3	3.86e-3±2.22e-3	1.88e-3±1.96e-3	41.2 ± 1.65
MFM $k_Q = 10$	-4.44 ± 0.07	3.15e-3±2.28e-3	3.49e-3±2.28e-3	8.13e-4±4.41e-4	$117. \pm 5.65$
– w/ HTE	-4.46 ± 0.06	4.80e-3±3.17e-3	5.15e-3±3.17e-3	1.37e-3±9.65e-4	$147. \pm 9.44$
AT-SMC	-4.48 ± 0.04	4.07e-3±1.24e-3	4.42e-3±1.24e-3	3.95e-2±2.90e-2	2.18 ± 0.26

Table 4: Diagnostics for the 4-mode Gaussian mixture with $K = 5 \cdot 10^3$. $\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$ is the Monte Carlo approximation of the log-target density using the learned flow to generate samples; KSD U-stat and V-stat are the Kernel Stein Discrepancy U- and V-statistics between the target and samples generated from the learned flow; MMD is the Maximum Mean Discrepancy between real samples from the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

C.3 16-mode Gaussian Mixture

Like the 4-mode example, all methods use $N = 128$ parallel chains for training and 128 hidden dimensions for all neural networks. Methods with a MALA kernel use a step size of 0.2, and methods with splines use 4 coupling layers with 8 bins and range limited to $[-16, 16]$. In Table 5 we present results for $K = 10^3$ iterations. In Table 6, we provide results for MFM and AT-SMC for $K = 5 \times 10^3$ learning iterations. In the main text, we present results for $K = 5 \cdot 10^3$ learning iterations for MFM and AT-SMC and $K = 10^3$ iterations for all other algorithms, which yields a more comparable total computation time.

https://doi.org/10.52202/079017-3316

$K = 10^3$	$\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$	KSD U-stat.	KSD V-stat.	MMD	seconds
FM w/ π samples	-7.09 ± 0.31	2.94e-2±1.32e-2	2.99e-2±1.32e-2	8.89e-3±1.79e-3	6.67 ± 0.19
MFM $k_Q = K$	-6.95 ± 0.47	1.67e-2±1.14e-2	1.71e-2±1.15e-2	3.71e-2±4.81e-3	10.4 ± 0.64
MFM $k_Q = 10^2$	-7.21 ± 0.73	1.80e-2±9.31e-3	1.85e-2±9.43e-3	1.81e-2±8.97e-3	11.4 ± 0.74
– w/ HTE	-7.34 ± 0.81	2.10e-2±8.65e-3	2.15e-2±8.75e-3	1.74e-2±8.12e-3	13.0 ± 0.83
MFM $k_Q = 10$	-7.21 ± 0.58	2.95e-2±1.03e-2	3.01e-2±1.04e-2	1.06e-2±2.76e-3	20.3 ± 1.17
– w/ HTE	-7.18 ± 0.85	3.17e-2±1.97e-2	3.23e-2±1.99e-2	1.30e-2±3.33e-3	22.5 ± 1.81
DDS	-5.86 ± 0.20	6.65e-3±6.69e-3	6.94e-3±6.69e-3	1.02e-1±4.10e-2	$115. \pm 0.64$
NF-MCMC	-5.74 ± 0.35	1.23e-2±1.71e-2	1.26e-2±1.72e-2	8.05e-3±1.42e-2	67.0 ± 12.3
FAB	-5.89 ± 0.28	4.22e-3±3.31e-3	4.58e-3±3.34e-3	1.51e-3±1.06e-3	$102. \pm 4.32$
AT-SMC	-5.91 ± 0.07	2.07e-3±1.03e-3	2.38e-3±1.04e-3	3.72e-2±4.45e-3	1.36 ± 0.20

Table 5: Diagnostics for the 16-mode Gaussian mixture with $K = 10^3$. $\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$ is the Monte Carlo approximation of the log-target density using the learned flow to generate samples; KSD U-stat and V-stat are the Kernel Stein Discrepancy U- and V-statistics between the target and samples generated from the learned flow; MMD is the Maximum Mean Discrepancy between real samples from the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

