MixFlows: principled variational inference via mixed flows

Zuheng Xu [1] **Naitong Chen** [1] **Trevor Campbell** [1]

Abstract

This work presents *mixed variational flows (MixFlows)*, a new variational family that consists of a mixture of repeated applications of a map to an initial reference distribution. First, we provide efficient algorithms for i.i.d. sampling, density evaluation, and unbiased ELBO estimation. We then show that MixFlows have MCMC-like convergence guarantees when the flow map is ergodic and measure-preserving, and provide bounds on the accumulation of error for practical implementations where the flow map is approximated. Finally, we develop an implementation of MixFlows based on uncorrected discretized Hamiltonian dynamics combined with deterministic momentum refreshment. Simulated and real data experiments show that MixFlows can provide more reliable posterior approximations than several black-box normalizing flows, as well as samples of comparable quality to those obtained from state-of-the-art MCMC methods.

1. Introduction

Bayesian statistical modelling and inference provides a principled approach to learning from data. However, for all but the simplest models, exact inference is not possible and computational approximations are required. A standard methodology for Bayesian inference is Markov chain Monte Carlo (MCMC) [Robert & Casella, 2004; Robert & Casella, 2011; Gelman et al., 2013, Ch. 11,12], which involves simulating a Markov chain whose stationary distribution is the Bayesian posterior distribution, and then treating the sequence of states as draws from the posterior. MCMC methods are supported by theory that guarantees that if one simulates the chain for long enough, Monte Carlo averages based on the sequence of states will converge to the exact posterior expectation of interest (e.g., (Roberts & Rosenthal, 2004)). This "exactness" property is quite compelling: regardless of how well one is able to tune the Markov chain, the method is guaranteed to eventually produce an accurate result given enough computation time. Nevertheless, it remains a challenge to assess and optimize the performance of MCMC in practice with a finite computational budget. One option is to use a Stein discrepancy (Gorham & Mackey, 2015; Liu et al., 2016; Chwialkowski et al., 2016; Gorham & Mackey, 2017; Anastasiou et al., 2021), which quantifies how well a set of MCMC samples approximates the posterior distribution. But standard Stein discrepancies are not reliable in the presence of multimodality (Wenliang & Kanagawa, 2020), are computationally expensive to estimate, and suffer from the curse of dimensionality, although recent work addresses the latter two issues to an extent (Huggins & Mackey, 2018; Gong et al., 2021). The other option is to use a traditional diagnostic, e.g., Gelman–Rubin (Gelman & Rubin, 1992; Brooks & Gelman, 1998), effective sample size (Gelman et al., 2013, p. 286), Geweke (1992), or others (Cowles & Carlin, 1996). These diagnostics detect mixing issues, but do not comprehensively quantify how well the MCMC samples approximate the posterior.

Variational inference (VI) (Jordan et al., 1999; Wainwright & Jordan, 2008; Blei et al., 2017) is an alternative to MCMC that does provide a straightforward quantification of posterior approximation error. In particular, VI involves approximating the posterior with a probability distribution—typically selected from a parametric family—that enables both i.i.d. sampling and density evaluation (Wainwright & Jordan, 2008; Rezende & Mohamed, 2015a; Ranganath et al., 2016; Papamakarios et al., 2021). Because one can both obtain i.i.d. draws and evaluate the density, one can estimate the ELBO (Blei et al., 2017), i.e., the Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951) to the posterior up to a constant. The ability to estimate this quantity, in turn, enables scalable tuning via straightforward stochastic gradient descent algorithms (Hoffman et al., 2013; Ranganath et al., 2014), optimally mixed approximations (Jaakkola & Jordan, 1998; Gershman et al., 2012; Zobay, 2014; Guo et al., 2016; Wang, 2016; Miller et al., 2017; Locatello et al., 2018b;a; Campbell & Li, 2019), model selection (Corduneau & Bishop, 2001; Masa-aki, 2001; Constantinopoulos et al., 2006; Ormerod et al., 2017; Chérief-Abdellatif & Alquier, 2018; Tao et al., 2018), and more. However,

[1]University of British Columbia, Department of Statistics, Vancouver, Canada. Correspondence to: Trevor Campbell <trevor@stat.ubc.ca>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

VI typically does not possess the same "exactness regardless of tuning" that MCMC does. The optimal variational distribution is not usually equal to the posterior due to the use of a limited parametric variational family; and even if it were, one could not reliably find it due to nonconvexity of the KL objective (Xu & Campbell, 2022). Recent work addresses this problem by constructing variational families from parametrized Markov chains targeting the posterior. Many are related to annealed importance sampling (Salimans et al., 2015; Wolf et al., 2016; Geffner & Domke, 2021; Zhang et al., 2021; Thin et al., 2021b; Jankowiak & Phan, 2021); these methods introduce numerous auxiliary variables and only have convergence guarantees in the limit of increasing dimension of the joint distribution. Those based on flows (Neal, 2005; Caterini et al., 2018; Chen et al., 2022) avoid the increase in dimension with flow length, but typically do not have guaranteed convergence to the target. Methods involving the final-state marginal of finite simulations of standard MCMC methods (e.g., Zhang & Hernández-Lobato, 2020) do not enable density evaluation.

The first contribution of this work is a new family of *mixed variational flows (MixFlows)*, constructed via averaging over repeated applications of a pushforward map to an initial reference distribution. We develop efficient methods for i.i.d. sampling, density evaluation, and unbiased ELBO estimation. **The second contribution** is a theoretical analysis of MixFlows. We show (Theorems 4.1 and 4.2) that when the map family is ergodic and measure-preserving, MixFlows converge to the target distribution for any value of the variational parameter, and hence have guarantees regardless of tuning as in MCMC. We then extend these results (Theorem 4.3 and Corollary 4.4) to MixFlows based on approximated maps—which are typically necessary in practice—with bounds on error with increasing flow length. **The third contribution** of the work is an implementation of MixFlows via uncorrected discretized Hamiltonian dynamics. Simulated and real data experiments compare performance to the No-U-Turn sampler (NUTS) (Hoffman & Gelman, 2014), standard Hamiltonian Monte Carlo (HMC) (Neal, 2011), and several black-box normalizing flows (Rezende & Mohamed, 2015b; Dinh et al., 2017). Our results demonstrate a comparable sample quality to NUTS, similar computational efficiency to HMC, and more reliable posterior approximations than standard normalizing flows.

Related work. Averages of Markov chain state distributions in general were studied in early work on shift-coupling (Aldous & Thorisson, 1993), with convergence guarantees established by Roberts & Rosenthal (1997). However, these guarantees involve minorization and drift conditions that are designed for stochastic Markov chains, and are not applicable to MixFlows. Averages of deterministic pushforwards specifically to enable density evaluation have also appeared in more recent work. Rotskoff & Vanden-Eijnden (2019); Thin et al. (2021a) use an average of pushforwards generated by simulating nonequilibrium dynamics as an importance sampling proposal. The proposal distribution itself does not come with any convergence guarantees—due to the use of non-measure-preserving, non-ergodic damped Hamiltonian dynamics—or tuning guidance. Our work provides comprehensive convergence theory and establishes a convenient means of optimizing hyperparameters.

MixFlows are also related to past work on deterministic MCMC (Murray & Elliott, 2012; Neal, 2012; ver Steeg & Galstyan, 2021; Neklyudov et al., 2021). Murray & Elliott (2012) developed a Markov chain Monte Carlo procedure based on an arbitrarily dependent random stream via augmentation and measure-preserving bijections. ver Steeg & Galstyan (2021) designed a specialized momentum distribution that generates valid Monte Carlo samples solely through the simulation of deterministic Hamiltonian dynamics. Neklyudov et al. (2021) proposed a general form of measure-preserving dynamics that can be utilized to construct deterministic Gibbs samplers. These works all involve only deterministic updates, but do not construct variational approximations, provide total variation convergence guarantees, or provide guidance on hyperparameter tuning. Finally, some of these works involve discretization of dynamical systems, but do not characterize the resultant error (ver Steeg & Galstyan, 2021; Neklyudov et al., 2021). In contrast, our work provides a comprehensive convergence theory, with error bounds for when approximate maps are used.

2. Background

2.1. Variational inference with flows

Consider a set $\mathcal{X} \subseteq \mathbb{R}^d$ and a target probability distribution π on $\mathcal{X}$ whose density with respect to the Lebesgue measure we denote $\pi(x)$ for $x \in \mathcal{X}$. In the setting of Bayesian inference, π is the posterior distribution that we aim to approximate, and we are only able to evaluate a function $p(x)$ such that $p(x) = Z \cdot \pi(x)$ for some unknown normalization constant $Z > 0$. Throughout, we will assume all distributions have densities with respect to the Lebesgue measure on $\mathcal{X}$, and will use the same symbol to denote a distribution and its density; it will be clear from context what is meant.

Variational inference involves approximating the target distribution π by minimizing the Kullback-Leibler (KL) divergence from π to members of a parametric family $\{q_\lambda : \lambda \in \Lambda\}$, $\Lambda \subseteq \mathbb{R}^p$, i.e.,

$$\begin{aligned} \lambda^\star &= \arg\min_{\lambda \in \Lambda} \mathrm{D}_{\mathrm{KL}}\left(q_\lambda || \pi\right) \\ &= \arg\min_{\lambda \in \Lambda} \int q_\lambda(x) \log \frac{q_\lambda(x)}{p(x)} \mathrm{d}x. \end{aligned} \tag{1}$$

The two objective functions in Equation (1) differ only by

the constant $\log Z$. In order to be able to optimize λ using standard techniques, the variational family q_λ, $\lambda \in \Lambda$ must enable both i.i.d. sampling and density evaluation. A common approach to constructing such a family is to pass draws from a simple reference distribution q_0 through a measurable function $T_\lambda : \mathcal{X} \to \mathcal{X}$; T_λ is often referred to as a *flow* when comprised of repeated composed functions (Tabak & Turner, 2013; Rezende & Mohamed, 2015a; Kobyzev et al., 2021). If T_λ is a *diffeomorphism*, i.e., differentiable and has a differentiable inverse, then we can express the density of $X = T_\lambda(Y)$, $Y \sim q_0$ as

$$\forall x \in \mathcal{X},\ q_\lambda(x) = \frac{q_0(T_\lambda^{-1}(x))}{J_\lambda(T_\lambda^{-1}(x))},\ J_\lambda(x) = |\det \nabla_x T_\lambda(x)|\,.$$

In this case the optimization in Equation (1) can be rewritten using a transformation of variables as

$$\lambda^\star = \arg\min_{\lambda \in \Lambda} \int q_0(x) \log \frac{q_0(x)}{J_\lambda(x) p(T_\lambda(x))} \mathrm{d}x. \quad (2)$$

One can solve the optimization problem Equation (2) using unbiased stochastic estimates of the gradient[1] with respect to λ based on draws from q_0 (Salimans & Knowles, 2013; Kingma & Welling, 2014),

$$\nabla_\lambda \mathrm{D}_{\mathrm{KL}}\left(q_\lambda || \pi\right) \approx \nabla_\lambda \log \frac{q_0(X)}{J_\lambda(X) p(T_\lambda(X))},\ X \sim q_0.$$

2.2. Measure-preserving and ergodic maps

Variational flows are often constructed from a flexible, general-purpose parametrized family $\{T_\lambda : \lambda \in \Lambda\}$ that is not specialized for any particular target distribution (Papamakarios et al., 2021); it is the job of the KL divergence minimization Equation (2) to adapt the parameter λ such that q_λ becomes a good approximation of the target π. However, there are certain functions—in particular, those that are both *measure-preserving* and *ergodic* for π—that naturally provide a means to approximate expectations of interest under π without the need for tuning. Intuitively, a measure-preserving map T will not change the distribution of draws from π: if $X \sim \pi$, then $T(X) \sim \pi$. And an ergodic map T, when applied repeatedly, will not get "stuck" in a subset of $\mathcal{X}$ unless it has probability either 0 or 1 under π. The precise definitions are given in Definitions 2.1 and 2.2.

Definition 2.1 (Measure-preserving map (Eisner et al., 2015, pp. 73, 105))**.** $T : \mathcal{X} \to \mathcal{X}$ is *measure-preserving* for π if $T\pi = \pi$, where $T\pi$ is the pushforward measure given by $\pi(T^{-1}(A))$ for each measurable set $A \subseteq \mathcal{X}$.

Definition 2.2 (Ergodic map (Eisner et al., 2015, pp. 73, 105))**.** $T : \mathcal{X} \to \mathcal{X}$ is *ergodic* for π if for all measurable sets $A \subseteq \mathcal{X}$, $T(A) = A$ implies that $\pi(A) \in \{0, 1\}$.

[1] We assume throughout that differentiation and integration can be swapped wherever necessary.

If a map T satisfies both Definitions 2.1 and 2.2, then long-run averages resulting from repeated applications of T will converge to expectations under π, as shown by Theorem 2.3. When $\mathcal{X}$ is compact, this result shows that the discrete measure $\frac{1}{N}\sum_{n=0}^{N-1} \delta_{T^n x}$ converges weakly to π (Dajani & Dirksin, 2008, Theorem 6.1.7).

Theorem 2.3 (Ergodic Theorem [Birkhoff, 1931; Eisner et al., 2015, p. 212])**.** *Suppose $T : \mathcal{X} \to \mathcal{X}$ is measure-preserving and ergodic for π, and $f \in L^1(\pi)$. Then*

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} f(T^n x) = \int f \mathrm{d}\pi \qquad \pi\text{-a.e. } x \in \mathcal{X}.$$

Based on this result, one might reasonably consider building a measure-preserving variational flow, i.e., $X = T(X_0)$ where $X_0 \sim q_0$. However, it is straightforward to show that measure-preserving bijections T do not decrease the KL divergence (or any other f-divergence, e.g., total variation and Hellinger (Qiao & Minematsu, 2010, Theorem 1)):

$$\mathrm{D}_{\mathrm{KL}}\left(Tq || \pi\right) = \mathrm{D}_{\mathrm{KL}}\left(Tq || T\pi\right) = \mathrm{D}_{\mathrm{KL}}\left(q || \pi\right).$$

3. Mixed variational flows (MixFlows)

In this section, we develop a general family of *mixed variational flows* (MixFlows) as well as algorithms for tractable i.i.d. sampling, density evaluation, and ELBO estimation. MixFlows consist of a mixture of normalizing flows obtained via repeated application of a pushforward map. This section introduces general MixFlows; later in Sections 4 and 5 we will provide convergence guarantees and examples based on Hamiltonian dynamics.

3.1. Variational family

Define a reference distribution q_0 on $\mathcal{X}$ for which i.i.d. sampling and density evaluation is tractable, and a collection of measurable functions $T_\lambda : \mathcal{X} \to \mathcal{X}$ parametrized by $\lambda \in \Lambda$. Then the MixFlow family generated by q_0 and T_λ is

$$q_{\lambda,N} = \frac{1}{N} \sum_{n=0}^{N-1} T_\lambda^n q_0 \qquad \text{for} \qquad \lambda \in \Lambda,\ N \in \mathbb{N},$$

where $T_\lambda^n q_0$ denotes the pushforward of the distribution q_0 under n repeated applications of T_λ.

3.2. Density evaluation and sampling

If $T_\lambda : \mathcal{X} \to \mathcal{X}$ is a diffeomorphism with Jacobian $J_\lambda : \mathcal{X} \to \mathbb{R}$, we can express the density of $q_{\lambda,N}$ by using a transformation of variables formula on each component in the mixture:

$$q_{\lambda,N}(x) = \frac{1}{N} \sum_{n=0}^{N-1} \frac{q_0(T_\lambda^{-n} x)}{\prod_{j=1}^n J_\lambda(T_\lambda^{-j} x)}.$$

This density can be computed efficiently using $N-1$ evaluations of T_λ^{-1} and J_λ each (Algorithm 2). For sampling, we can obtain an independent draw $X \sim q_{\lambda,N}$ by treating $q_{\lambda,N}$ as a mixture of N distributions:

$$K \sim \mathsf{Unif}\{0,1,\ldots,N-1\} \quad X_0 \sim q_0 \quad X = T_\lambda^K(X_0).$$

The procedure is given in Algorithm 1. On average, this computation requires $\mathbb{E}[K] = \frac{N-1}{2}$ applications of T_λ, and at most it requires $N-1$ applications. However, one often takes samples from $q_{\lambda,N}$ to estimate the expectation of some test function $f : \mathcal{X} \to \mathbb{R}$. In this case, one can use all intermediate states over a single pass of a trajectory rather than individual i.i.d. draws. In particular, we obtain an unbiased estimate of $\int f(x) q_{\lambda,N}(\mathrm{d}x)$ via

$$X_0 \sim q_0 \quad \frac{1}{N}\sum_{n=0}^{N-1} f(T_\lambda^n X_0). \tag{3}$$

We refer to this estimate as the *trajectory-averaged estimate* of f. The trajectory-averaged estimate of f is preferred over the naïve estimate based on a single draw $f(X), X \sim q_{\lambda,N}$, as its cost is of the same order ($N-1$ applications of T_λ) and its variance is bounded above by that of the naïve estimate $f(X)$, as shown in Proposition 3.1. See Appendix E.2, and Figure 6 in particular, for empirical verification of Proposition 3.1.

Proposition 3.1.

$$\mathrm{Var}\left[\frac{1}{N}\sum_{n=0}^{N-1} f(T^n X_0)\right] \le \mathrm{Var}\left[f(X)\right].$$

3.3. ELBO estimation

We can minimize the KL divergence from $q_{\lambda,N}$ to π by maximizing the ELBO (Blei et al., 2017), given by

$$\begin{aligned}\mathrm{ELBO}\,(\lambda, N) &= \int q_{\lambda,N}(x) \log \frac{p(x)}{q_{\lambda,N}(x)} \mathrm{d}x \\ &= \int q_0(x) \left(\frac{1}{N}\sum_{n=0}^{N-1} \log \frac{p(T_\lambda^n x)}{q_{\lambda,N}(T_\lambda^n x)} \mathrm{d}x \right).\end{aligned}$$

The trajectory-averaged ELBO estimate is thus

$$X_0 \sim q_0,\ \widehat{\mathrm{ELBO}}(\lambda, N) = \frac{1}{N}\sum_{n=0}^{N-1} \log \frac{p(T_\lambda^n X_0)}{q_{\lambda,N}(T_\lambda^n X_0)}. \tag{4}$$

The naïve method to compute this estimate—sampling X_0 and then computing the log density ratio for each term—requires $O(N^2)$ computation because each evaluation of $q_{\lambda,N}(x)$ is $O(N)$. Algorithm 3 provides an efficient way of computing $\widehat{\mathrm{ELBO}}(\lambda, N)$ in $O(N)$ operations, which is on par with taking a single draw from $q_{\lambda,N}$ or evaluating $q_{\lambda,N}(x)$ once. The key insight in Algorithm 3 is that we can evaluate the collection of values $\{q_{\lambda,N}(X_0), q_{\lambda,N}(T_\lambda X_0), \ldots, q_{\lambda,N}(T_\lambda^{N-1} X_0)\}$ incrementally, starting from $q_{\lambda,N}(X_0)$ and iteratively computing each $q_{\lambda,N}(T_\lambda^n X_0)$ for increasing n in constant time. Specifically, in practice one computes $q_{\lambda,N}(x)$ with a complexity of $O(N)$ and iteratively updates $q_{\lambda,N}(T_\lambda^k x)$ and the Jacobian for $k = 1, 2, \ldots N-1$. Each update requires only constant cost if one precomputes and stores $T_\lambda^{-N+1}x, \ldots, T_\lambda^{N-1}x$ and $J_\lambda(T_\lambda^{-N+1}x), \ldots, J_\lambda(T_\lambda^{N-1}x)$. This then implies that Algorithm 3 requires $O(N)$ memory; Algorithm 5 in Appendix B provides a slightly more complicated $O(N)$ time, $O(1)$ memory implementation of Equation (4).