$K = 5 \cdot 10^3$	$\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$	KSD U-stat.	KSD V-stat.	MMD	seconds
FM w/ π samples	-5.74 ± 0.11	2.91e-3±1.23e-3	3.26e-3±1.24e-3	1.35e-3±6.66e-4	22.4 ± 1.01
MFM $k_Q = K$	-6.09 ± 0.10	3.00e-3±7.87e-4	3.34e-3±7.95e-4	1.88e-2±3.67e-3	28.2 ± 2.84
MFM $k_Q = 10^2$	-5.90 ± 0.08	5.37e-3±2.00e-3	5.74e-3±2.01e-3	2.98e-3±1.37e-3	34.8 ± 1.95
– w/ HTE	-5.88 ± 0.12	5.43e-3±2.32e-3	5.80e-3±2.33e-3	3.86e-3±1.23e-3	38.7 ± 2.53
MFM $k_Q = 10$	-5.98 ± 0.13	5.48e-3±2.07e-3	5.86e-3±2.09e-3	2.87e-3±9.67e-4	89.6 ± 5.19
– w/ HTE	-5.92 ± 0.09	9.18e-3±4.48e-3	9.57e-3±4.50e-3	8.58e-3±1.00e-3	$110. \pm 6.77$
AT-SMC	-5.84 ± 0.05	2.09e-3±8.53e-4	2.40e-3±8.56e-4	1.73e-2±5.30e-3	2.19 ± 0.21

Table 6: Diagnostics for the 16-mode Gaussian mixture with $K = 5 \cdot 10^3$. $\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$ is the Monte Carlo approximation of the log-target density using the learned flow to generate samples; KSD U-stat and V-stat are the Kernel Stein Discrepancy U- and V-statistics between the target and samples generated from the learned flow; MMD is the Maximum Mean Discrepancy between real samples from the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

C.4 Many Well

We also present a synthetic problem approximating the 32-dimensional Many Well distribution given by the product of 16 copies of the 2-dimensional Double Well distribution [e.g., 52, 58, 81],

$$\log p(x_1, x_2) = -x_1^4 + 6x_1^2 + \frac{1}{2}x_1 - \frac{1}{2}x_2^2 + \text{constant}, \tag{41}$$

where each copy of the Double Well is evaluated on a different pair of the 32 inputs. The 32-dimensional Many Well has $2^{16} = 65536$ modes, one for each possible choice of mode in each of the 16 copies of the double well. We can obtain exact samples from the Many Well by sampling each independent copy of the Double Well.

Like previous synthetic examples, all methods use $N = 128$ parallel chains for training and 128 hidden dimensions for all neural networks. Methods with a MALA kernel use a step size of 0.1, and methods with splines use 4 coupling layers with 8 bins and a range limited to $[-16, 16]$. Table 7 presents results for $K = 10^3$ iterations. In Table 8, we provide results for MFM and AT-SMC for $K = 5 \times 10^3$ learning iterations. In the main text, we present results for $K = 5 \cdot 10^3$ learning iterations for MFM and AT-SMC and $K = 10^3$ iterations for all other algorithms, which yields a more comparable total computation time.

C.5 Field system

The stochastic Allen–Cahn equation is defined in terms of a random field $\phi : [0, 1] \to \mathbb{R}$ satisfying the following stochastic partial differential equation [e.g., 24, Section V]:

$$\frac{\partial \phi}{\partial t} = a\frac{\partial^2 \phi}{\partial s^2} + a^{-1}(\phi - \phi^3) + \sqrt{2\beta^{-1}}\eta(t, s), \tag{42}$$