Algorithm 1 `Sample`$(q_{\lambda,N})$: Take a draw from $q_{\lambda,N}$

Input: reference distribution q_0, flow map T_λ, number of steps N
$K \leftarrow$ `Sample`$(\mathsf{Unif}\{0,1,\ldots,N-1\})$
$x_0 \leftarrow$ `Sample`(q_0)
Return: $T_\lambda^K(x_0)$

Algorithm 2 $\log q_{\lambda,N}(x)$: Evaluate the log-density of $q_{\lambda,N}$

Input: location x, reference distribution q_0, flow map T_λ, Jacobian J_λ, number of steps N
$L \leftarrow 0$
$w_0 \leftarrow \log q_0(x)$
for $n = 1, \ldots, N-1$ **do**
  $x \leftarrow T_\lambda^{-1}(x)$
  $L \leftarrow L + \log J_\lambda(x)$
  $w_n \leftarrow \log q_0(x) - L$
end for
Return: `LogSumExp`$(w_0, \ldots, w_{N-1}) - \log N$

4. Guarantees for MixFlows

In this section, we show that when T_λ is measure-preserving and ergodic for π—or approximately so—MixFlows come with convergence guarantees and bounds on their approximation error as a function of flow length. Proofs for all results may be found in Appendix A.

4.1. Ergodic MixFlows

Ergodic MixFlow families are those where $\forall \lambda \in \Lambda$, T_λ is measure-preserving and ergodic for π. In this setting, Theorems 4.1 and 4.2 show that MixFlows converge to the target weakly and in total variation as $N \to \infty$ regardless of the choice of λ (i.e., regardless of parameter tuning effort). Thus, ergodic MixFlow families provide the same compelling convergence result as MCMC, but with the added benefit of unbiased ELBO estimates. We first demonstrate setwise and weak convergence. Recall that a sequence of

Algorithm 3 `EstELBO`(λ, N): Obtain an unbiased estimate of the ELBO for $q_{\lambda,N}$.

Input: reference q_0, unnormalized target p, flow map T_λ, Jacobian J_λ, number of flow steps N
$x_0 \leftarrow \texttt{Sample}(q_0), \quad J_0 \leftarrow J_\lambda(x_0)$
for $n = 1, \ldots, N-1$ **do**
  $x_n \leftarrow T_\lambda(x_{n-1}), \quad J_n \leftarrow J_\lambda(x_n)$
  $x_{-n} \leftarrow T_\lambda^{-1}(x_{-n+1}), \quad J_{-n} \leftarrow J_\lambda(x_{-n})$
end for
$z_0 \leftarrow q_{\lambda,N}(x_0)$ (via Algorithm 2)
$J \leftarrow \prod_{j=1}^{N-1} J_{-j}$
for $n = 1, \ldots, N-1$ **do**
  $\bar{q}_n \leftarrow \frac{1}{N} q_0(x_{-N+n})/J$
  $z_n \leftarrow (z_{n-1} - \bar{q}_n)/J_{n-1} + \frac{1}{N} q_0(x_n)$
  if $n < N-1$ **then**
    $J \leftarrow J \cdot J_{n-1}/J_{-N+n}$
  end if
end for
$\widehat{\mathrm{ELBO}}(\lambda, N) \leftarrow \frac{1}{N}\sum_{n=0}^{N-1}(\log p(x_n) - \log z_n)$
Return: $\widehat{\mathrm{ELBO}}(\lambda, N)$

distributions q_n converges *weakly* to π if for all bounded, continuous $f : \mathcal{X} \to \mathbb{R}$, $\lim_{n\to\infty} \int f(x) q_n(\mathrm{d}x) = \int f(x)\pi(\mathrm{d}x)$, and converges *setwise* to π if for all measurable $A \subseteq \mathcal{X}$, $\lim_{n\to\infty} q_n(A) = \pi(A)$.

Theorem 4.1. *Suppose $q_0 \ll \pi$ and T_λ is measure-preserving and ergodic for π. Then $q_{\lambda,N}$ converges both setwise and weakly to π as $N \to \infty$.*

Using Theorem 4.1 as a stepping stone, we can obtain convergence in total variation. Recall that a sequence of distributions q_n converges in *total variation* to π if $\mathrm{D}_{\mathrm{TV}}(q_n, \pi) = \sup_A |q_n(A) - \pi(A)| \to 0$ as $n \to \infty$. Note that similar nonasymptotic results exist for the ergodic average law of Markov chains (Roberts & Rosenthal, 1997), but as previously mentioned, these results do not apply to deterministic Markov kernels.

Theorem 4.2. *Suppose $q_0 \ll \pi$ and T_λ is measure-preserving and ergodic for π. Then $q_{\lambda,N}$ converges in total variation to π as $N \to \infty$.*

4.2. Approximated MixFlows

In practice, it is rare to be able to evaluate a measure-preserving map exactly; but *approximations* are commonly available. For example, in Section 5 we will create a measure-preserving map using Hamiltonian dynamics, and then approximate that map with a discretized leapfrog integrator. We therefore need to extend the result in Section 4.1 to apply to approximated maps.

Suppose we have two maps, T and $\hat{T}$, with corresponding MixFlow families q_n and $\hat{q}_n$ (suppressing λ for brevity). Theorem 4.3 shows that the error of the MixFlow family $\hat{q}_N$ reflects an accumulation of the difference between the pushforward of each q_n under $\hat{T}$ and T.

Theorem 4.3. *Suppose $\hat{T}$ is a bijection. Then*

$$\mathrm{D}_{\mathrm{TV}}(\hat{q}_N, \pi) \le \mathrm{D}_{\mathrm{TV}}(q_N, \pi) + \sum_{n=0}^{N-1} \frac{n}{N} \mathrm{D}_{\mathrm{TV}}\left(Tq_n, \hat{T}q_n\right).$$

Suppose T is measure-preserving and ergodic for π and $q_0 \ll N$. Then Theorem 4.2 implies that $\mathrm{D}_{\mathrm{TV}}(q_N, \pi) \to 0$, and for the second term, we (very loosely) expect that $\mathrm{D}_{\mathrm{TV}}(Tq_n, \hat{T}q_n) \approx \mathrm{D}_{\mathrm{TV}}(\pi, \hat{T}\pi)$ for large n. Therefore, the second term should behave like $O(N \cdot \mathrm{D}_{\mathrm{TV}}(\pi, \hat{T}\pi))$, i.e., increase linearly in N proportional to how "non-measure-preserving" $\hat{T}$ is. This presents a trade-off between flow length and approximation quality: better approximations enable longer flows and lower minimal error bounds. Our empirical findings in Section 6 generally confirm this behavior. Corollary 4.4 further provides a more explicit characterization of this trade-off when $\hat{T}$ and its log-Jacobian $\log \hat{J}$ are uniformly close to their exact counterparts T and $\log J$, and $\log q_n$ and $\log J$ are uniformly (over $n \in \mathbb{N}$) Lipschitz continuous. The latter assumption is reasonable in practical settings where we observe that the log-density of q_n closely approximates π (see the experimental results in Section 6).

Corollary 4.4. *Suppose that for all $x \in \mathcal{X}$,*

$$\max\left\{\|\hat{T}^{-1}x - T^{-1}x\|, |\log \hat{J}(x) - \log J(x)|\right\} \le \epsilon,$$

for all $n \in [N-1]$, $\log q_n$ and $\log J$ are ℓ-Lipschitz continuous, and that $T, \hat{T}$ are diffeomorphisms. Then

$$\mathrm{D}_{\mathrm{TV}}(\hat{q}_N, \pi) \le \mathrm{D}_{\mathrm{TV}}(q_N, \pi) + N\epsilon(\ell+1)e^{(2\ell+1)\epsilon}.$$

This result states that the overall map-approximation error is $O(N\epsilon)$ when ϵ is small. Theorem 1 of Butzer & Westphal (1971) also suggests that $\mathrm{D}_{\mathrm{TV}}(q_N, \pi) = O(1/N)$ in many cases, which hints that the bound should decrease roughly until $N = O(1/\sqrt{\epsilon})$, with error bound $O(\sqrt{\epsilon})$. We leave a more careful investigation of this trade-off for future work.

4.3. Discussion

Our main convergence results (Theorems 4.1 and 4.2) require that π dominates the reference q_0, and that T is both measure-preserving and ergodic. It is often straightforward to design a dominated reference q_0. Designing a measure-preserving map T with an implementable approximation $\hat{T} \approx T$ is also often feasible. However, verifying the ergodicity of T conclusively is typically a very challenging task. As a consequence, most past work involving measure-preserving transformations just asserts the *ergodic hypothesis* without proof; see the discussions in ver Steeg & Galstyan (2021, p. 4) and Tupper (2005, p. 2-3).

But because MixFlows provide the ability to estimate the Kullback-Leibler divergence up to a constant, it is not critical to prove convergence a priori (as it is in the case of MCMC, for example). Instead, we suggest using the results of Theorems 4.1 to 4.3 and Corollary 4.4 as a guiding recipe for constructing principled variational families. First, start by designing a family of measure-preserving maps T_λ, $\lambda \in \Lambda$. Next, approximate T_λ with some tractable $T_{\lambda,\epsilon}$ including a tunable fidelity parameter $\epsilon \geq 0$, such as that $\epsilon \to 0$, $T_{\lambda,\epsilon}$ becomes closer to T_λ. Finally, build a MixFlow from $T_{\lambda,\epsilon}$, and tune both λ and ϵ by maximizing the ELBO. We follow this recipe in Section 5 and verify it empirically in Section 6.

5. Uncorrected Hamiltonian MixFlows

In this section, we provide an example of how to design a MixFlow by starting from an exactly measure-preserving map and then creating a tunable approximation to it. The construction is inspired by Hamiltonian Monte Carlo (HMC) (Neal, 2011; 1996), in which each iteration involves simulating Hamiltonian dynamics followed by a stochastic momentum refreshment; our method replaces the stochastic refreshment with a deterministic transformation. In particular, consider the *augmented* target density on $\mathcal{X} \times \mathbb{R}^d \times [0,1]$,

$$\bar{\pi}(x,\rho,u) = \pi(x)m(\rho)\mathbb{1}[0 \leq u \leq 1], \quad m(\rho) = \prod_{i=1}^{d} r(\rho_i),$$

with auxiliary variables $\rho \in \mathbb{R}^d$, $u \in [0,1]$, and some almost-everywhere differentiable univariate probability density $r : \mathbb{R} \to \mathbb{R}_+$. The x-marginal distribution of $\bar{\pi}$ is the original target distribution π

5.1. Measure-preserving map via Hamiltonian dynamics

We construct T_λ by composing the following three steps, which are all measure-preserving for $\bar{\pi}$:

(1) Hamiltonian dynamics We first apply

$$(x', \rho') \leftarrow H_L(x, \rho),$$

where $H_L : \mathcal{X} \times \mathbb{R}^d \to \mathcal{X} \times \mathbb{R}^d$ is the map of Hamiltonian dynamics with position x and momentum ρ simulated for a time interval of length $L \in \mathbb{R}_+$,

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = \nabla \log \pi(x) \qquad \frac{\mathrm{d}x}{\mathrm{d}t} = -\nabla \log m(\rho). \tag{5}$$

One can show that this preserves density (and hence is measure preserving) and is also unit Jacobian (Neal, 2011).

(2) Pseudotime shift Second, we apply a constant shift to the pseudotime variable u,

$$u' \leftarrow u + \xi \mod 1,$$

where $\xi \in \mathbb{R}$ is a fixed irrational number (say, $\xi = \pi/16$). As this is a constant shift, it is unit Jacobian and density-preserving (and hence measure-preserving). The u component will act as a notion of "time" of the flow, and ensures that the refreshment of ρ in step (3) below will take a different form even if x visits the same location again.

(3) Momentum refreshment Finally, we refresh each of the momentum variables via

$$\forall i = 1, \ldots, d, \quad \rho_i'' \leftarrow R^{-1}(R(\rho_i') + z(x_i', u') \mod 1),$$

where R is the cumulative distribution function (CDF) of density r, and $z : \mathcal{X} \times [0,1] \to \mathbb{R}$ is any differentiable function; this generalizes the map from Neal (2012); Murray & Elliott (2012) to enable the shift to depend on the state x and pseudotime u. This step (3) is an attempt to replicate the independent resampling of $\rho \sim m$ from HMC using only deterministic maps. This map is measure-preserving as it involves mapping the momentum to a $\mathsf{Unif}[0,1]$ random variable via the CDF, shifting by an amount that depends only on x, u, and then mapping back using the inverse CDF. The Jacobian is the momentum density ratio $m(\rho')/m(\rho'')$.

5.2. Approximation via uncorrected leapfrog integration

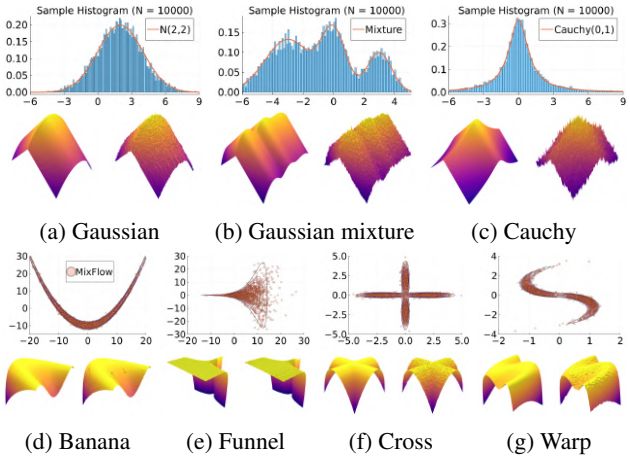
In practice, we cannot simulate the dynamics in step (1) perfectly. Instead, we approximate the dynamics in Equation (5) by running L steps of the leapfrog method, where each leapfrog map $\hat{H}_\epsilon : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$ involves interleaving three discrete transformations with step size $\epsilon > 0$,

$$\begin{aligned} \hat{\rho}_{k+1} &= \rho_k + \frac{\epsilon}{2}\nabla \log \pi(x_k) \\ x_{k+1} &= x_k - \epsilon \nabla \log m(\hat{\rho}_{k+1}) \\ \rho_{k+1} &= \hat{\rho}_{k+1} + \frac{\epsilon}{2}\nabla \log \pi(x_{k+1}) \end{aligned} .$$

Denote the map $T_{\lambda,\epsilon}$ to be the composition of the three steps with Hamiltonian dynamics replaced by the leapfrog integrator; Algorithm 6 in Appendix C provides the pseudocode. The final variational tuning parameters for the MixFlow are the step size ϵ—which controls how close $T_{\lambda,\epsilon}$ is to being measure-preserving for $\bar{\pi}$—the Hamiltonian simulation length $L \in \mathbb{N}$, and the flow length N. In our experiments we tune the number of leapfrog steps L and the step size ϵ by maximizing the ELBO. We also tune the number of refreshments N to achieve a desirable computation-quality tradeoff by visually inspecting the convergence of the ELBO.

5.3. Numerical stability

Density evaluation (Algorithm 2) and ELBO estimation (Algorithm 3) both involve repeated applications of T_λ and T_λ^{-1}. This poses no issue in theory, but in a computer—with



(a) Gaussian (b) Gaussian mixture (c) Cauchy

(d) Banana (e) Funnel (f) Cross (g) Warp

Figure 1. Marginal samples (first row) and pairs of exact and approximate joint log density (second row) for Gaussian (1a), mixture (1b), and Cauchy (1c) targets. Marginal samples (third row), pairs of sliced exact and approximate joint log density (fourth row) for banana (1d), funnel (1e), cross (1f), and warped Gaussian (1g).

floating point computations—one should code the map T_λ and its inverse T_λ^{-1} in a numerically precise way such that $(T_\lambda^K) \circ (T_\lambda^{-K})$ is the identity map for large $K \in \mathbb{N}$. Figure 8 in Appendix E.2 displays the severity of the numerical error when T_λ and T_λ^{-1} are not carefully implemented. In practice, we check the stability limits of our flow by taking draws from q_0 and evaluating T_λ^K followed by T_λ^{-K} (and vice versa) for increasing K until we cannot reliably invert the flow. See Figure 7 in the appendix for an example usage of this diagnostic. Note that for sample generation specifically (Algorithm 1), numerical stability is not a concern as it only requires forward evaluation of the map T_λ.

6. Experiments

In this section, we demonstrate the performance of our method (`MixFlow`) on 7 synthetic targets and 7 real data targets. See Appendix E for the details of each target. Both our synthetic and real data examples are designed to cover a range of challenging features such as heavy tails, high dimensions, multimodality, and weak identifiability. We compare posterior approximation quality to several black-box normalizing flow methods (`NF`): `PlanarFlow`, `RadialFlow`, and `RealNVP` with various architectures (Papamakarios et al., 2021). To make the methods comparable via the ELBO, we train all `NF`s on the same joint space as `MixFlow`. We also compare the marginal sample quality of `MixFlow` against 5,000 samples from `NUTS` and `NF`s. Finally, we compare sampling time with all competitors, and effective sample size (ESS) per second with `HMC`. For all experiments, we use the standard Laplace distribution as the momentum distribution due to its numerical stability (see Figure 7 in Appendix E). Additional comparisons to variational inference based on uncorrected Hamiltonian annealing (`UHA`) (Geffner & Domke, 2021) and nonequilibrium orbit sampling (`NEO`) (Thin et al., 2021a, Algorithm 2) may be found in Appendix E. All experiments were conducted on a machine with an AMD Ryzen 9 3900X and 32GB of RAM. Code is available at https://github.com/zuhengxu/Ergodic-variational-flow-code.

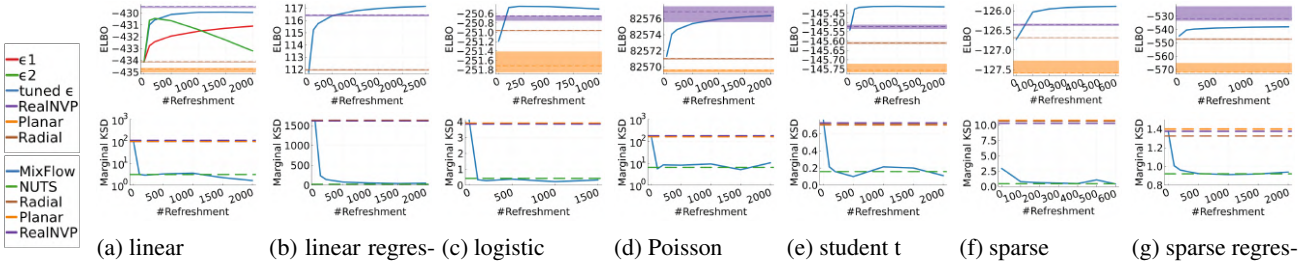
6.1. Qualitative assessment

We begin with a qualitative examination of the i.i.d. samples and the approximated targets produced by `MixFlow` initialized at $q_0 = \mathcal{N}(0, 1)$ for three one-dimensional synthetic distributions: a Gaussian, a mixture of Gaussians, and the standard Cauchy. We excluded the pseudotime variable u here in order to visualize the full joint density of (x, ρ) in 2 dimensions. More details can be found in Appendix E.1. Figures 1a to 1c show histograms of 10,000 i.i.d. x-marginal samples generated by `MixFlow` for each of the three targets, which nearly perfectly match the true target marginals. Figures 1a to 1c also show that $\log q_{\lambda,N}$ is generally a good approximation of the log target density.