$K = 10^3$	$\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$	KSD U-stat.	KSD V-stat.	MMD	seconds
FM w/ π samples	86.6 ± 1.88	17.4 ± 6.24	19.3 ± 6.25	1.24e-8±1.81e-8	6.95 ± 0.31
MFM $k_Q = K$	$101. \pm 0.70$	0.12 ± 0.12	0.77 ± 0.13	2.28e-8±1.57e-8	11.1 ± 0.67
MFM $k_Q = 10^2$	$101. \pm 0.70$	0.12 ± 0.11	0.77 ± 0.13	2.28e-8±1.57e-8	$120. \pm 17.6$
– w/ HTE	$101. \pm 0.70$	0.11 ± 0.12	0.77 ± 0.13	2.28e-8±1.57e-8	35.6 ± 3.95
MFM $k_Q = 10$	$101. \pm 0.77$	0.11 ± 0.11	0.76 ± 0.12	2.28e-8±1.57e-8	$1100. \pm 90.2$
– w/ HTE	$101. \pm 0.69$	0.11 ± 0.09	0.76 ± 0.10	2.27e-8±1.56e-8	$259. \pm 17.6$
DDS	$133. \pm 3.92$	0.13 ± 0.14	0.61 ± 0.13	1.49e-5±2.03e-5	$227. \pm 0.91$
NF-MCMC	39.2 ± 1.61	1.17 ± 0.65	2.29 ± 0.66	4.79e-7±2.41e-7	$184. \pm 44.1$
FAB	$137. \pm 0.33$	4.79e-3±2.55e-2	4.32e-1±4.14e-2	3.87e-7±2.90e-7	$304. \pm 3.64$
AT-SMC	$130. \pm 2.93$	0.02 ± 0.03	0.57 ± 0.03	1.75e-5±1.19e-5	1.35 ± 0.36

Table 7: Diagnostics for the many well with $K = 10^3$. $\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$ is the Monte Carlo approximation of the log-target density using the learned flow to generate samples; KSD U-stat and V-stat are the Kernel Stein Discrepancy U- and V-statistics between the target and samples generated from the learned flow; MMD is the Maximum Mean Discrepancy between real samples from the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

$K = 5 \cdot 10^3$	$\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$	KSD U-stat.	KSD V-stat.	MMD	seconds
FM w/ π samples	98.6 ± 5.32	2.57 ± 4.07	4.56 ± 4.07	−1.13e-9±2.81e-8	25.6 ± 0.48
MFM $k_Q = K$	$101. \pm 0.64$	1.02e-1±9.03e-2	7.64e-1±9.78e-2	2.28e-8±1.57e-8	29.7 ± 1.03
MFM $k_Q = 10^2$	$101. \pm 0.63$	1.02e-1±9.05e-2	7.64e-1±9.73e-2	2.28e-8±1.57e-8	$587. \pm 27.0$
– w/ HTE	$102. \pm 0.78$	1.02e-1±8.32e-2	7.62e-1±8.91e-2	2.27e-8±1.58e-8	$154. \pm 7.51$
MFM $k_Q = 10$	$102. \pm 0.81$	9.93e-2±7.24e-2	7.63e-1±8.15e-2	2.27e-8±1.57e-8	$5130. \pm 104.$
– w/ HTE	$103. \pm 2.20$	3.85e-1±2.69e-1	1.05 ± 0.27	2.21e-8±1.57e-8	$1190. \pm 27.6$
AT-SMC	$130. \pm 3.17$	2.44e-2±5.99e-2	5.79e-1±6.76e-2	1.52e-5±1.00e-5	2.59 ± 0.27

Table 8: Diagnostics for the many well with $K = 5 \cdot 10^3$. $\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$ is the Monte Carlo approximation of the log-target density using the learned flow to generate samples; KSD U-stat and V-stat are the Kernel Stein Discrepancy U- and V-statistics between the target and samples generated from the learned flow; MMD is the Maximum Mean Discrepancy between real samples from the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

where $a > 0$ is a parameter, β is the inverse temperature, $s \in [0, 1]$ denotes the spatial variable, and η is spatiotemporal white noise. We impose Dirichlet boundary conditions throughout, so that $\phi(s = 0) = \phi(s = 1) = 0$.

The associated Hamiltonian, reflecting a spatial coupling term penalizing changes in ϕ, takes the form:

$$U_*[\phi] = \beta \int_0^1 \left[\frac{a}{2} \left(\frac{\partial \phi}{\partial s} \right)^2 + \frac{1}{4a} \left(1 - \phi^2(s) \right)^2 \right] ds. \tag{43}$$

At low temperatures, this coupling induces alignment of the field in either the positive or negative direction, leading to two global minima, ϕ_+ and ϕ_-, with typical values of ± 1.