We then present similar visualizations on four more challenging synthetic target distributions: the banana (Haario et al., 2001), Neal's funnel (Neal, 2003), a cross-shaped Gaussian mixture, and a warped Gaussian. All four examples have a 2-dimensional state $x \in \mathbb{R}^2$, and hence $(x, \rho, u) \in \mathbb{R}^5$. In each example we set the initial distribution q_0 to be the mean-field Gaussian approximation. More details can be found in Appendix E.2. Figures 1d to 1g shows the scatter plots consisting of 1,000 i.i.d. x-marginal samples drawn from `MixFlow`, as well as the approximated `MixFlow` log density and exact log density sliced as a function of $x \in \mathbb{R}^2$ for a single value of (ρ, u) chosen randomly via $(\rho, u) \sim \mathsf{Lap}(0, I) \times \mathsf{Unif}[0, 1]$ (which is required for visualization, as $(x, \rho, u) \in \mathbb{R}^5$). We see that, qualitatively, both the samples and approximated densities from `MixFlow` closely match the target. Finally, Figure 9 in Appendix E.2 provides a more comprehensive set of sample histograms (showing the x-, ρ-, and u-marginals). These visualizations support our earlier theoretical analysis.

6.2. Posterior approximation quality

Next, we provide a quantitative comparison of `MixFlow`, `NF`s, `NUTS` on 7 real data experiments outlined in Appendix E.5. We tune each `NF` method under various settings (Tables 1 and 2 and Appendix E.5.5), and present the best one for each example. ELBOs of `MixFlow` are estimated with Algorithm 3, averaging over 1,000 independent trajectories. ELBOs of `NF`s are based on 2,000 samples. To obtain an assessment for the target marginal distribution itself (not the augmented target), we also compare methods using the



(a) linear regression (b) linear regression (heavy) (c) logistic regression (d) Poisson regression (e) student t regression (f) sparse regression (g) sparse regression (high dim)

Figure 2. ELBO and KSD comparison for real data examples. Figure 2a displays the effect of step size for the linear regression problem: $\epsilon_1 = 0.0001, \epsilon_2 = 0.001$ and tuned $\epsilon = 0.0005$; see Figure 17 for step sizes for all other experiments. Lines indicate the median, and error regions indicate 25th to 75th percentile from 5 runs. Figure 2b does not include ELBOs of `PlanarFlow` as values are significantly worse than all other methods and are hard to visualize (see its ELBOs in Figure 17b).

kernel Stein discrepancy (KSD) with an inverse quadratic (IMQ) kernel (Gorham & Mackey, 2017). `NUTS` and `NF`s use 5,000 samples for KSD estimation, while `MixFlow` is based on 2,000 i.i.d. draws (Algorithm 1). For KSD comparisons, all variational methods are tuned by maximizing the ELBO (Figures 2 and 17).

Augmented target distribution Figure 2 displays the ELBO comparison. First, Figure 2a shows how different step sizes ϵ and number of refreshments N affects approximation quality. Overly large step sizes typically result in errors due to the use of discretized Hamiltonian dynamics, while overly small step sizes result in a flow that makes slow progress towards the target. `MixFlow` with a tuned step size generally shows a comparable joint ELBO value to the best `NF` method, yielding a competitive target approximation. Similar comparisons and assessment of the effect of step size for the synthetic examples are presented in Figures 5, 10 and 16. Note that in three examples (Figures 2a, 2d and 2g), the tuned `RealNVP` ELBO exceeds `MixFlow` by a small amount; but this required expensive architecture search and parameter tuning, and as we will describe next, `MixFlow` is actually more reliable in terms of target marginal sample quality and density estimation.

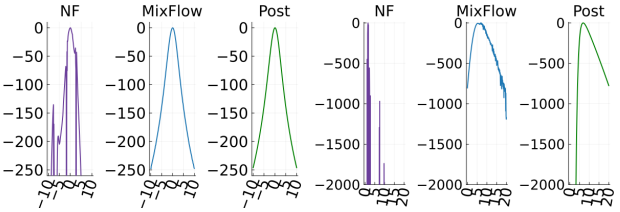
Original target distribution The second row of Figure 2 displays a comparison of KSD for the target distribution itself (instead of the augmented target). In particular, `MixFlow` produces comparable sample quality to that from `NUTS`—an exact MCMC method—and clearly outperforms all of the `NF` competitors. The scatter plots of samples in Figure 20 confirm the improvement in sample quality of `MixFlow` over variational competitors. Further, Figure 3 shows the (sliced) densities on two difficult real data examples: Bayesian student-t regression (with a heavy-tailed posterior), and a high-dimensional sparse regression (parameter dimension is 84). This result demonstrates that the densities provided by `MixFlow` more closely match those of the original target distribution than those of the best `NF`. Notice that `MixFlow` accurately captures the skew and heavy tails of the exact target, while the `NF` density fails to do so and contains many spurious modes.

6.3. Ease of tuning

In order to tune `MixFlow`, we simply run a 1-dimensional parameter sweep for the step size ϵ, and use a visual inspection of the ELBO to set an appropriate number of flow steps N. Tuning an `NF` requires optimizing its architecture, number of layers, and its (typically many) parameters. Not only is this time consuming—in our experiments, tuning took 10 minutes to roughly 1 hour (Figure 4)—but the optimization can also behave in unintuitive ways. For example, performance can be heavily dependent on the number of flow layers, and adding more layers does not necessarily improve quality. Figure 16, Tables 1 and 2, and Appendix E.5.5 show that using more layers does not necessarily help, and slows tuning considerably. In the case of `RealNVP` specifically, tuning can be unstable, especially for more complex models. The optimizer often returns NaN values for flow parameters during training (see Table 1). This instability has been noted in earlier work (Dinh et al., 2017, Sec. 3.7).

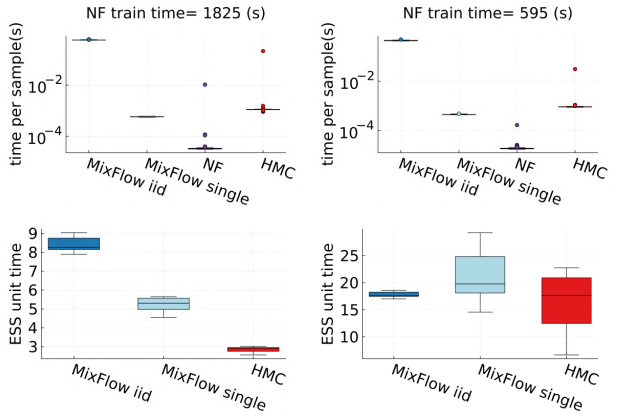
6.4. Time efficiency

Finally, Figure 4 presents timing results for two of the real data experiments (additional comparisons in Figures 14 and 19). In this figure, we use `MixFlow iid` to refer to i.i.d. sampling from `MixFlow`, and `MixFlow single` to refer to collecting all intermediate states on a single trajectory. This result shows that the per sample time of `MixFlow single` is similar to `NUTS` and `HMC` as one would expect. `MixFlow iid` is the slowest because each sample is generated by passing through the entire flow. The `NF` generates the fastest draws, but recall that this comes at the cost of significant initial tuning time; in the time it takes `NF` to generate its first sample, `MixFlow single` has generated millions of samples in Figure 4. See Ap-



(a) student t regression (b) sparse regression (high dim)

Figure 3. Sliced log conditional densities on student-t regression (Figure 3a) and high-dimensional sparse regression (Figure 3b). We visualize the log conditional density of the first coordinate by fixing other coordinates to value 0. The `NF` methods are chosen to be the best performing ones from Figures 2e and 2g. Since we only know the log posterior density up to an unknown constant, we shift all lines to have maximal value 0 for visualization.



(a) sparse regression (high dim) (b) student t regression

Figure 4. Timing results (100 trials), showing sampling time (first row) and ESS per second (second row).

pendix E.4 for a detailed discussion of this trade-off.

Figures 4 and 19 further show the computational efficiency in terms of ESS per second on real data examples, which reflects the autocorrelation between drawn samples. Results show that `MixFlow` produce comparable ESS per second to `HMC`. `MixFlow single` behaves similarly to `HMC` as expected since the pseudo-momentum refreshment we proposed (steps (2-3) of Section 5) resembles the momentum resample step of `HMC`. The ESS efficiency of `MixFlow iid` depends on the trade-off between a slower sampling time and i.i.d. nature of drawn samples. In these real data examples, `MixFlow iid` typically produces a high ESS per second; but in the synthetic examples (Figure 14b), `MixFlow iid` is similar to the others.

7. Conclusion

This work presented MixFlows, a new variational family constructed from mixtures of pushforward maps. Experiments demonstrate a comparable sample quality to NUTS and more reliable posterior approximations than standard normalizing flows. A main limitation of our methodology is numerical stability; reversing the flow for long trajectories can be unstable in practice. Future work includes developing more stable momentum refreshment schemes and extensions via involutive MCMC (Neklyudov et al., 2020; Spanbauer et al., 2020; Neklyudov & Welling, 2022).

Acknowledgements

The authors gratefully acknowledge the support of a National Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant and a UBC four year doctoral fellowship.

References

Aldous, D. and Thorisson, H. Shift-coupling. *Stochastic Processes and their Applications*, 44:1–14, 1993.

Anastasiou, A., Barp, A., Briol, F.-X., Ebner, B., Gaunt, R., Ghaderinezhad, F., Gorham, J., Gretton, A., Ley, C., Liu, Q., Mackey, L., Oates, C., Reinert, G., and Swan, Y. Stein's method meets statistics: a review of some recent developments. *arXiv:2105.03481*, 2021.

Birkhoff, G. Proof of the ergodic theorem. *Proceedings of the National Academy of Sciences*, 17(12):656–660, 1931.

Blei, D., Kucukelbir, A., and McAuliffe, J. Variational inference: a review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Brooks, S. and Gelman, A. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455, 1998.

Butzer, P. and Westphal, U. The mean ergodic theorem and saturation. *Indiana University Mathematics Journal*, 20 (12):1163–1174, 1971.

Campbell, T. and Li, X. Universal boosting variational inference. In *Advances in Neural Information Processing Systems*, 2019.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1), 2017.

Caterini, A., Doucet, A., and Sejdinovic, D. Hamiltonian variational auto-encoder. In *Advances in Neural Information Processing Systems*, 2018.

Chen, N., Xu, Z., and Campbell, T. Bayesian inference via sparse Hamiltonian flows. In *Advances in Neural Information Processing Systems*, 2022.

Chérief-Abdellatif, B.-E. and Alquier, P. Consistency of variational Bayes inference for estimation and model selection in mixtures. *Electronic Journal of Statistics*, 12 (2):2995–3035, 2018.

Chwialkowski, K., Strathmann, H., and Gretton, A. A kernel test of goodness of fit. In *International Conference on Machine Learning*, 2016.

Constantinopoulos, C., Titsias, M., and Likas, A. Bayesian feature and model selection for Gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1013–1018, 2006.

Corduneau, A. and Bishop, C. Variational Bayesian model selection for mixture distributions. In *Artificial Intelligence and Statistics*, 2001.

Cowles, M. K. and Carlin, B. Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

Dajani, K. and Dirksin, S. A simple introduction to ergodic theory, 2008. URL: https://webspace.science.uu.nl/ kraai101/lecturenotes2009.pdf.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. In *International Conference on Learning Representations*, 2017.

Eisner, T., Farkas, B., Haase, M., and Nagel, R. *Operator Theoretic Aspects of Ergodic Theory*. Graduate Texts in Mathematics. Springer, 2015.

Fjelde, T. E., Xu, K., Tarek, M., Yalburgi, S., and Ge, H. Bijectors.jl: Flexible transformations for probability distributions. In *Symposium on Advances in Approximate Bayesian Inference*, pp. 1–17, 2020.

Geffner, T. and Domke, J. MCMC variational inference via uncorrected Hamiltonian annealing. In *Advances in Neural Information Processing Systems*, 2021.

Gelman, A. and Rubin, D. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4): 457–472, 1992.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., and Rubin, D. *Bayesian data analysis*. CRC Press, 3rd edition, 2013.

Gershman, S., Hoffman, M., and Blei, D. Nonparametric variational inference. In *International Conference on Machine Learning*, 2012.

Geweke, J. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In Bernardo, J. M., Berger, J. O., and Dawid, A. P. (eds.), *Bayesian Statistics*, volume 4, pp. 169–193, 1992.

Gong, W., Li, Y., and Hernández-Lobato, J. M. Sliced kernelized Stein discrepancy. In *International Conference on Learning Representations*, 2021.

Gorham, J. and Mackey, L. Measuring sample quality with Stein's method. In *Advances in Neural Information Processing Systems*, 2015.

Gorham, J. and Mackey, L. Measuring sample quality with kernels. In *International Conference on Machine Learning*, 2017.

Guo, F., Wang, X., Fan, K., Broderick, T., and Dunson, D. Boosting variational inference. In *Advances in Neural Information Processing Systems*, 2016.

Haario, H., Saksman, E., and Tamminen, J. An adaptive Metropolis algorithm. *Bernoulli*, pp. 223–242, 2001.

Hamidieh, K. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018.

Harrison Jr., D. and Rubinfeld, D. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, 1978.

Hoffman, M. and Gelman, A. The No-U-Turn Sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

Huggins, J. and Mackey, L. Random feature Stein discrepancies. In *Advances in Neural Information Processing Systems*, 2018.

Jaakkola, T. and Jordan, M. Improving the mean field approximation via the use of mixture distributions. In *Learning in graphical models*, pp. 163–173. Springer, 1998.

Jankowiak, M. and Phan, D. Surrogate likelihoods for variational annealed importance sampling. *arXiv:2112.12194*, 2021.

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

Kakutani, S. Iteration of linear operations in complex Banach spaces. *Proceedings of the Imperial Academy*, 14 (8):295–300, 1938.

Kingma, D. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Kobyzev, I., Prince, S., and Brubaker, M. Normalizing flows: an introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, 2021.

Kullback, S. and Leibler, R. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1): 79–86, 1951.

Liu, C. and Rubin, D. ML estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, pp. 19–39, 1995.

Liu, Q., Lee, J., and Jordan, M. A kernelized Stein discrepancy for goodness-of-fit tests and model evaluation. In *International Conference on Machine Learning*, 2016.

Locatello, F., Dresdner, G., Khanna, R., Valera, I., and Rätsch, G. Boosting black box variational inference. In *Advances in Neural Information Processing Systems*, 2018a.

Locatello, F., Khanna, R., Ghosh, J., and Rätsch, G. Boosting variational inference: an optimization perspective. In *International Conference on Artificial Intelligence and Statistics*, 2018b.

Masa-aki, S. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

Miller, A., Foti, N., and Adams, R. Variational boosting: iteratively refining posterior approximations. In *International Conference on Machine Learning*, 2017.

Moro, S., Cortez, P., and Rita, P. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Murray, I. and Elliott, L. Driving Markov chain Monte Carlo with a dependent random stream. *arXiv:1204.3187*, 2012.

Neal, R. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics, No. 118. Springer-Verlag, 1996.

Neal, R. Slice sampling. *The Annals of Statistics*, 31(3): 705–767, 2003.

Neal, R. Hamiltonian importance sampling. Banff International Research Station (BIRS) Workshop on Mathematical Issues in Molecular Dynamics, 2005.

Neal, R. MCMC using Hamiltonian dynamics. In Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (eds.), *Handbook of Markov chain Monte Carlo*, chapter 5. CRC Press, 2011.

Neal, R. How to view an MCMC simulation as permutation, with applications to parallel simulation and improved importance sampling. *arXiv:1205.0070*, 2012.

Neklyudov, K. and Welling, M. Orbital MCMC. In *Artificial Intelligence and Statistics*, 2022.

Neklyudov, K., Welling, M., Egorov, E., and Vetrov, D. Involutive MCMC: a unifying framework. In *International Conference on Machine Learning*, 2020.

Neklyudov, K., Bondesan, R., and Welling, M. Deterministic gibbs sampling via ordinary differential equations. *arXiv:2106.10188*, 2021.

Ormerod, J., You, C., and Müller, S. A variational Bayes approach to variable selection. *Electronic Journal of Statistics*, 11(2):3549–3594, 2017.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22:1–64, 2021.

Qiao, Y. and Minematsu, N. A study on invariance of f-divergence and its application to speech recognition. *IEEE Transactions on Signal Processing*, 58(7):3884–3890, 2010.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *International Conference on Artificial Intelligence and Statistics*, 2014.

Ranganath, R., Tran, D., and Blei, D. Hierarchical variational models. In *International Conference on Machine Learning*, 2016.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015a.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538. PMLR, 2015b.

Riesz, F. Some mean ergodic theorems. *Journal of the London Mathematical Society*, 13(4):274–278, 1938.

Robert, C. and Casella, G. *Monte Carlo Statistical Methods*. Springer, 2nd edition, 2004.

Robert, C. and Casella, G. A short history of Markov Chain Monte Carlo: subjective recollections from incomplete data. *Statistical Science*, 26(1):102–115, 2011.

Roberts, G. and Rosenthal, J. Shift-coupling and convergence rates of ergodic averages. *Stochastic Models*, 13 (1):147–165, 1997.

Roberts, G. and Rosenthal, J. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1: 20–71, 2004.

Rotskoff, G. and Vanden-Eijnden, E. Dynamical computation of the density of states and Bayes factors using nonequilibrium importance sampling. *Physical Review Letters*, 122(15):150602, 2019.

Salimans, T. and Knowles, D. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

Salimans, T., Kingma, D., and Welling, M. Markov chain Monte Carlo and variational inference: bridging the gap. In *International Conference on Machine Learning*, 2015.

Spanbauer, S., Freer, C., and Mansinghka, V. Deep involutive generative models for neural MCMC. *arXiv:2006.15167*, 2020.

Tabak, E. and Turner, C. A family of non-parametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

Tao, C., Chen, L., Zhang, R., Henao, R., and Carin, L. Variational inference and model selection with generalized evidence bounds. In *International Conference on Machine Learning*, 2018.

Thin, A., Janati, Y., Le Corff, S., Ollion, C., Doucet, A., Durmus, A., Moulines, É., and Robert, C. NEO: non equilibrium sampling on the orbit of a deterministic transform. *Advances in Neural Information Processing Systems*, 2021a.

Thin, A., Kotelevskii, N., Durmus, A., Panov, M., Moulines, E., and Doucet, A. Monte Carlo variational autoencoders. In *International Conference on Machine Learning*, 2021b.

Tupper, P. Ergodicity and the numerical simulation of Hamiltonian systems. *SIAM Journal on Applied Dynamical Systems*, 4(3):563–587, 2005.

U.S. Department of Justice Federal Bureau of Investigation. Crime in the United States, 1995. URL: https://ucr.fbi.gov/crime-in-the-u.s/1995.

ver Steeg, G. and Galstyan, A. Hamiltonian dynamics with non-Newtonian momentum for rapid sampling. In *Advances in Neural Information Processing Systems*, 2021.

Wainwright, M. and Jordan, M. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

Wang, X. Boosting variational inference: theory and examples. Master's thesis, Duke University, 2016.

Wenliang, L. and Kanagawa, H. Blindness of score-based methods to isolated components and mixing proportions. *arXiv:2008.10087*, 2020.

Wolf, C., Karl, M., and van der Smagt, P. Variational inference with Hamiltonian Monte Carlo. *arXiv:1609.08203*, 2016.

Xu, K., Ge, H., Tebbutt, W., Tarek, M., Trapp, M., and Ghahramani, Z. AdvancedHMC.jl: A robust, modular and efficient implementation of advanced HMC algorithms. In *Symposium on Advances in Approximate Bayesian Inference*, 2020.

Xu, Z. and Campbell, T. The computational asymptotics of variational inference and the Laplace approximation. *Statistics and Computing*, 32(4):1–37, 2022.

Yosida, K. Mean ergodic theorem in Banach spaces. *Proceedings of the Imperial Academy*, 14(8):292–294, 1938.

Zhang, G., Hsu, K., Li, J., Finn, C., and Grosse, R. Differentiable annealed importance sampling and the perils of gradient noise. In *Advances in Neural Information Processing Systems*, 2021.

Zhang, Y. and Hernández-Lobato, J. M. Ergodic inference: accelerate convergence by optimisation. In *arXiv:1805.10377*, 2020.

Zobay, O. Variational Bayesian inference with Gaussian-mixture approximations. *Electronic Journal of Statistics*, 8:355–389, 2014.

A. Proofs

Proof of Proposition 3.1. Because both estimates are unbiased, it suffices to show that $\mathbb{E}[f^2(X)] \geq \mathbb{E}\left[\left(\frac{1}{N}\sum_{n=0}^{N-1} f(T^n X_0)\right)^2\right]$, which itself follows by Jensen's inequality:

$$\mathbb{E}\left[\left(\frac{1}{N}\sum_{n=0}^{N-1} f(T^n X_0)\right)^2\right] \leq \mathbb{E}\left[\frac{1}{N}\sum_{n=0}^{N-1} f^2(T^n X_0)\right] = \mathbb{E}[f^2(X)].$$

□

Proof of Theorem 4.1. Since setwise convergence implies weak convergence, we will focus on proving setwise convergence. We have that $q_{\lambda,N}$ converges setwise to π if and only if for all measurable bounded $f : \mathcal{X} \to \mathbb{R}$,

$$\mathbb{E}f(X_N) \to \mathbb{E}f(X), \qquad X_N \sim q_{\lambda,N}, \quad X \sim \pi.$$

The proof proceeds by directly analyzing $\mathbb{E}f(X_N)$:

$$\begin{aligned}
\mathbb{E}f(X_N) &= \int f(x) q_{\lambda,N}(\mathrm{d}x) \\
&= \frac{1}{N}\sum_{n=0}^{N-1} \int f(x)(T_\lambda^n q_0)(\mathrm{d}x) \\
&= \frac{1}{N}\sum_{n=0}^{N-1} \int f(T_\lambda^n x) q_0(\mathrm{d}x) \\
&= \int \frac{1}{N}\sum_{n=0}^{N-1} f(T_\lambda^n x) q_0(\mathrm{d}x).
\end{aligned}$$

Since $q_0 \ll \pi$, by the Radon-Nikodym theorem, there exists a density of q_0 with respect to π, so

$$\mathbb{E}f(X_N) = \int \frac{1}{N}\sum_{n=0}^{N-1} f(T_\lambda^n x) \frac{\mathrm{d}q_0}{\mathrm{d}\pi}(x)\pi(\mathrm{d}x).$$

By the pointwise ergodic theorem (Theorem 2.3), $f_N(x) = \frac{1}{N}\sum_{n=0}^{N-1} f(T_\lambda^n x)$ converges pointwise π-a.e. to $\int f \mathrm{d}\pi$; and because f is bounded, f_N is uniformly bounded for all $N \in \mathbb{N}$. Hence by the Lebesgue dominated convergence theorem,

$$\begin{aligned}
\lim_{N\to\infty} \mathbb{E}f(X_N) &= \lim_{N\to\infty} \int \frac{1}{N}\sum_{n=0}^{N-1} f(T_\lambda^n x) \frac{\mathrm{d}q_0}{\mathrm{d}\pi}(x)\pi(\mathrm{d}x) \\
&= \int \left(\int f \mathrm{d}\pi\right) \frac{\mathrm{d}q_0}{\mathrm{d}\pi}(x)\pi(\mathrm{d}x) \\
&= \left(\int f \mathrm{d}\pi\right) \cdot 1 = \mathbb{E}f(X).
\end{aligned}$$

□

Theorem A.1 (Mean ergodic theorem in Banach spaces [Yosida, 1938; Kakutani, 1938; Riesz, 1938; Eisner et al., 2015, Theorem 8.5])**.** *Let T be a bounded linear operator on a Banach space E, define the operator*

$$A_N = \frac{1}{N}\sum_{n=0}^{N-1} T^n,$$

and let

$$\mathrm{fix}(T) = \{v \in E : Tv = v\} = \ker(I - T).$$

Suppose that $\sup_{N\in\mathbb{N}} \|A_N\| < \infty$ *and that* $\frac{1}{N}T^N v \to 0$ *for all* $v \in E$*. Then the subspace*

$$V = \left\{ v \in E : \lim_{N\to\infty} A_N v \text{ exists} \right\}$$

is closed, T*-invariant, and decomposes into a direct sum of closed subspaces*

$$V = \mathrm{fix}(T) \oplus \overline{\mathrm{range}}(I - T).$$

The operator $T|_V$ *on* V *is mean ergodic. Furthermore, the operator*

$$A : V \to \mathrm{fix}(T) \qquad Av = \lim_{N\to\infty} A_N v$$

is a bounded projection with kernel $\ker(A) = \overline{\mathrm{range}}(I - T)$ *and* $AT = A = TA$*.*

Proof of Theorem 4.2. We suppress λ subscripts for brevity. Consider the Banach space $\mathcal{M}(\pi)$ of signed finite measures dominated by π endowed with the total variation norm $\|m\| = \sup_{A\subseteq\mathcal{X}} m(A) - \inf_{A\subseteq\mathcal{X}} m(A)$. Then the pushforward Tm of $m \in \mathcal{M}(\pi)$ under T is dominated by π since

$$\pi(A) = 0 \implies \pi(T^{-1}(A)) = 0 \implies m(T^{-1}(A)) = 0.$$

Note the slight abuse of notation involving the same symbol for $T : \mathcal{X} \to \mathcal{X}$ and the associated operator. Hence $T : \mathcal{M}(\pi) \to \mathcal{M}(\pi)$ is a linear operator on $\mathcal{M}(\pi)$. Further,

$$\|Tm\| = \sup_{A\subseteq\mathcal{X}} m(T^{-1}(A)) - \inf_{A\subseteq\mathcal{X}} m(T^{-1}(A)) \leq \sup_{A\subseteq\mathcal{X}} m(A) - \inf_{A\subseteq\mathcal{X}} m(A) = \|m\|,$$

and so T is bounded with $\|T\| \leq 1$. Hence $\frac{1}{N}T^N m \to 0$ for all $m \in \mathcal{M}(\pi)$, and if we define the operator

$$\forall N \in \mathbb{N}, \quad A_N = \frac{1}{N}\sum_{n=0}^{N-1} T^n,$$

we have that $\sup_{N\in\mathbb{N}} \|A_N\| \leq 1$. Therefore by the mean ergodic theorem in Banach spaces [Yosida, 1938; Kakutani, 1938; Riesz, 1938; Eisner et al., 2015, Theorem 8.5], we have that

$$A : V \to \mathrm{fix}(T) \quad Am = \lim_{N\to\infty} A_N m$$

where $V = \{m \in \mathcal{M}(\pi) : \lim_{N\to\infty} A_N m \text{ exists}\}$. Eisner et al. (2015, Theorem 8.20) guarantees that $V = \mathcal{M}(\pi)$ as long as the weak limit of $A_N m$ exists for each $m \in \mathcal{M}(\pi)$. Note that since $\mathcal{M}(\pi)$ and $L^1(\pi)$ are isometric (via the map $m \mapsto \frac{\mathrm{d}m}{\mathrm{d}\pi}$), and π is σ-finite, the dual of $\mathcal{M}(\pi)$ is the set of linear functionals

$$m \mapsto \int \phi(x) m(\mathrm{d}x), \qquad \phi \in L^\infty(\pi).$$

So therefore we have that $V = \mathcal{M}(\pi)$ if for each $m \in \mathcal{M}(\pi)$, there exists a $g \in \mathcal{M}(\pi)$ such that

$$\forall \phi \in L^\infty(\pi), \qquad \int \phi(x)(A_N m)(\mathrm{d}x) \to \int \phi(x) g(\mathrm{d}x).$$

The same technique using a transformation of variables and the pointwise ergodic theorem from the proof of Theorem 4.1 provides the desired weak convergence. Therefore we have that $A_N m$ converges in total variation to $\mathrm{fix}(T)$ for all $m \in \mathcal{M}(\pi)$, and hence $A_N q_0$ converges in total variation to $\mathrm{fix}(T)$. Furthermore Theorem 4.1 guarantees weak convergence of $A_N q_0$ to π, and $\pi \in \mathrm{fix}(T)$, so thus $\mathrm{D}_{\mathrm{TV}}(q_{\lambda,N}, \pi) \to 0$. □

Proof of Theorem 4.3. We suppress λ subscripts for brevity. By the triangle inequality,

$$\mathrm{D}_{\mathrm{TV}}\left(\hat{q}_N, \pi\right) \leq \mathrm{D}_{\mathrm{TV}}\left(\hat{q}_N, q_N\right) + \mathrm{D}_{\mathrm{TV}}\left(q_N, \pi\right).$$

Focusing on the error term, we have that

$$\begin{aligned}
\mathrm{D}_{\mathrm{TV}}\left(\hat{q}_N, q_N\right) &= \mathrm{D}_{\mathrm{TV}}\left(\frac{1}{N}\sum_{n=1}^{N-1}\hat{T}^n q_0, \frac{1}{N}\sum_{n=1}^{N-1}T^n q_0\right) \\
&= \frac{N-1}{N}\mathrm{D}_{\mathrm{TV}}\left(\frac{1}{N-1}\sum_{n=1}^{N-1}\hat{T}^n q_0, \frac{1}{N-1}\sum_{n=1}^{N-1}T^n q_0\right) \\
&= \frac{N-1}{N}\mathrm{D}_{\mathrm{TV}}\left(\hat{T}\frac{1}{N-1}\sum_{n=1}^{N-1}\hat{T}^{n-1} q_0, T\frac{1}{N-1}\sum_{n=1}^{N-1}T^{n-1} q_0\right) \\
&= \frac{N-1}{N}\mathrm{D}_{\mathrm{TV}}\left(\hat{T}\frac{1}{N-1}\sum_{n=0}^{N-2}\hat{T}^{n} q_0, T\frac{1}{N-1}\sum_{n=0}^{N-2}T^{n} q_0\right) \\
&= \frac{N-1}{N}\mathrm{D}_{\mathrm{TV}}\left(\hat{T}\hat{q}_{N-1}, Tq_{N-1}\right) \\
&= \frac{N-1}{N}\mathrm{D}_{\mathrm{TV}}\left(\hat{q}_{N-1}, \hat{T}^{-1}Tq_{N-1}\right),
\end{aligned}$$

where the last equality is due to the fact that $\hat{T}$ is a bijection. The triangle inequality yields

$$\begin{aligned}
\mathrm{D}_{\mathrm{TV}}\left(\hat{q}_N, q_N\right) &\leq \frac{N-1}{N}\left(\mathrm{D}_{\mathrm{TV}}\left(\hat{q}_{N-1}, q_{N-1}\right) + \mathrm{D}_{\mathrm{TV}}\left(q_{N-1}, \hat{T}^{-1}Tq_{N-1}\right)\right) \\
&= \frac{N-1}{N}\left(\mathrm{D}_{\mathrm{TV}}\left(\hat{q}_{N-1}, q_{N-1}\right) + \mathrm{D}_{\mathrm{TV}}\left(\hat{T}q_{N-1}, Tq_{N-1}\right)\right).
\end{aligned}$$

Then iterating that technique yields

$$\mathrm{D}_{\mathrm{TV}}\left(\hat{q}_N, q_N\right) \leq \sum_{n=1}^{N-1}\frac{n}{N}\mathrm{D}_{\mathrm{TV}}\left(\hat{T}q_n, Tq_n\right), \tag{6}$$

which completes the proof. □

Proof of Corollary 4.4. Examining the total variation in its L^1 distance formulation yields that for $n = 1, \ldots, N-1$,

$$\begin{aligned}
\mathrm{D}_{\mathrm{TV}}\left(\hat{T}q_n, Tq_n\right) &= \int\left|\frac{\hat{T}q_n(x)}{Tq_n(x)} - 1\right| Tq_n(\mathrm{d}x) \\
&= \int\left|\exp\left\{\log q_n(\hat{T}^{-1}x) - \log q_n(T^{-1}x) + \log J(T^{-1}x) - \log \hat{J}(\hat{T}^{-1}x)\right\} - 1\right| Tq_n(\mathrm{d}x).
\end{aligned}$$

By assumption,

$$\left|\log q_n(\hat{T}^{-1}x) - \log q_n(T^{-1}x)\right| \leq \ell\left\|\hat{T}^{-1}x - T^{-1}x\right\| \leq \ell\epsilon,$$

and

$$\begin{aligned}
\left|\log J(T^{-1}x) - \log \hat{J}(\hat{T}^{-1}x)\right| &\leq \left|\log J(T^{-1}x) - \log J(\hat{T}^{-1}x)\right| + \left|\log J(\hat{T}^{-1}x) - \log \hat{J}(\hat{T}^{-1}x)\right| \\
&\leq \ell\left\|\hat{T}^{-1}x - T^{-1}x\right\| + \epsilon \\
&\leq (\ell+1)\epsilon.
\end{aligned}$$

Combining these two bounds yields

$$\mathrm{D}_{\mathrm{TV}}\left(\hat{T}q_n, Tq_n\right) \leq \exp((2\ell+1)\epsilon) - 1 \leq (2\ell+1)\epsilon\exp((2\ell+1)\epsilon).$$

Therefore, by Equation (6), we obtain that

$$\mathrm{D}_{\mathrm{TV}}\left(\hat{q}_N, q_N\right) \leq \sum_{n=1}^{N-1}\frac{n}{N}\mathrm{D}_{\mathrm{TV}}\left(\hat{T}q_n, Tq_n\right) \leq \frac{N-1}{2}e^{(2\ell+1)\epsilon}\cdot(2\ell+1)\epsilon \leq N\epsilon(\ell+1)e^{(2\ell+1)\epsilon}.$$

Finally, combining the above with Theorem 4.3 yields the desired result. □

Algorithm 4 `DensityTriple`(x): Evaluate $T_\lambda^{-N+1}(x)$, $q_{\lambda,N}(x)$, and $\prod_{j=1}^{N-1} J_\lambda(T_\lambda^{-j}x)$ with $O(N)$-time, $O(1)$-memory

Input: location x, reference distribution q_0, flow map T_λ, Jacobian J_λ, number of steps N
$L \leftarrow 0$
$w \leftarrow q_0(x)$
for $n = 1, \ldots, N-1$ **do**
  $x \leftarrow T_\lambda^{-1}(x)$
  $L \leftarrow L + \log J_\lambda(x)$
  $w \leftarrow$ `LogSumExp`$(w, \log q_0(x) - L)$
end for
$w \leftarrow w - \log N$
Return: x, $\exp(w)$, $\exp(L)$

Algorithm 5 `EstELBO`(λ, N): Estimate the ELBO for $q_{\lambda,N}$ in $O(N)$-time, $O(1)$-memory.

Input: reference q_0, unnormalized target p, flow map T_λ, Jacobian J_λ, number of flow steps N
$x \leftarrow$ `Sample`(q_0)
$x', z, J \leftarrow$ `DensityTriple`(x)
$f \leftarrow \log p(x)$
$g \leftarrow \log z$
for $n = 1, \ldots, N-1$ **do**
  $\bar{q} \leftarrow \frac{1}{N} q_0(x')/J$
  $J_{n-1} \leftarrow J_\lambda(x)$
  $z \leftarrow (z - \bar{q})\,/J_{n-1}$
  $x \leftarrow T_\lambda(x)$
  $z \leftarrow z + \frac{1}{N} q_0(x)$
  $f \leftarrow f + \log p(x)$
  $g \leftarrow g + \log z$
  if $n < N-1$ **then**
    $J \leftarrow J \cdot J_{n-1}/J_\lambda(x')$
  end if
  $x' \leftarrow T_\lambda(x')$
end for
Return: $\widehat{\mathrm{ELBO}}(\lambda, N) \leftarrow \frac{1}{N}(f - g)$
$\widehat{\mathrm{ELBO}}(\lambda, N)$

B. Memory efficient ELBO estimation

C. Hamiltonian flow pseudocode

D. Extensions

Tunable reference So far we have assumed that the reference distribution q_0 for the flow is fixed. Given that q_0 is often quite far from the target π, this forces the variational flow to spend some of its steps just moving the bulk of the mass to π. But this can be accomplished much easier with, say, a simple linear transformation that efficiently allows large global moves in mass. For example, if $q_0 = \mathcal{N}(0, I)$, we can include a map M_θ

$$q_{\lambda,N} = \frac{1}{N} \sum_{n=0}^{N-1} T_\lambda^n M_\theta q_0,$$

where $M_\theta(x) = \theta_1 x + \theta_2$, where $\theta_1 \in \mathbb{R}^{d\times d}$ and $\theta_2 \in \mathbb{R}^d$. Note that it is possible to optimize the reference and flow jointly, or to optimize the reference distribution by itself first and then use that fixed reference in the flow optimization.

Algorithm 6 Compute $T_{\lambda,\epsilon}$ and its Jacobian $J_{\lambda,\epsilon}$ for Hamiltonian flow with leapfrog integrator

Input: initial state $x \in \mathcal{X}$, $\rho \in \mathbb{R}^d$, $u \in [0,1]$, step size ϵ, shift $\xi \in \mathbb{R}$, pseudorandom shift $z(\cdot,\cdot)$
$x_0, \rho_0 \leftarrow x, \rho$
for $\ell = 1, \ldots, L$ **do**
  $x_\ell, \rho_\ell \leftarrow \hat{H}_\epsilon(x_{\ell-1}, \rho_{\ell-1})$
end for
$x', \rho' \leftarrow x_L, \rho_L$
$u' \leftarrow u + \xi \mod 1$
for $i = 1, \ldots, d$ **do**
  $\rho''_i \leftarrow R^{-1}(R(\rho'_i) + z(x'_i, u') \mod 1)$
end for
$J \leftarrow m(\rho')/m(\rho'')$
Return: $(x', \rho'', u'), J$

Automated burn-in A common practice in MCMC is to throw away a first fraction of the states in the sequence to ensure that the starting sample is in a high probability region of the target distribution, thus reducing the bias from initialization ("burn-in"). Usually one needs a diagnostic to check when burn-in is completed. In the case of `MixFlow`, we can monitor the burn-in phase in a principled way by evaluating the ELBO. Once the flow is trained, the variational distribution with M burn-in samples is simply

$$q_{\lambda,M,N} = \frac{1}{N-M} \sum_{n=M}^{N-1} T_\lambda^n q_0,$$

We can easily optimize this by estimating the ELBOs for $M = 1, \ldots, N$.

Mixtures of MixFlows One can build multiple MixFlows starting from multiple different initial reference distributions; when the posterior is multimodal, it may be the case that some of these MixFlows converge to different modes but do not mix across modes. In this scenario, it can be helpful to average several MixFlows, i.e., build an approximation of the form

$$q^\star_{\lambda,N} = \sum_{k=1}^{K} w_k (q_{\lambda,N})_k, \quad \sum_{k=1}^{K} w_k = 1.$$

Because each component flow provides access to i.i.d. samples and density evaluation, `MixFlow` provides the ability to optimize the weights by maximizing the ELBO (i.e., minimizing the KL divergence).