For this example, all methods use $N = 1024$ parallel chains for training and 256 hidden dimensions for all neural networks. Methods with a MALA kernel use a step size of 0.0001, and methods with splines use 8 coupling layers with 8 bins and range limited to $[-5, 5]$. Results for $K = 10^4$ learning iterations are presented in Table 9. In the main text, we present results for $k_Q = 10^2$ for MFM.

Interestingly, in high-dimensional problems such as this and the following, the version of the algorithm using Hutchinson's trace estimator [27, 36] to calculate the MH acceptance probability has little apparent effect on the approximation quality. It does, however, have a significant impact on the computation time.

C.6 Log-Gaussian Cox process

The original 10×10 square meter plot is standardized to the unit square. We discretize the unit square $[0, 1]^2$ into a $M = 40 \times 40$ regular grid. The latent intensity process $\Lambda = \{\Lambda_m\}_{m \in M}$ is specified as $\Lambda_m = \exp(X_m)$, where $X = \{X_m\}_{m \in M}$ is a Gaussian process with a constant mean $\mu_0 \in \mathbb{R}$ and exponential covariance function $\Sigma_0(m, n) = \sigma^2 \exp(-|m - n|/(40\beta))$ for $m, n \in M$,

https://doi.org/10.52202/079017-3316

$K = 10^4$	$\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$	KSD U-stat.	KSD V-stat.	seconds
MFM $k_Q = K$	-74.7 ± 5.67	2.61 ± 2.00	20.9 ± 2.49	52.3 ± 1.23
MFM $k_Q = K/10$	-69.6 ± 6.07	2.90 ± 2.50	21.2 ± 3.16	61.7 ± 4.37
– w/ HTE	-74.2 ± 6.51	2.67 ± 2.16	21.0 ± 2.66	53.6 ± 1.33
MFM $k_Q = K/100$	-55.4 ± 4.19	2.84 ± 3.31	20.9 ± 3.33	180 ± 42.2
– w/ HTE	-72.1 ± 11.6	3.84 ± 3.04	22.6 ± 4.26	71.5 ± 6.93
DDS	-76.3 ± 17.4	15.2 ± 35.9	18.0 ± 36.9	2400 ± 8.65
NF-MCMC	-26.9 ± 9.62	548 ± 325	549 ± 325	2000 ± 15.6
FAB	-50.4 ± 0.14	0.14 ± 0.42	1.78 ± 0.42	3880 ± 7.19
AT-SMC	-63.5 ± 9.91	1.61 ± 2.33	18.4 ± 2.35	4.13 ± 0.30

Table 9: Diagnostics for the field system with $K = 10^4$. $\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$ is the Monte Carlo approximation of the log-target density using the learned flow to generate samples; KSD U-stat and V-stat are the Kernel Stein Discrepancy U- and V-statistics between the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

i.e. $X \sim \mathcal{N}(\mu_0 1_d, \Sigma_0)$ for $1_d = [1, \ldots, 1]^T \in \mathbb{R}^d$ with dimension $d = 1600$. The chosen parameter values are $\sigma^2 = 1.91$, $\beta = 1/33$, and $\mu_0 = \log(126) - \sigma^2/2$, corresponding to the values estimated in [53]. The number of points in each grid cell $Y = \{Y_m\}_{m \in M} \in \mathbb{N}^{40 \times 40}$ are modelled as conditionally independent and Poisson distributed with means $a\Lambda_m$,

$$\mathcal{L}(Y|X) = \prod_{m \in [1:30]^2} \exp(x_m y_m - a \exp(x_m)), \tag{44}$$

where $a = 1/40^2$ represents the area of each grid cell. For this example, all methods use $N = 128$ parallel chains for training and 1024 hidden dimensions for all neural networks. Methods with a MALA kernel use a step size of 0.01, and methods with splines use 8 coupling layers with 8 bins and range limited to $[-10, 10]$. Results for $K = 10^4$ learning iterations are presented in Table 10. In the main text, we present results for $k_Q = 10^3$ for MFM. We were unable to run NF-MCMC and FAB for $K = 10^4$ iterations because of memory issues; instead, we present results for $K = 10^3$ iterations only for the models using discrete normalizing flows.