E. Additional experimental details

In this section, we provide details for each experiment presented in the main text, as well as additional results regarding numerical stability and a high-dimensional synthetic experiment. Aside from the univariate synthetic example, all examples include a pseudotime shift step with $\xi = \pi/16$, and a momentum refreshment with $z(x,u) = 0.5\sin(2x+u) + 0.5$. For the kernel Stein discrepancy, we use a IMQ kernel $k(x,y) = (c^2 + \|x-y\|_2^2)^\beta$ with $\beta = -0.5, c = 1$, the same setting as in (Gorham & Mackey, 2017).

For all experiments, unless otherwise stated, `NUTS` uses 20,000 steps for adaptation, targeting at an average acceptance ratio 0.7, and generates 5,000 samples for KSD estimation. The KSD for `MixFlow` is estimated using 2,000 samples. The KSD estimation for `NEO` is based on 5,000 samples generated by a *tuned* `NEO` after 20,000 burn-in steps. We adopted two tuning strategies for `NEO`: (1) choosing among the combinations of several fixed settings of discretization step ($\epsilon = 0.2, 0.5, 1.0$), friction parameter γ of nonequilibrium Hamiltonian dynamics ($\gamma = 0.2, 0.5, 1.0$), integration steps ($K = 10, 20$), and mass matrix of momentum distribution ($M = I$); (2) fixing the integration steps ($K = 10, 20$) and friction parameter ($\gamma = 0.2, 0.5, 1.0$), and using windowed adaptation (Carpenter et al., 2017) to adapt the mass matrix and integration step size, targeting at an average acceptance ratio at 0.7. The optimal setting of `NEO` is considered to be the one that produces lowest average marginal KSD over 3 runs with no NaN values encountered. Performance of `NEO` across various settings for each example is summarized in Table 4. In each of `NEO` MCMC transition, we run 10 deterministic orbits and computes

corresponding normalizing constant estimates *in parallel*. Aside from NEO, all other methods are deployed using a single processor.

As for NUTS, we use the Julia package AdvancedHMC.jl (Xu et al., 2020) with all default settings. NEO adaptation is also conducted using AdvancedHMC.jl with the number of simulation steps set to the number of integration steps of NEO. The ESS is computed using the R package mcmcse, which is based on batch mean estimation. We implement all NFs using the Julia package Bijectors.jl (Fjelde et al., 2020). The flow layers of PlanarFlow and RadialFlow, as well as the coupling layers of RealNVP are implemented in Bijectors.jl. The affine coupling functions of RealNVP—a scaling function and a shifting function—are both parameterized using fully connected neural networks of three layers, of which the activation function is LeakyReLU and number of hidden units is by default the same dimension as the target distribution, unless otherwise stated.

E.1. Univariate synthetic examples

The three target distributions tested in this experiment were

- normal: $\mathcal{N}(2, 2^2)$,
- synthetic Gaussian mixture: $0.5\mathcal{N}(-3, 1.5^2) + 0.3\mathcal{N}(0, 0.8^2) + 0.2\mathcal{N}(3, 0.8^2)$, and
- Cauchy: $\mathsf{Cauchy}(0, 1)$.

For all three examples, we use a momentum refreshment without introducing the pseudotime variable; this enables us to plot the full joint density of $(x, \rho) \in \mathbb{R}^2$:

$$\rho'' \leftarrow R_{\text{Lap}}^{-1}(R_{\text{Lap}}(\rho') + (\sin(2x') + 1)/2 \mod 1).$$

For the three examples, we used the leapfrog stepsize $\epsilon = 0.05$ and run $L = 50$ leapfrogs between each refreshment. For both the Gaussian and Gaussian mixture targets, we use 100 refreshments. In the case of the Cauchy, we used 1,000 refreshments.

E.2. Multivariate synthetic examples

The three target distributions tested in this experiment were

- the banana distribution (Haario et al., 2001):

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}\right), \quad x = \begin{bmatrix} y_1 \\ y_2 + by_1^2 - 100b \end{bmatrix}, \quad b = 0.1;$$

- Neals' funnel (Neal, 2003):

$$x_1 \sim \mathcal{N}\left(0, \sigma^2\right), \quad x_2 \mid x_1 \sim \mathcal{N}\left(0, \exp\left(\frac{x_1}{2}\right)\right), \quad \sigma^2 = 36;$$

- a cross-shaped distribution: in particular, a Gaussian mixture of the form

$$\begin{aligned} x \sim \frac{1}{4}\mathcal{N}\left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 0.15^2 & 0 \\ 0 & 1 \end{bmatrix}\right) + \frac{1}{4}\mathcal{N}\left(\begin{bmatrix} -2 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0.15^2 \end{bmatrix}\right) \\ + \frac{1}{4}\mathcal{N}\left(\begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0.15^2 \end{bmatrix}\right) + \frac{1}{4}\mathcal{N}\left(\begin{bmatrix} 0 \\ -2 \end{bmatrix}, \begin{bmatrix} 0.15^2 & 0 \\ 0 & 1 \end{bmatrix}\right); \end{aligned}$$

- and a warped Gaussian distribution

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & 0 \\ 0 & 0.12^2 \end{bmatrix}\right), \quad x = \begin{bmatrix} \|y\|_2 \cos\left(\operatorname{atan2}\left(y_2, y_1\right) - \frac{1}{2}\|y\|_2\right) \\ \|y\|_2 \sin\left(\operatorname{atan2}\left(y_2, y_1\right) - \frac{1}{2}\|y\|_2\right) \end{bmatrix},$$

  where $\operatorname{atan2}(y, x)$ is the angle, in radians, between the positive x axis and the ray to the point (x, y).

We used flows with 500 and 2000 refreshments for the banana distribution, Neal's Funnel respectively, and a flow with 1000 refreshments for both cross distribution and warped Gaussian. Between each refreshment we used 200 leapfrog steps for the banana distribution, 60 for the cross distribution, and 80 for the funnel and warped Gaussian. Note that in all four examples, we individually tuned the step size ϵ by maximizing the estimated ELBO, as shown in Figure 5a. Figure 5b also demonstrates how the ELBO varies versus the number of refreshments N. For small step sizes ϵ, the discretized Hamiltonian dynamics approximates the continuous dynamics, and the ELBO generally increases with N indicating convergence to the target. For larger step sizes ϵ, the ELBO increases to a peak and then decreases, indicating that the discretized dynamics do not exactly target the desired distribution.

Figure 6 presents a comparison of the uncertainty involved in estimating the expectations of a test function f for both `MixFlow iid` and `MixFlow single`. Specifically, it examines the streaming estimation of $\mathbb{E}[f(X)], X \sim q_N$, where $f(x) = \|x\|_1$, based on samples generated from `MixFlow iid` and `MixFlow single` under 50,000 flow map evaluations over 10 independent trials. Note that we assess computational cost via the number of flow map evaluations, not by the number of draws, because the cost per draw is random in `MixFlow iid` (due to $K \sim \mathsf{Unif}\{0, \dots, N-1\}$), while the cost per draw is fixed to N evaluations for the trajectory estimate in `MixFlow single`. The results indicate that, given equivalent computational resources, the trajectory-averaged estimates generally exhibit lower variances between trials than the naïve i.i.d. Monte Carlo estimate. This observation validates Proposition 3.1.

Figure 7 shows why we opted to use a Laplace momentum distribution as opposed to a Gaussian momentum. In particular, the numerical error of composing $T_\lambda^K \circ T_\lambda^{-K}$ and $T_\lambda^{-K} \circ T_\lambda^K$ (denoted as "Bwd" and "Fwd" in the legend, respectively) indicate that the flow using the Laplace momentum is more reliably invertible for larger numbers of refreshments than the flow with a Gaussian momentum. Figure 8 uses a high-precision (256 bit) floating point representation to further illustrate the rapid escalation of numerical error when evaluating forward and backward trajectories on Gaussian Hamiltonian `MixFlows`. For all four synthetic examples, after approximately 100 flow transformations, `MixFlows` with Gaussian momentum exhibits an error on the scale of the target distribution itself (per the contour plots (d)-(g) in Figure 1). This may be due to the fact that the normal has very light tails; for large momentum values in the right tail, the CDF is ≈ 1, which gets rounded to exactly 1 in floating point representation. Our implementation of the CDF and inverse CDF was fairly naïve, so it is possible that stability could be improved with more care taken to prevent rounding error. We leave a more careful examination of this error to future work; for this work, using a Laplace distribution momentum sufficed to maintain a low numerical error.

Finally, Figure 9 provides a more comprehensive set of sample histograms for these experiments, showing the x-, ρ-, and u-marginals. It is clear that `MixFlow` generates samples for each variable reliably from the target marginal.

E.3. Higher-dimensional synthetic experiment

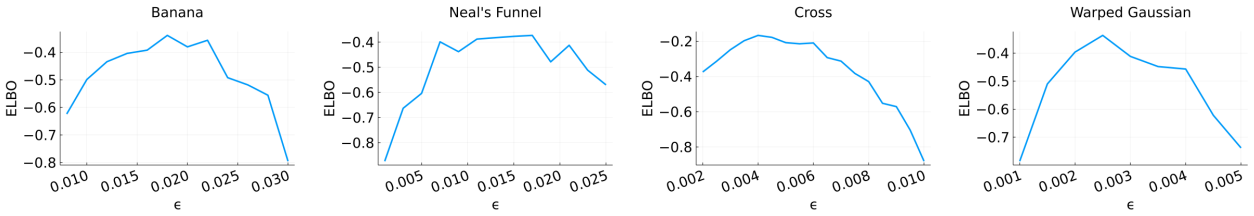
We also tested two higher dimensional Neal's funnel target distributions of $x \in \mathbb{R}^d$ where $d = 5, 20$. In particular, we used the target distribution

$$x_1 \sim \mathcal{N}\left(0, \sigma^2\right), \quad \text{and} \quad \forall i \in \{2, \cdots, d\},\ x_i \mid x_1 \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(0, \exp\left(\frac{x_1}{2}\right)\right),$$

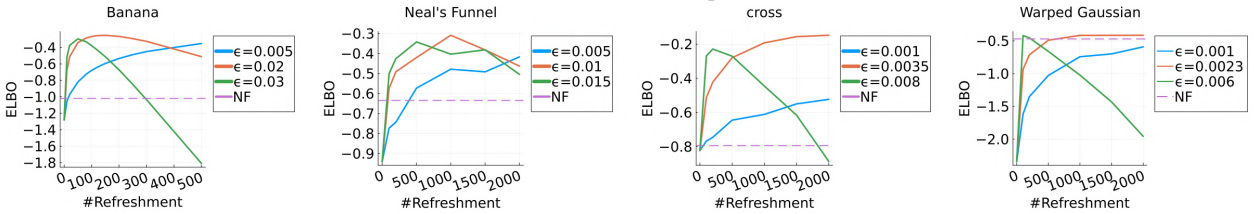
with $\sigma^2 = 36$. We use 80 leapfrogs between refreshments and $\epsilon = 0.0009$ when $d = 5$, and 100 leapfrogs between refreshments and $\epsilon = 0.001$ when $d = 20$ (Figures 10a and 10c shows the ELBO comparison used to tune the step size ϵ). Figures 10b and 10d confirm via the KSD that the method performs as well as `NUTS` in higher-dimensional cases.

E.4. Additional experiments for synthetic examples

In this section, we provide additional comparisons of `MixFlow` against `NUTS`, `HMC`, `NEO` and a generic normalizing flow method—planar flow (`NF`) (Rezende & Mohamed, 2015b) on all four synthetic examples in Appendix E.2. For this set of experiments, all of the settings for `MixFlow` are the same as outlined in Appendix E.2. `HMC` uses the same leapfrog step size and number of leapfrogs steps (between refreshments) as `MixFlow`. For `NF`, 5 Planar layers (Rezende & Mohamed, 2015b) that contain 45 parameters to be optimized (9 parameters for each Planar layer of a 4-dimensional Planar flow) are used unless otherwise stated. We train `NF` using ADAM until convergence (100,000 iterations except where otherwise noted) with the initial step size set to 0.001. And at each step, 10 samples are used for gradient estimation. The initial distribution for `MixFlow`, `NF` and `NUTS` is set to be the mean-field Gaussian approximation, and `HMC` and `NUTS` are initialized using the learned mean of the mean-field Gaussian approximation. All parameters in `NF` are initialized using random samples from the standard Gaussian distribution.



(a): ELBO versus step size ϵ.



(b): ELBO of `MixFlow` versus number of refreshments N over different step sizes, and comparision to tuned `PlanarFlow`

Figure 5. `MixFlow` tuning for the four multivariate synthetic examples

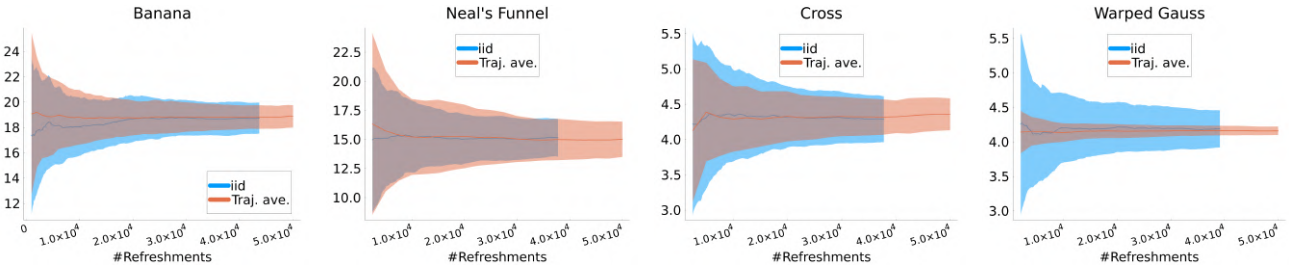


Figure 6. Comparison of Monte Carlo estimates of $\mathbb{E}[f(X)]$, $X \sim q_N$, $f(x) = \|x\|_1$ based on individual i.i.d. draws (blue) and trajectory-averaged estimates in Equation (3) (orange) on four synthetic examples. The vertical axis indicates the estimate of $\mathbb{E}[f]$, and the horizontal axis indicates the total number of flow transformations evaluated, i.e., total computational cost. Note that in each example, N is fixed; the number of refreshments on the horizontal axis increases because we average over increasingly many draws $X \overset{\text{i.i.d.}}{\sim} q_N$ (blue) or increasingly many trajectory averages (orange). We run 10 trials to assess the quality of each estimate: lines indicate the mean, and error regions indicate standard deviation.

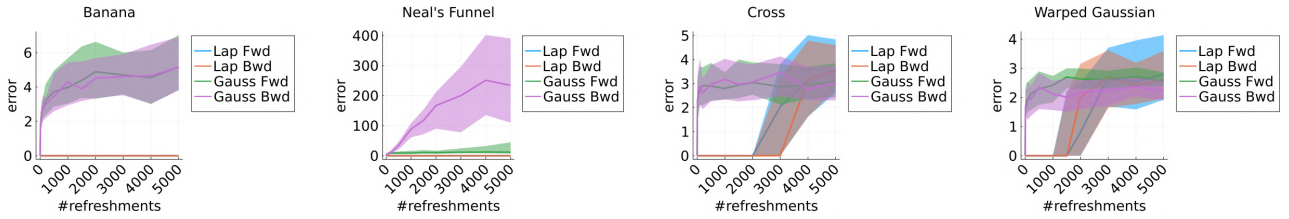
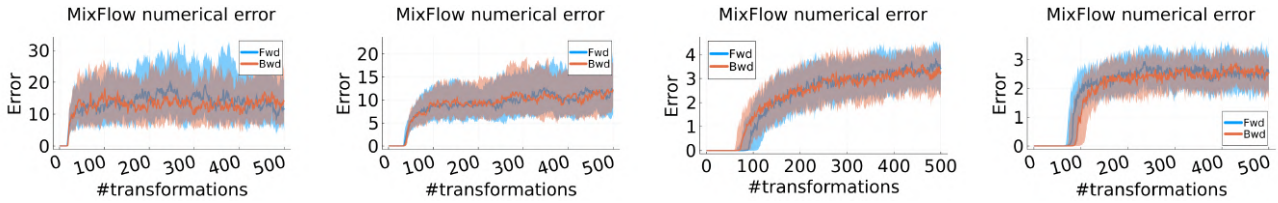


Figure 7. Stability of composing $T_\lambda^{-K} \circ T_\lambda^K$ (`Fwd`) and $T_\lambda^K \circ T_\lambda^{-K}$ (`Bwd`) for the four multivariate experiments with flows constructed using Gaussian (`Gauss`) and Laplace (`Lap`) momentum distributions. The vertical axis shows the 2-norm error of reconstructing (x, ρ, u) sampled from q_0; the horizontal axis shows increasing numbers of refreshments K. The lines indicate the median, and error regions indicate 25th to 75th percentile for 100 independent samples.

Figure 11 compares the sliced log density of tuned `MixFlow` and the importance sampling proposal of tuned `NEO`, visualized in a similar fashion as Figure 1 in Section 6.1. It is clear that `NEO` does not provide a high-quality density approximation, while the log density of `MixFlow` visually matches the target. Indeed, unlike our method, due to the use of a nonequilibrium dynamic, `NEO` does not provide a similar convergence guarantee of its proposal distribution as simulation length increases.

Figure 12 presents comparisons of the marginal sample qualities of `MixFlow` to `NUTS` and `NEO`—both are exact MCMC methods. Results show that `MixFlow` produces comparable marginal sample quality, if not slightly better, than both MCMC



(a): Banana (b): Neal's Funnel (c): Cross (d): Warped Gaussian

Figure 8. Large numerical errors exhibited by Hamiltonian MixFlow with Gaussian momentum on synthetic examples. Figure shows forward (`fwd`) error $\|T^k x - \hat{T}^k x\|$ and backward (`bwd`) error $\|T^{-k} x - \hat{B}^k x\|$ comparing k transformations of the forward approximate/exact maps $\hat{T} \approx T$ and backward approximate/exact maps $\hat{B} \approx T^{-1}$. For the `exact` maps we use a 256-bit `BigFloat` representation, and for the `numerical` approximations we use 64-bit `Float` representation. The lines indicate the median, and error regions indicate 25th to 75th percentile over 100 initialization draws from the reference distribution q_0.

methods. Note that it involves a nontrivial tuning process to achieve the displayed performance of `NEO`. Concrete tuning strategies are explained in the beginning of Appendix E. In fact, finding a good combination of discretization step, friction parameter, simulation length, and mass matrix of momentum distribution is necessary for `NEO` to behave reasonably; Table 4 summarizes the performance of `NEO` under various settings for both synthetic and real data experiments. The standard deviation of marginal sample KSD can change drastically across different settings, meaning that the marginal sample quality can be sensitive to the choice of hyperparameters. Moreover, due to the usage of an unstable Hamiltonian dynamics (with friction), one must choose hyperparameters carefully to avoid NaN values. The last column of Table 4 shows the number of hyperparameter combinations that lead to NaN values during sampling.

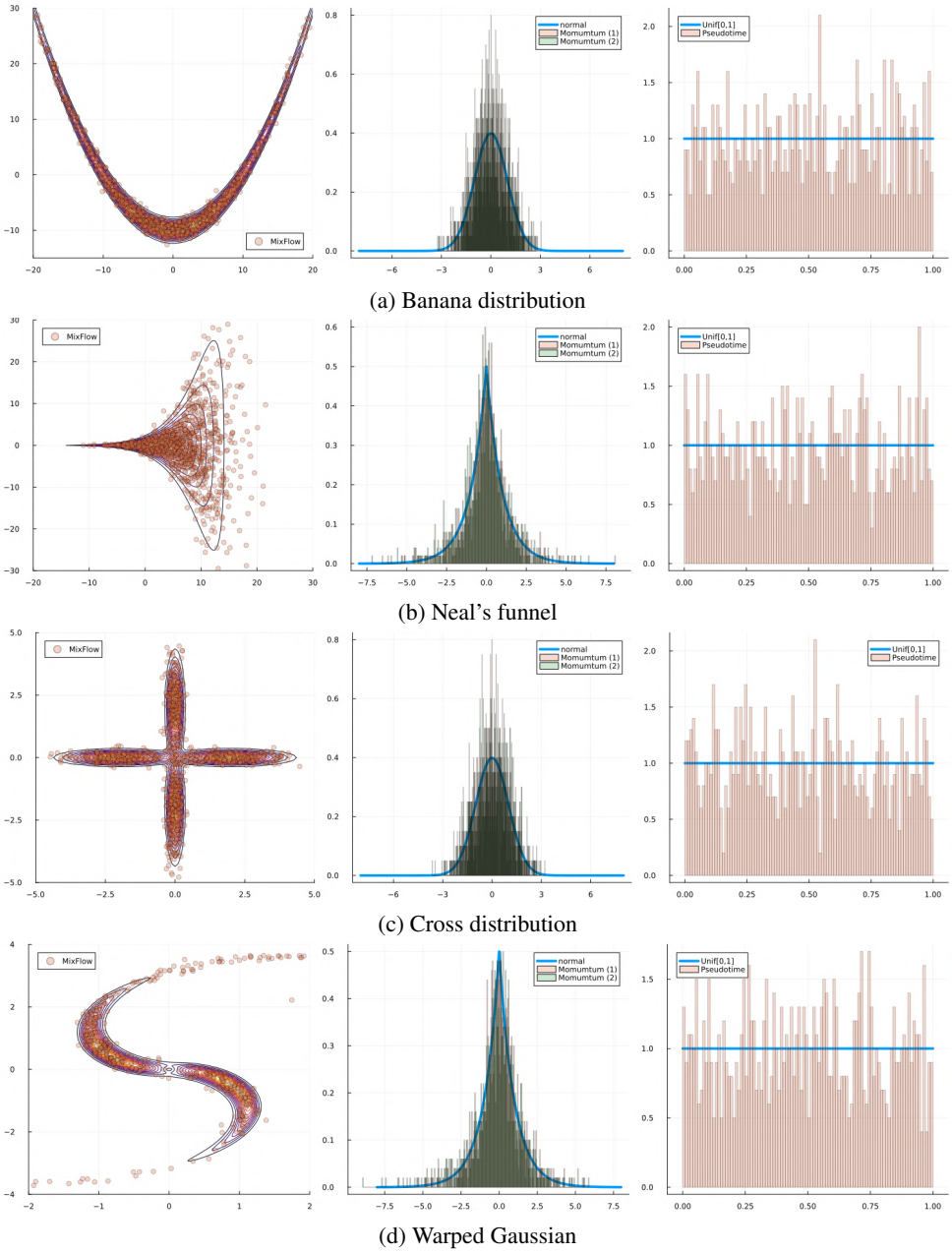
Figure 5b compares the joint ELBO values across various leapfrog step sizes for `MixFlow` against that of `NF`. We see that in all four examples, `NF` produces a smaller ELBO than `MixFlow` with a reasonable step size, which implies a lower quality target approximation. Indeed, Figure 13 shows that the trained `NF`s fail to capture the shape of the target distributions. Although one may expect the performance of `NF` to improve if it were given more layers, we will show that this is not the case in a later paragraph.

Figure 14a compares the sampling efficiency of `MixFlow` against `NUTS`, `HMC`, and `NF`. We see that `MixFlow iid` is the slowest, because each sample is generated by passing through the entire flow. However, we see that by taking all intermediate samples as in `MixFlow single`, we can generate samples just as fast as `NUTS` and `HMC`. On the other hand, while `NF` is fastest for sampling, it requires roughly 2 minutes for training, which alone allows `MixFlow single`, `NUTS`, and `HMC` to generate over 1 million samples in these examples. A more detailed discussion about this trade-off for `NF` is presented later.

Figure 14b further shows the computational efficiency in terms of effective samples size (ESS) per second. The smaller per second ESS of `MixFlow iid` is due to its slower sampling time. However, we emphasize that these samples are i.i.d.. `NUTS` overall achieves a higher per second ESS. `NUTS` is performant because of the much longer trajectories it produces (it only terminates once it detects a "U-turn"). This is actually an illustration of a limitation of the ESS per unit time as a measurement of performance. Because NUTS generates longer trajectories, it has a lower sample autocorrelation and a higher ESS; but Figure 15 shows that the actual estimation performance of `NUTS` is comparable to the other methods. Note that it is also possible to incorporate the techniques used in `NUTS` to our method, which we leave for future work.

As mentioned above, ESS mainly serves as a practical index for the relative efficiency of using a sequence of dependent samples, as opposed to independent samples, to estimate certain summary statistics of the target distribution. In this case, `MixFlow single` can be very useful. Figure 15 demonstrate the performance of `MixFlow single`, `NUTS`, `HMC`, and `NF` when estimating the coordinate-wise means and standard deviations (SD) of target distributions. We see that `MixFlow single`, `NUTS`, and `HMC` generally show similar performance in terms of convergence speed and estimation precision. While `NF` converges very quickly due to i.i.d. sampling, it does seem to struggle more at identifying the correct statistics, particularly the standard deviation, given its limited approximation quality. It is worth noting that, unlike `MixFlow` and general MCMC methods, sample estimates of target summaries obtained from `NF` are typically not asymptotically exact, as the sample quality is fundamentally limited by the choice of variational family and how well the flow is trained.

Finally, we provide an additional set of results for `NF` (Figure 16), examining its performance as we increase the number of planar layers $(5, 10, 20, 50)$. All settings for `NF` are identical to the above, except that we increase the optimization iteration to $500,000$ to ensure the convergence of flows with increased numbers of layers. As demonstrated in the second



(a) Banana distribution

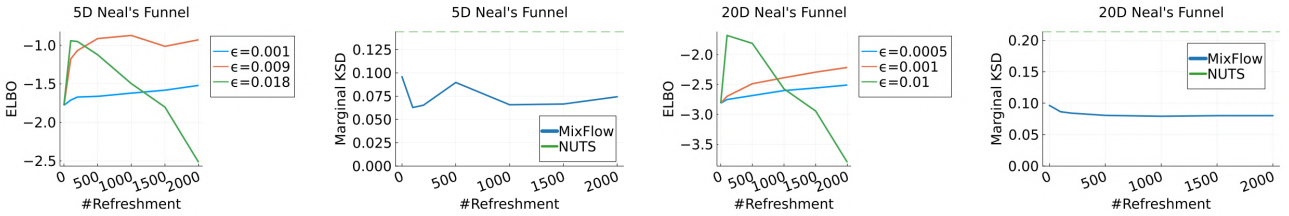
(b) Neal's funnel

(c) Cross distribution

(d) Warped Gaussian

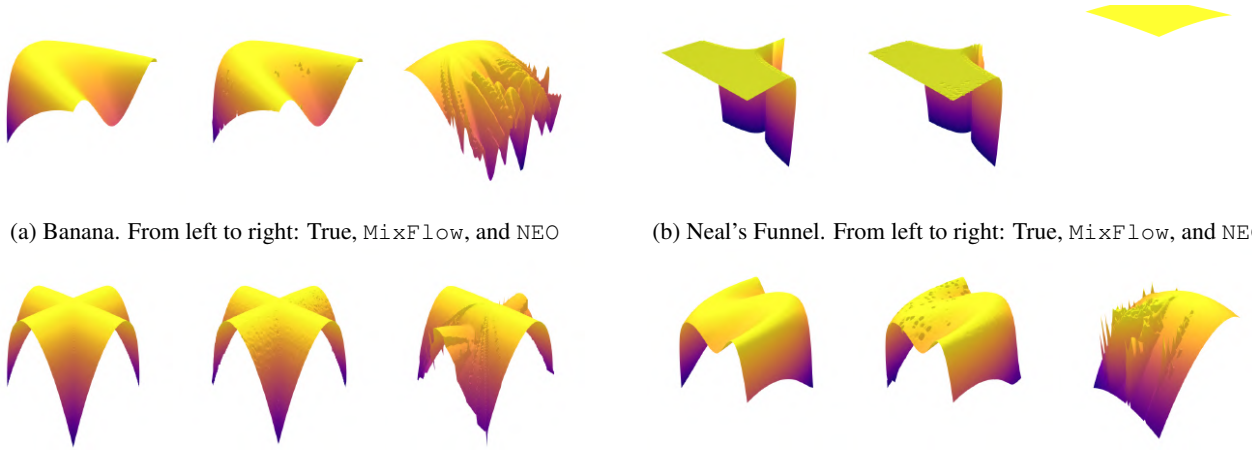
Figure 9. Scatter plots and histograms for the x-, ρ-, and u-marginals in the four synthetic experiments.

and third column of Figure 16, both training time and sampling time scale with the number of layers roughly linearly. Although a trained `NF` is still generally faster in sample generation (see also Figure 14a), for these synthetic examples with 4-dimensional joint target distributions, training time can take up to 30 minutes. More importantly, the corresponding target approximations of `NF` are still not as good as those of `MixFlow` in all four examples, even when we increase the number of layers to 50, which corresponds to optimizing 450 parameters from the flow. One may also notice from Figure 16 that a more complex normalizing flow does not necessarily lead to better performance. This is essentially because the (usually non-convex) KL optimization problem of standard `NF` becomes more complex to solve as the flow becomes more flexible. As a result, even though theoretically, `NF` becomes more expressive with more layers, there is no guarantee on how well it approximates the target distribution. In contrast, `MixFlow` is optimization-free and is asymptotically exact—with a proper choice of hyper-parameter, more computation typically leads to better performance (Figure 5b). With the 30 minutes training time of `NF`, `MixFlow iid` can generate $18,000$ i.i.d. samples and `MixFlow single` can generate over 10



(a) (b) (c) (d)

Figure 10. ELBO versus step size (10a,10c) and KSD comparison with `NUTS` (10b,10d) for the 5- and 20-dimensional Neal's funnel examples.



(a) Banana. From left to right: True, `MixFlow`, and `NEO`

(b) Neal's Funnel. From left to right: True, `MixFlow`, and `NEO`

(c) Cross. From left to right: True, `MixFlow`, and `NEO`

(d) Warped Gauss. From left to right: True, `MixFlow`, and `NEO`

Figure 11. Visualization of sliced exact (left) and `MixFlow`-approximated (middle) joint log density (middle), and sliced joint log density of tuned `NEO` importance sampling proposal (right) for banana (11a), funnel (11b), cross (11c), and warped Gaussian (11d).

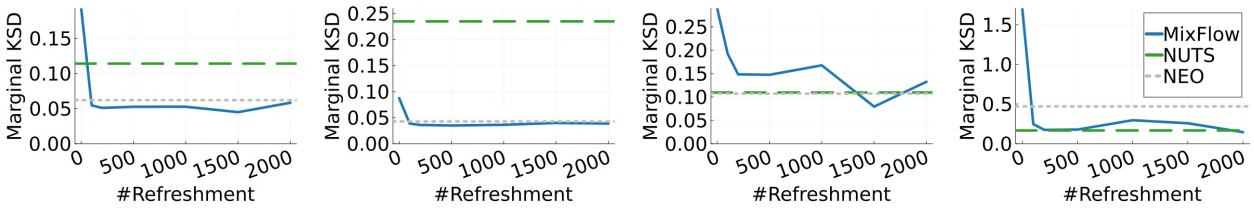


Figure 12. Marginal KSD of tuned `MixFlow` versus number of refreshments N, and comparison to `NUTS` and `NEO`

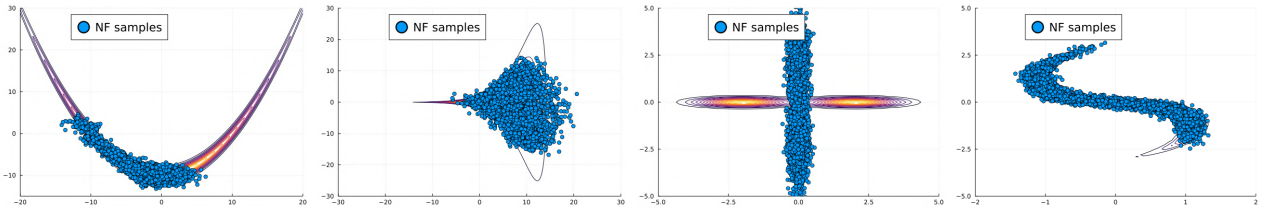
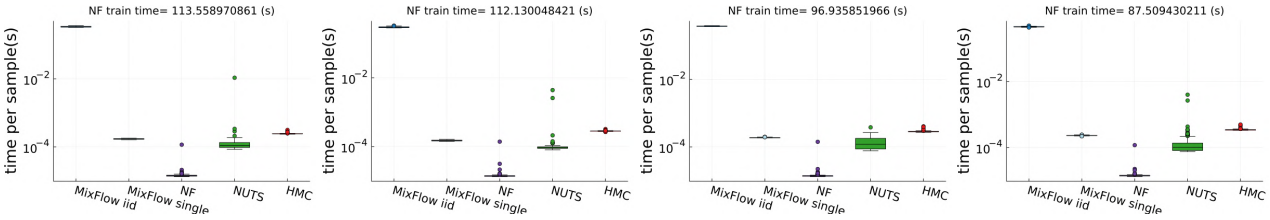
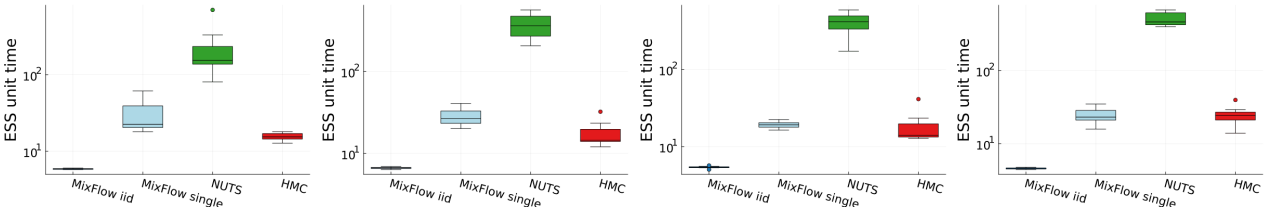


Figure 13. Scatter plot of 5000 i.i.d. samples generated from trained `NF`.

million samples, both of which are more than sufficient for most estimations under these target distributions.



(a): Sampling time comparison to NF,NUTS, and HMC (100 trials).



(b): Per second ESS comparison to NUTS and HMC (100 trials).

Figure 14. Time results for four multivariate synthetic examples: from left to right, each column corresponds to Banana, Neal's funnel, Cross, and warped Gaussian respectively.

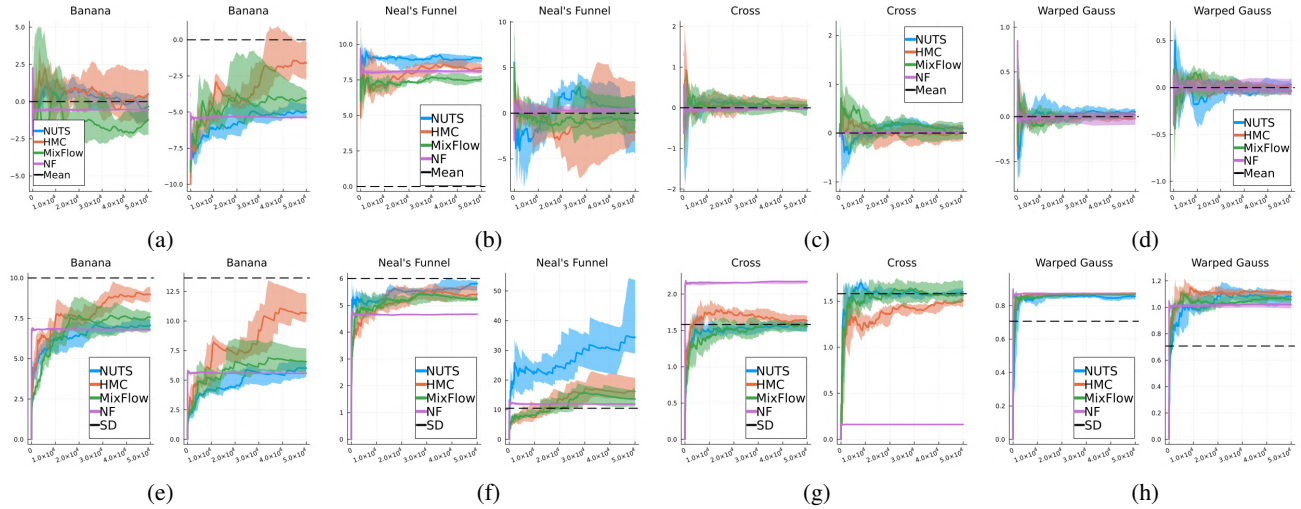
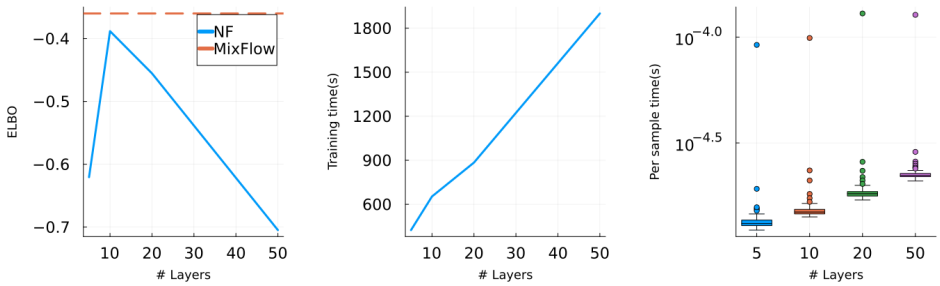


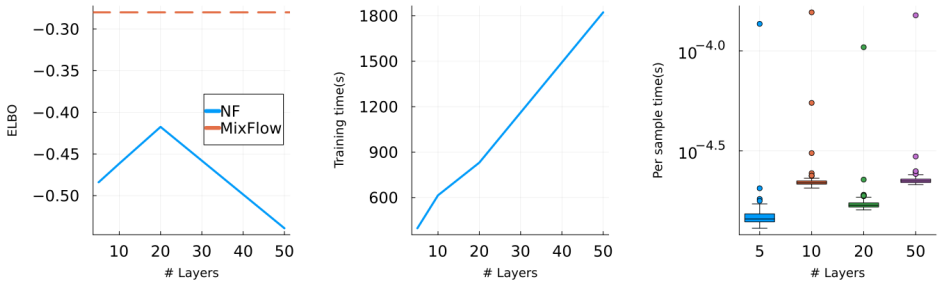
Figure 15. Streaming mean and standard deviation (SD) estimation ($50,000$ samples) for four 2-dimensional synthetic distributions. For each distribution, the two plots on the top row correspond to the two marginal means, and the two plots on the bottom row correspond to the two marginal SDs. Each plot shows the evolution of coordinate-wise mean/SD estimates, using samples from MixFlow single (green), NUTS (blue), HMC (red), and NF (purple); black dashed line indicates the true target mean/SD, which is estimated using $50,000$ samples from the actual synthetic target distribution. The lines indicate the median, and error regions indicate 25th to 75th percentile from 10 runs.