$K = 10^4$	$\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$	KSD U-stat.	KSD V-stat.	seconds
MFM $k_Q = K$	-1960 ± 10.8	1.13e-1±5.18e-2	28.1 ± 0.246	117 ± 4.19
MFM $k_Q = 10^3$	-1960 ± 10.8	1.14e-1±5.13e-2	28.1 ± 0.241	1690 ± 870
– w/ HTE	-1960 ± 12.2	1.12e-1±4.93e-2	28.1 ± 0.236	143 ± 14.5
MFM $k_Q = 10^2$	-1960 ± 12.6	1.15e-1±5.12e-2	28.0 ± 0.265	17300 ± 9970
– w/ HTE	-1960 ± 11.1	1.15e-1±5.17e-2	28.1 ± 0.239	394 ± 157
DDS	-1850 ± 8.59	7.59e-2±2.24e-2	24.7 ± 0.08	3260 ± 8.41
NF-MCMC	-1410 ± 53.8	11.8 ± 7.55	89.0 ± 238	215 ± 46.4
FAB	-3070 ± 80.7	1.55e-1±6.12e-2	52.3 ± 2.02	1040 ± 2.78
AT-SMC	-1910 ± 4.21	1.39e-2±1.14e-2	25.0 ± 0.12	6.11 ± 0.44

Table 10: Log Gaussian Cox point process diagnostics for $K = 10^4$ where $\mathbb{E}_{[\phi_1]_{\#}p_0} \log \pi$ is the Monte Carlo approximation of the log-target density using the learned flow to generate samples; KSD U-stat and V-stat are the Kernelized Stein discrepancy's U- and V-statistics between the target and samples generated from the learned flow. Results are averaged and empirical 95% confidence intervals over 10 independent runs.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and the introduction provide an accurate reflection of the theoretical, methodological, and experiment contributions in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The final section includes a discussion of the limitations of this work, including the computational efficiency of the proposed method, situations in which the proposed method may be outperformed by other baselines, and hyperparameters which may strongly influence the performance of the proposed method.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

 https://doi.org/10.52202/079017-3316

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Our theoretical result (Proposition 3.1 in Section 3) is accompanied by a full set of assumptions and a complete proof (in Appendix B). Any results upon which the proof relies are properly cited. In addition, all of the theorems, formulas, and proofs in the paper are numbered and cross-referenced.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper includes full details of the algorithm used to obtain the experimental results in the paper, including all of the hyper-parameter settings for each of the different numerical experiments. Our code is also made publicly available on GitHub (see Section 5).

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our code is made publicly available on GitHub (see Section 5).

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all of the training and test details, including hyperparameter settings, train-test splits, choice of optimizer, etc., necessary to understand and reproduce our experimental results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error bars for all of our numerical experiments (see Section 5). In the text, we provide full details regarding how these error bars are calculated, as well as the factors which vary between each experimental run.

https://doi.org/10.52202/079017-3316

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include full details of the type of compute workers used in the main text (see Appendix C). We also include details of the running time for each of the main numerical experiments, for both our proposed method and the relevant baselines.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with NeurIPS Code of Ethics in every respect, including with regards to *Potential Harms Caused by the Research Process*, *Societal Impact and Potential Harmful Consequences*, and *Impact Mitigation Measures*. All of the authors have read and reviewed the NeurIPS Code of Ethics to ensure that the research conducted in this paper conforms with the guidelines.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper contains foundational research on a new method for approximate statistical inference. It is not tied to any particular applications and, as such, these are not directly discussed in the main text.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not include any datasets or models associated with a high risk of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

https://doi.org/10.52202/079017-3316

Justification: The creators and original owners of assets used in the paper (e.g., code for relevant baselines) are properly credited (see Section 5). We both cite the original papers that produced these assets, and include URLs to the code.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release any new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing experiments or any research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing experiments or any research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-3316