E.5. Real data examples

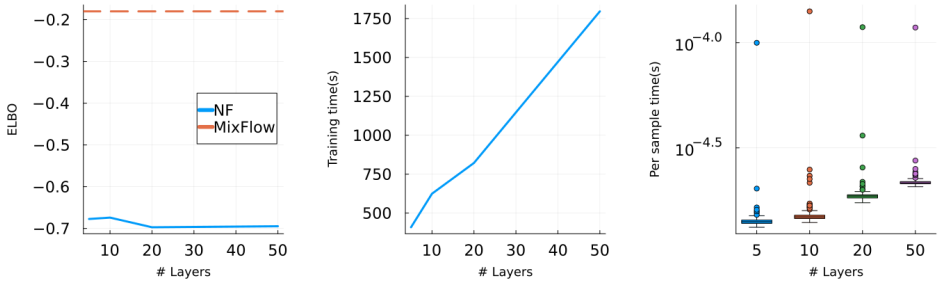
All settings for MixFlow, NUTS, NEO and HMC are identical to those in synthetic examples. All NFs are trained using ADAM for 200,000 iterations with initial step size set to be 0.001. We remark that comparison NEO is not included for high dimensional sparse regression example—the most challenging example—due to its high consumption of RAM, which hits the ceiling of our computation resources. Overall we observe similar phenomenons as for synthetic examples Appendix E.2. Additionally, we include comparison to UHA on real data examples. The tuning procedure of UHA involves architectural search (i.e., leapfrog steps between refreshments, and number of refreshments), and hyperparameter optimization (i.e., leapfrog step size, and tempering path). We choose among the combination of several fixed architectural settings of leapfrog steps between refreshments ($L = 10, 20, 50$), and number of refreshment ($N = 5, 10$), and optimize hyperparameters using ADAM for 20,000 iterations (each gradient estimate is based on 5 independent samples); each setting is repeated for 5 times.



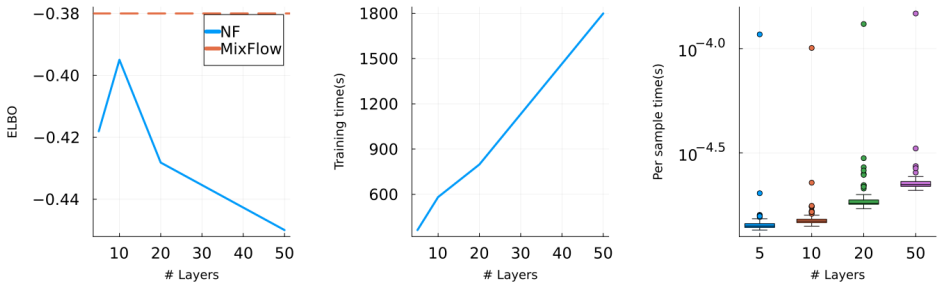
(a) Banana distribution



(b) Neal's funnel



(c) Cross distribution



(d) Warped Gaussian

Figure 16. Additional NF results with number of planar layers $(5, 10, 20, 50)$. First column shows the ELBO value. Each ELBO is estimated using 2000 i.i.d. samples from trained flow. The second and third columns correspond to the training time and per-sample time (100 trials) of NF given increasing planar layers.

Table 5 presents median, IQR of the resulting ELBOs of UHA under different architectural settings. We pick the best settings for UHA based on the ELBO performance and compare UHA under the selected settings with our method in terms of target approximation quality and marginal sample quality. Both ELBO and KSD of UHA are estimated using 5,000 samples.

Figure 17 offers a comparison of the ELBO performance between UHA and MixFlow. While MixFlow demonstrates similar or superior ELBO performance in the linear regression, logistic regression, student-t regression, and sparse regression problems, it is outperformed by UHA in the remaining three real data examples. The higher ELBO of UHA in these examples might be due to its incorporation of annealing, which facilitates exploration of complex target distributions. We leave studying the use of annealing in MixFlows to future work.

However, it is essential to note that for variational methods that augment the original target space, a higher ELBO does not necessarily equate to improved marginal sample quality. As demonstrated in Figure 18, the marginal sample quality of UHA, as evaluated by marginal KSD, is worse than that of MixFlow and the two Monte Carlo methods (NUTS and NEO), although it surpasses that of parametric flows (e.g. RealNVP).

It is also worth noting that tuning augmentation methods like UHA can be very expensive. For instance, in the Student-t regression problem, UHA required 20 minutes (1223.8 seconds) to run 20,000 optimization steps, with each gradient estimated using 5 Monte Carlo samples. This is approximately 40 times slower than RealNVP, which required around 10 minutes for 200,000 optimization steps with stochastic gradients based on 10 Monte Carlo samples. Conversely, evaluating one ELBO curve (averaged over 1,000 independent trajectories) for MixFlow across six different flow lengths (i.e., flow length = 100, 200, 500, 1000, 1500, 2000) only took 8 minutes, while adapting NUTS with 20,000 iterations was completed within a few seconds.

E.5.1. Bayesian linear regression

We consider two Bayesian linear regression problems, with both a standard normal prior and a heavy tail prior using two sets of real data. The two statistical models take the following form:

$$\log \sigma^2 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1), \quad y_j \mid \beta, \sigma^2 \overset{\text{indep}}{\sim} \mathcal{N}\left(x_j^T \beta, \sigma^2\right)$$
$$\textbf{Normal:} \beta \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1) \quad \textbf{Cauchy:} \beta \overset{\text{i.i.d.}}{\sim} \mathsf{Cauchy}(0,1)$$

where y_j is the response and $x_j \in \mathbb{R}^p$ is the feature vector for data point j. For linear regression problem with a normal prior, we use the Boston housing prices dataset (Harrison Jr. & Rubinfeld, 1978). Dataset available in the MLDatasets Julia package at https://github.com/JuliaML/MLDatasets.jl. containing $J = 506$ suburbs/towns in the Boston area; the goal is to use suburb information to predict the median house price. We standardize all features and responses. For linear regression with a heavy-tail prior, we use the communities and crime dataset (U.S. Department of Justice Federal Bureau of Investigation, 1995), available at http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime. The original dataset contains 122 attributes that potentially connect to crime; the goal is to predict per-capita violent crimes using the information of the community, such as the median family income, per capita number of poluce officers, and etc. For the data preprocessing, we drop observations with missing values, and using Principle component analysis for feature dimension reduction; we selected 50 principal components with leading eigenvalues. The posterior dimensions of the two linear regression inference problems are 15 and 52, repsectively.

E.5.2. Bayesian generalized linear regression

We then consider two Bayesian generalized linear regression problems—a hierachical logistic regression and a poisson regression:

$$\begin{aligned}
\textbf{Logis. Reg.:}\ & \alpha \sim \mathsf{Gam}(1, 0.01), \beta \mid \alpha \sim \mathcal{N}(0, \alpha^{-1} I), \\
& y_j \mid \beta \overset{\text{indep}}{\sim} \mathsf{Bern}\left(\frac{1}{1+e^{-x_j^T \beta}}\right), \\
\textbf{Poiss. Reg.:}\ & \beta \sim \mathcal{N}(0, I), \\
& y_j \mid \beta \overset{\text{indep}}{\sim} \mathsf{Poiss}\left(\log\left(1+e^{-x_j^T \beta}\right)\right),
\end{aligned}$$

For logistic regression, we use a bank marketing dataset (Moro et al., 2014) downsampled to $J = 400$ data points. Original dataset is available at https://archive.ics.uci.edu/ml/datasets/bank+marketing. the goal is to use client information to predict whether they subscribe to a term deposit. We include 8 features from the bank marketing dataset (Moro et al., 2014): client age, marital status, balance, housing loan status, duration of last contact, number of contacts during campaign, number of days since last contact, and number of contacts before the current campaign. For each of the binary variables (marital

status and housing loan status), all unknown entries are removed. All features of the dataset are also standardized. Hence the posterior dimension of the logistic regression problem is 9 and the overall (x, ρ, u) state dimension of the logistic regression inference problems are 19.

For Poisson regression problem, we use an airport delays dataset with 15 features and $J = 500$ data points (subsampled), resulting a 16-dimensional posterior distribution. The airport delays dataset was constructed using flight delay data from http://stat-computing.org/dataexpo/2009/the-data.html and historical weather information from https://www.wunderground.com/history/. relating daily weather information to the number of flights leaving an airport with a delay of more than 15 minutes. All features are standardized as well.

E.5.3. Bayesian Student-t regression

We also consider a Bayesian Student-t regression problem, of which the posterior distribution is heavy-tail. The Student-t regression model is as follows:

$$y_i \mid X_i, \beta \sim \mathcal{T}_5(X_i^T \beta, 1), \quad \beta \overset{\text{i.i.d.}}{\sim} \mathsf{Cauchy}(0, 1).$$

In this example, we use the creatinine dataset (Liu & Rubin, 1995), containing a clinical trial on 34 male patients with 3 covariates. Original dataset is available in https://github.com/faosorios/heavy/blob/master/data/creatinine.rda. The 3 covariates consist of body weight in kg(WT), serum creatininte concentration (SC), and age in years. The goal is to predict the endogenous cretinine clearance (CR) using these covariates. We apply the data transformation recommended by Liu & Rubin (1995) by transferring response into $\log(\text{CR})$, and transferring covariats into $\log(\text{WT})$, $\log(\text{SC})$, $\log(140 - \text{age})$.

E.5.4. Bayesian sparse regression

Finally, we compare the methods on the Bayesian sparse regression problem applied to two datasets: a prostate cancer dataset containing 9 covariates and 97 observations, and a superconductivity dataset (Hamidieh, 2018), containing 83 features and 100 observations (subsampled). The prostate cancer dataset is available at https://hastie.su.domains/ElemStatLearn/datasets/prostate.data. The superconductivity dataset is available at https://archive.ics.uci.edu/ml/datasets/superconductivty+data. The model is as follows:

$$\log \sigma^2 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1), \beta_i \overset{\text{i.i.d.}}{\sim} \frac{1}{2}\mathcal{N}(0, \tau_1^2) + \frac{1}{2}\mathcal{N}(0, \tau_2^2),$$
$$y_j \mid \beta, \sigma^2 \overset{\text{indep}}{\sim} \mathcal{N}\left(x_j^T \beta, \sigma^2\right)$$

For both two datasets, we set $\tau_1 = 0.1, \tau_2 = 10$. The resulting posterior dimension for both datasets are 10 and 84 respectively. When data information is weak, the posterior distribution in this model typically contains multiple modes (Xu & Campbell, 2022). We standardize the covariates during the preprocessing procedure for both datasets.

E.5.5. Additional experiment for real data examples

Table 1. Comparison of ELBO between `MixFlow` and `RealNVP` with different number of layers on real data example. Each setting of `RealNVP` is run in 5 trials. ELBOs of `MixFlow` are estimated using 1000 independent trajectories and ELBOs of `RealNVP` are estimated using 2000 independent samples.

			MixFlow		
	STEPSIZE	**#LEAPFROG**	**#REFRESHMENT**	**ELBO**	**#SAMPLE**
Lin Reg	0.0005	30	2000	-429.98	1000
Lin Reg Heavy	0.000025	50	2500	117.1	1000
Log Reg	0.002	50	1500	-250.5	1000
Poiss Reg	0.0001	50	2000	82576.9	1000
Student t Reg	0.0008	80	2000	-145.41	1000
Sparse Reg	0.001	30	600	-125.9	1000
Sparse Reg High Dim	0.0012	50	2000	-538.36	2000
			RealNVP		
	#LAYER	**#HIDDEN**	**MEDIAN**	**IQR**	**#NAN**
Lin Reg	5	15	-429.43	(-429.56, -429.42)	0
Lin Reg	10	15	-429.41	(-429.43, -429.36)	1
Lin Reg Heavy	5	52	116.47	(116.34, 116.49)	0
Lin Reg Heavy	8	52	116.65	(116.56, 116.73)	3
Lin Reg Heavy	10	52	116.26	(116.11,116.41)	3
Log Reg	5	9	-250.76	(-250.76, -250.75)	3
Log Reg	8	9	-250.65	(-250.75, -250.64)	2
			RealNVP		
	#LAYER	**#HIDDEN**	**MEDIAN**	**IQR**	**#NAN**
Poiss Reg	3	16	82576.87	(82575.57, 82577.53)	1
Poiss Reg	5	16	82580.77	(82580.13, 82583.72)	2
Poiss Reg	8	16	N/A	N/A	5
Student t Reg	5	4	-145.52	(-145.53, -145.51)	0
Student t Reg	10	4	-145.52	(-145.52, -145.49)	0
			RealNVP		
	#LAYER	**#HIDDEN**	**MEDIAN**	**IQR**	**#NAN**
Sparse Reg	5	10	-126.33	(-126.36, -126.32)	0
Sparse Reg	8	10	-126.35	(-126.36, -126.33)	1
Sparse Reg High Dim	5	20	-531.75	(-533.41, -522.60)	0
Sparse Reg High Dim	8	20	N/A	N/A	5

Table 2. Comparison of ELBO between `MixFlow` and `PlanarFlow` with different number of layers on real data example. Each setting of `PlanarFlow` is run in 5 trials. ELBOs of `MixFlow` are estimated using 1000 independent trajectories and ELBOs of `PlanarFlow` are estimated using 2000 independent samples.

`MixFlow`

	STEPSIZE	#LEAPFROG	#REFRESHMENT	ELBO	#SAMPLE
Lin Reg	0.0005	30	2000	-429.98	1000
Lin Reg Heavy	2.50E-05	50	2500	117.1	1000
Log Reg	0.002	50	1500	-250.5	1000
Poiss Reg	0.0001	50	2000	82576.9	1000
Student t Reg	0.0008	80	2000	-145.41	1000
Sparse Reg	0.001	30	600	-125.9	1000
Sparse Reg High Dim	0.0012	50	2000	-538.36	2000

`PlanarFlow`

	#LAYER	MEDIAN	IQR	#NAN
Lin Reg	5	-434.73	(-435.01, -434.65)	0
Lin Reg	10	-438.79	(-439.25, -436.77)	0
Lin Reg	20	-442.4384675	(-444.64, -440.93)	0
Lin Reg Heavy	5	79.65	(-75.92, 99.10)	0
Lin Reg Heavy	10	23.13	(21.29, 100.10)	0
Lin Reg Heavy	20	-879.1909019	(-1641.19, -392.80)	0
Log Reg	5	-251.71	(-251.85, -251.41)	0
Log Reg	10	-251.8	(-252.03, -251.80)	0
Log Reg	20	-252.0937812	(-252.20, -251.93)	0

`PlanarFlow`

	#LAYER	MEDIAN	IQR	#NAN
Poiss Reg	5	82569.58	(82569.33, 82569.68)	0
Poiss Reg	10	82568.49	(82568.12, 82569.57)	0
Poiss Reg	20	82566.40441	(82552.57, 82568.05)	2
Student t Reg	5	-145.69	(-145.71, -145.64)	0
Student t Reg	10	-145.72	(-145.73, -145.69)	0
Student t Reg	20	-145.7579931	(-145.77, -145.72)	0

`PlanarFlow`

	#LAYER	MEDIAN	IQR	#NAN
Sparse Reg	5	-127.58	(-127.60, -127.42)	0
Sparse Reg	10	-127.75	(-127.87, -127.68)	0
Sparse Reg	20	-127.6677415	(-128.48, -127.65)	0
Sparse Reg High Dim	5	-571.97	(-572.09, -565.10)	0
Sparse Reg High Dim	10	-571.7	(-573.22, -565.55)	0
Sparse Reg High Dim	20	-577.1338885	(-580.30, -569.39)	0

Table 3. Comparison of ELBO between `MixFlow` and `RadialFlow` with different number of layers on real data example. Each setting of `RadialFlow` is run in 5 trials. ELBOs of `MixFlow` are estimated using 1000 independent trajectories and ELBOs of `RadialFlow` are estimated using 2000 independent samples.

`MixFlow`

	STEPSIZE	#LEAPFROG	#REFRESHMENT	ELBO	#SAMPLE
Lin Reg	0.0005	30	2000	-429.98	1000
Lin Reg Heavy	2.50E-05	50	2500	117.1	1000
Log Reg	0.002	50	1500	-250.5	1000
Poiss Reg	0.0001	50	2000	82576.9	1000
Student t Reg	0.0008	80	2000	-145.41	1000
Sparse Reg	0.001	30	600	-125.9	1000
Sparse Reg High Dim	0.0012	50	2000	-538.36	2000

`RadialFlow`

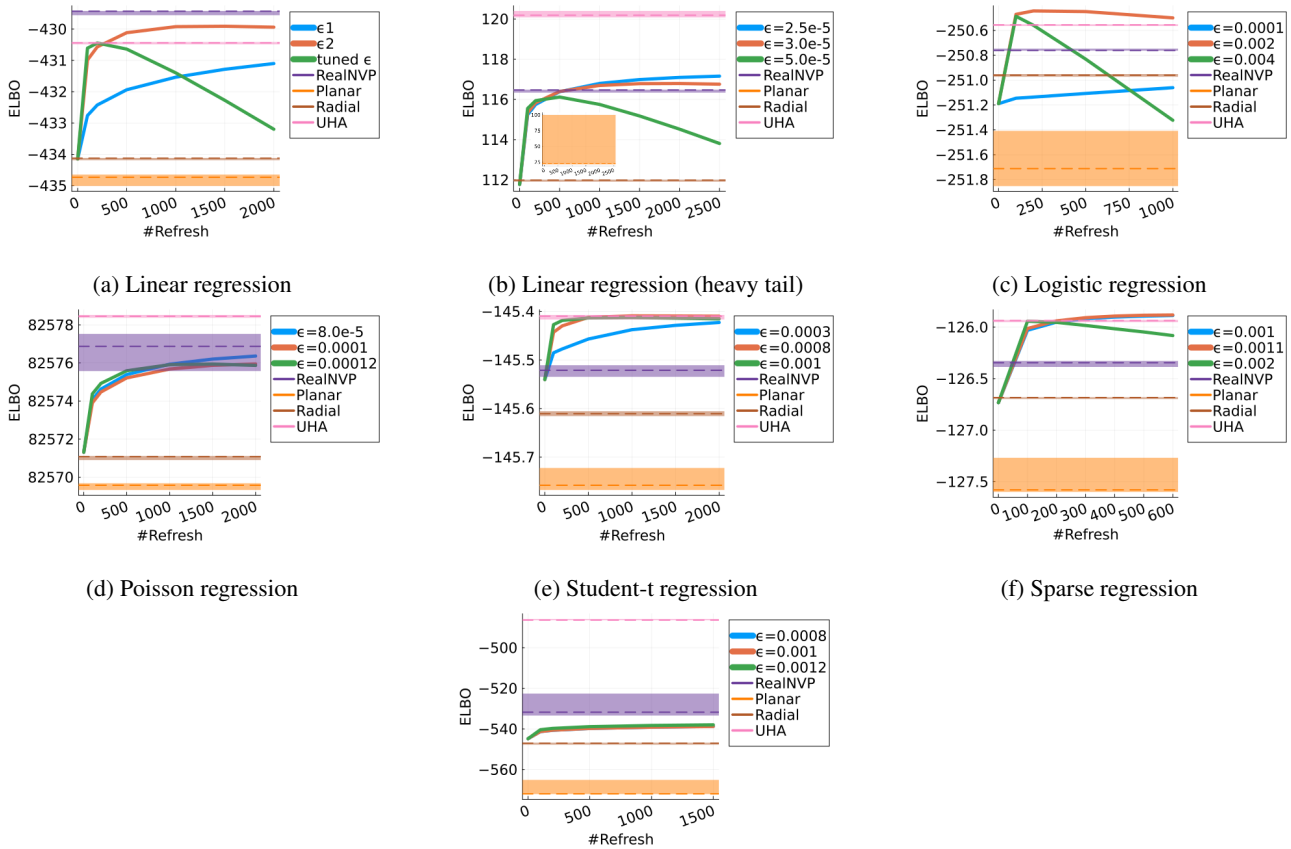
	#LAYER	MEDIAN	IQR	#NAN
Lin Reg	5	-434.14	(-434.36, -434.12)	0
Lin Reg	10	-434.12	(-434.19, -434.12)	0
Lin Reg	20	-433.82	(-434.21, -433.73)	0
Lin Reg Heavy	5	111.99	(111.91, 112.01)	0
Lin Reg Heavy	10	111.99	(111.89 , 112.00)	0
Lin Reg Heavy	20	111.7	(111.61, 111.71)	0
Log Reg	5	-251.33	(-251.49, -251.31)	0
Log Reg	10	-251.17	(-251.21 , -251.06)	0
Log Reg	20	-250.96	(-250.97, -250.95)	0

`RadialFlow`

	#LAYER	MEDIAN	IQR	#NAN
Poiss Reg	5	82570.5	(82570.19, 82570.57)	0
Poiss Reg	10	82571.02	(82570.76 , 82571.05)	0
Poiss Reg	20	82571.08	(82570.90, 82571.12)	0
Student t Reg	5	-145.61	(-145.62, -145.61)	0
Student t Reg	10	-145.59	(-145.60, -145.59)	0
Student t Reg	20	-145.58	(-145.60, -145.57)	0

`RadialFlow`

	#LAYER	MEDIAN	IQR	#NAN
Sparse Reg	5	-127.28	(-127.37, -127.27)	0
Sparse Reg	10	-127	(-127.03, -126.88)	0
Sparse Reg	20	-126.68	(-126.69, -126.61)	0
Sparse Reg High Dim	5	-548.05	(-548.54, -547.98)	0
Sparse Reg High Dim	10	-547.2	(-547.74, -547.16)	0
Sparse Reg High Dim	20	-547.03	(-547.78, -546.76)	0



(a) Linear regression

(b) Linear regression (heavy tail)

(c) Logistic regression

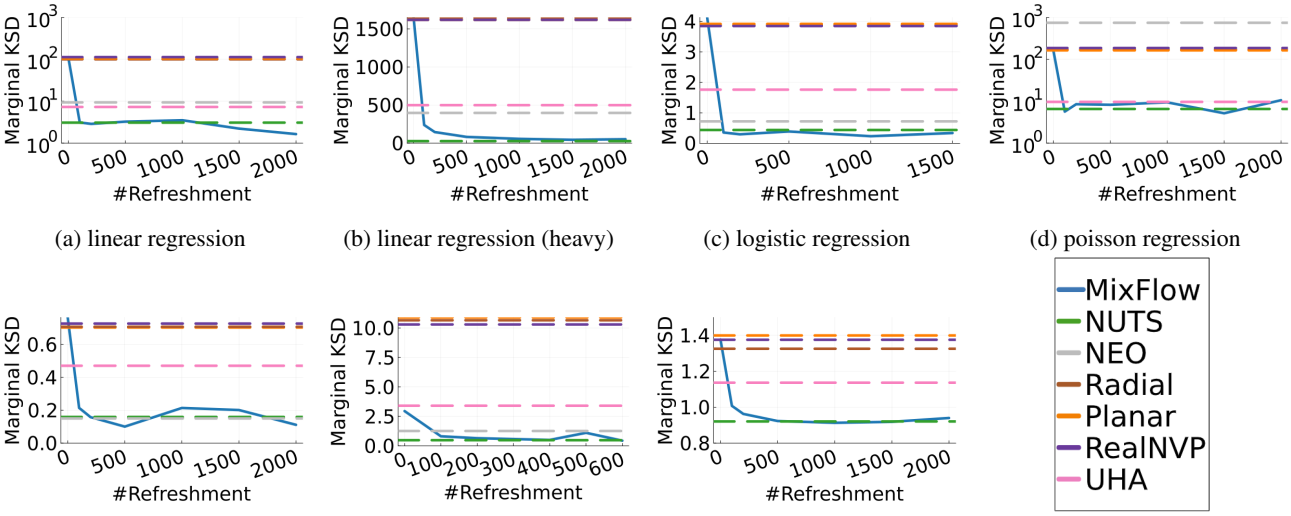
(d) Poisson regression

(e) Student-t regression

(f) Sparse regression

(g) Sparse regression (high dim)

Figure 17. ELBO comparison against NFs for real examples (except for linear regression, which is displayed in Figure 2a. Each NF method is tuned under various settings, and only the best one is present for each example. Each figure also shows the effect of step sizes on MixFlow; overly large or small step sizes influence the performance of MixFlow negatively. Lines indicate the median, and error regions indicate 25th to 75th percentile from 5 runs.



(a) linear regression

(b) linear regression (heavy)

(c) logistic regression

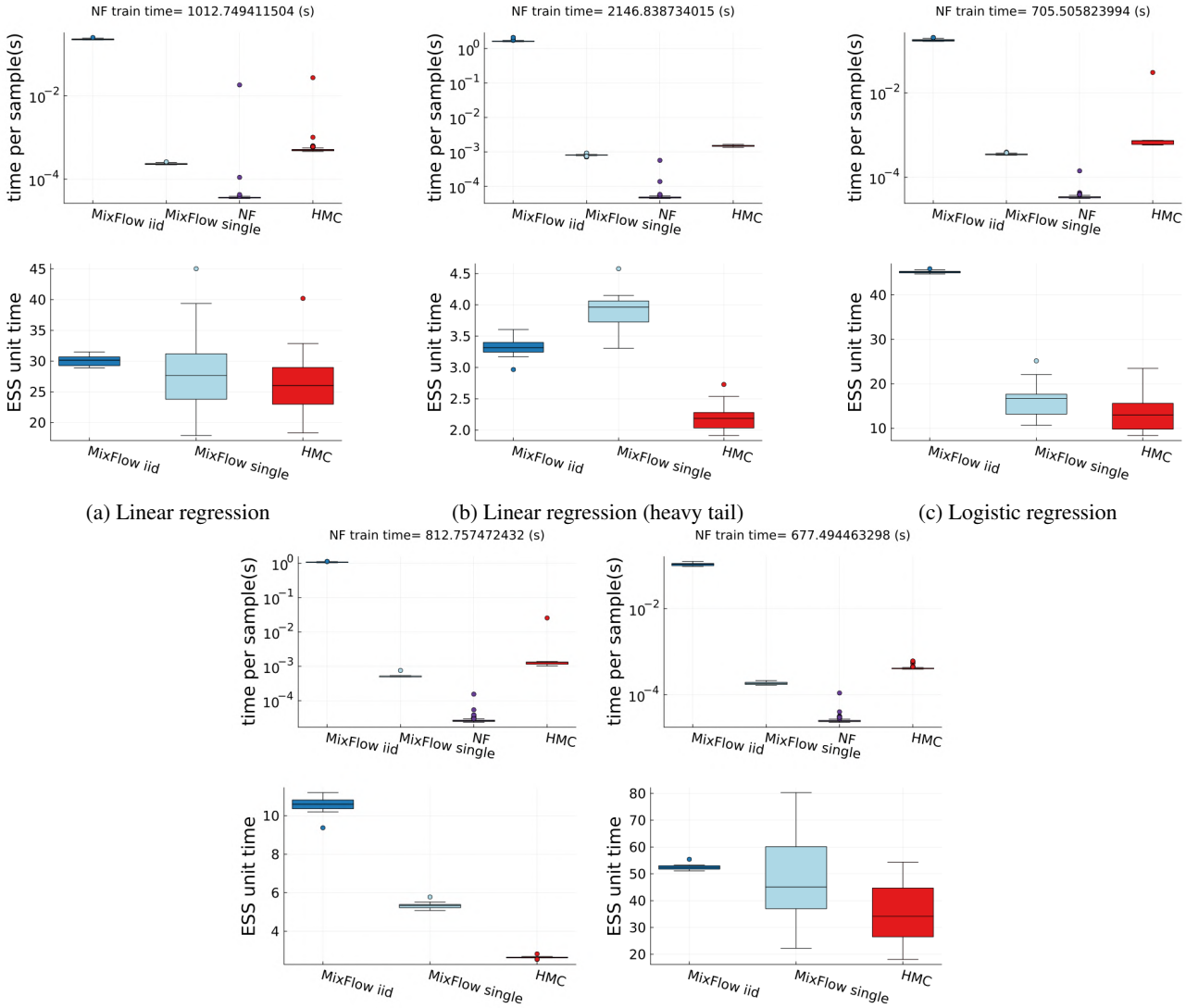
(d) poisson regression

(e) student t regression

(f) sparse regression

(g) sparse regression (high dim)

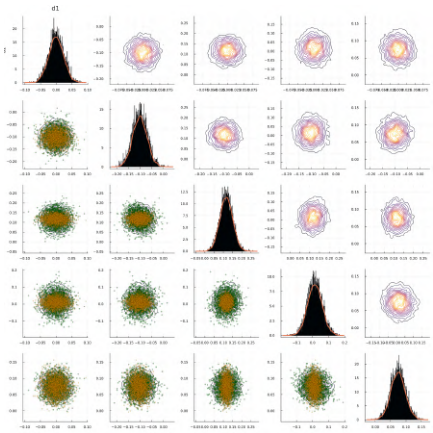
Figure 18. KSD comparison for real data examples.



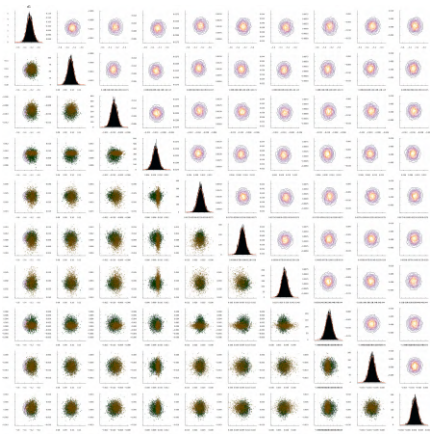
(a) Linear regression (b) Linear regression (heavy tail) (c) Logistic regression

(d) Poisson regression (e) Sparse regression

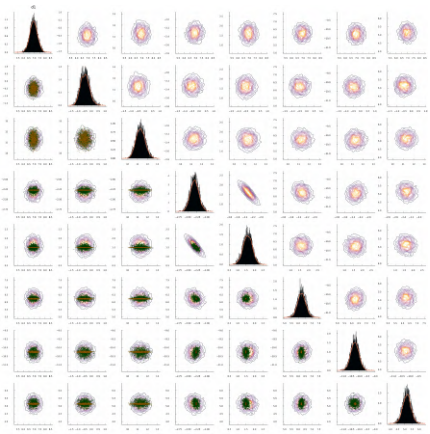
Figure 19. Time results (sampling time comparison to `NF`, `NUTS`, `NEO`, and `HMC`, and per second ESS comparison to `HMC`; each repeated 100 trials) for two linear regression problems (Figures 19a and 19b), two generalized linear regression problems (Figures 19c and 19d), and one sparse regression problem (Figure 19e).



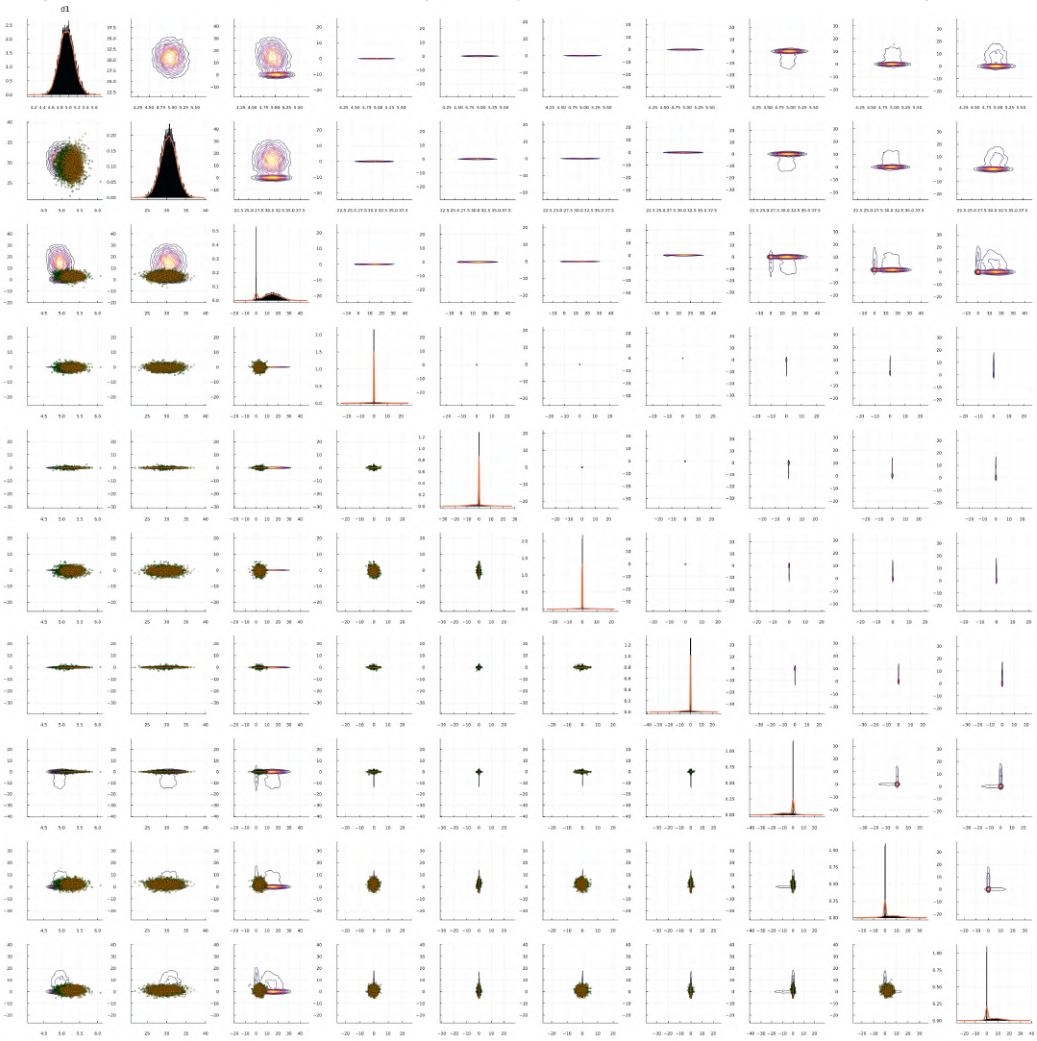
(a) Linear reg.: first 5 dimensions



(b) Linear reg. (Cauchy): first 10 dimensions



(c) Poisson reg.: all 8 dimensions



(d) Sparse reg. (high dim): first 10 dimensions

Figure 20. Sample quality visualization of 2,000 i.i.d. samples draw from each of `MixFlow` (green scatters) and `NF` (orange scatters) on 4 real data examples: linear regression, linear regression with heavy tail prior, poisson regression, and high-dimensional sparse regression. Pairwise kernel density estimation (KDE) is based on 20,000 `NUTS` samples, which is initialized with the Gaussian mean of the mean-field Gaussian approximation to posterior distribution, uses 20,000 steps for adaptation (targeting at an average acceptance rate 0.7). `NF` is chosen to be the same one as compared in Figures 4 and 19.

Table 4. Minimum, average, and standard deviation of KSD obtained from NEO for each of the examples across all tuning settings. #Fail indicates the number of tuning settings that resulted in NaN outputs. The final KSDs obtained from MixFlow are also included for comparion.

	MIXFLOW KSD	**MIN. KSD**	**AVG. KSD**	**SD. KSD**	**#FAIL**
banana	0.06	0.06	1.3	1.69	7/24
cross	0.13	0.11	2.78	3.09	12/24
Neal's funnel	0.04	0.04	0.76	1.84	6/24
warped Gaussian	0.15	0.23	0.55	0.25	0/14
linear regression	1.64	9.37	143.56	84.16	18/24
linear regression (heavy tail)	51.43	396.97	1589.71	599.47	18/24
logistic regression	0.34	0.72	19.28	11.7	18/24
Poisson regression	10.57	639.08	1642.81	670.8	18/24
t regression	0.11	0.15	4.79	5.03	12/24
sparse regression	0.44	1.26	43.83	25.58	18/24

Table 5. ELBO results of UHA with different number of leapfrogs and number of refreshments on real data examples. Each setting of UHA is run in 5 trials. ELBOs of UHA are estimated using 5000 independent samples.

	#LFRG	#REF	MEDIAN	IQR	#SAMPLE
Lin Reg	10	5	-430.438	(-430.488, -430.411)	5000
Lin Reg	20	5	-430.589	(-430.596, -430.541)	5000
Lin Reg	50	5	-432.215	(-432.268, -432.146)	5000
Lin Reg	10	10	-430.468	(-430.511, -430.457)	5000
Lin Reg	20	10	-431.404	(-431.441, -431.202)	5000
Lin Reg	50	10	-432.462	(-432.56, -432.327)	5000
Lin Reg Heavy	10	5	120.188	(120.08, 120.401)	5000
Lin Reg Heavy	20	5	119.126	(118.806, 119.796)	5000
Lin Reg Heavy	50	5	119.134	(118.994, 119.247)	5000
Lin Reg Heavy	10	10	118.222	(118.125, 118.245)	5000
Lin Reg Heavy	20	10	118.191	(118.151, 118.202)	5000
Lin Reg Heavy	50	10	118.059	(117.718, 118.175)	5000
Log Reg	10	5	-250.588	(-250.607, -250.553)	5000
Log Reg	20	5	-250.555	(-250.565, -250.551)	5000
Log Reg	50	5	-250.601	(-250.64, -250.567)	5000
Log Reg	10	10	-250.643	(-250.662, -250.584)	5000
Log Reg	20	10	-250.578	(-250.615, -250.563)	5000
Log Reg	50	10	-250.913	(-251.006, -250.839)	5000
Poiss Reg	10	5	82578.439	(82578.43, 82578.459)	5000
Poiss Reg	20	5	82578.453	(82578.372, 82578.522)	5000
Poiss Reg	50	5	82578.421	(82578.342, 82578.465)	5000
Poiss Reg	10	10	82578.45	(82578.435, 82578.488)	5000
Poiss Reg	20	10	82578.449	(82578.368, 82578.458)	5000
Poiss Reg	50	10	82578.369	(82578.322, 82578.414)	5000
Student t Reg	10	5	-145.46	(-145.461, -145.438)	5000
Student t Reg	20	5	-145.424	(-145.451, -145.423)	5000
Student t Reg	50	5	-145.427	(-145.43, -145.421)	5000
Student t Reg	10	10	-145.441	(-145.454, -145.404)	5000
Student t Reg	20	10	-145.439	(-145.466, -145.411)	5000
Student t Reg	50	10	-145.41	(-145.417, -145.407)	5000
Sparse Reg	10	5	-125.936	(-125.953, -125.931)	5000
Sparse Reg	20	5	-125.954	(-125.981, -125.944)	5000
Sparse Reg	50	5	-126.279	(-126.28, -126.226)	5000
Sparse Reg	10	10	-125.978	(-125.988, -125.931)	5000
Sparse Reg	20	10	-126.032	(-126.057, -125.961)	5000
Sparse Reg	50	10	-126.557	(-126.559, -126.525)	5000
Sparse Reg High Dim	10	5	-501.201	(-501.201, -500.778)	5000
Sparse Reg High Dim	20	5	-499.467	(-499.471, -499.387)	5000
Sparse Reg High Dim	50	5	-498.968	(-499.001, -498.656)	5000
Sparse Reg High Dim	10	10	-488.7	(-488.799, -488.561)	5000
Sparse Reg High Dim	20	10	-487.216	(-487.662, -487.042)	5000
Sparse Reg High Dim	50	10	-486.423	(-486.477, -485.83)	5000