Path-Guided Particle-based Sampling

Mingzhou Fan [*1] **Ruida Zhou** [*2] **Chao Tian** [1] **Xiaoning Qian** [1 3 4]

Abstract

Particle-based Bayesian inference methods by sampling from a partition-free target (posterior) distribution, e.g., Stein variational gradient descent (SVGD), have attracted significant attention. We propose a path-guided particle-based sampling (PGPS) method based on a novel Log-weighted Shrinkage (LwS) density path linking an initial distribution to the target distribution. We propose to utilize a Neural network to learn a vector field motivated by the Fokker-Planck equation of the designed density path. Particles, initiated from the initial distribution, evolve according to the ordinary differential equation defined by the vector field. The distribution of these particles is guided along a density path from the initial distribution to the target distribution. The proposed LwS density path allows for an efficient search of modes of the target distribution while canonical methods fail. We theoretically analyze the Wasserstein distance of the distribution of the PGPS-generated samples and the target distribution due to approximation and discretization errors. Practically, the proposed PGPS-LwS method demonstrates higher Bayesian inference accuracy and better calibration ability in experiments conducted on both synthetic and real-world Bayesian learning tasks, compared to baselines, such as SVGD and Langevin dynamics, etc.

[*]Equal contribution [1]Department of Electrical & Computer Engineering, Texas A&M University, College Station, Texas, USA [2]Department of Electrical and Computer Engineering, University of California, Los Angeles, CA, USA [3]Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA [4]Computational Science Initiative, Brookhaven National Laboratory, Upton, NY, USA. Correspondence to: Mingzhou Fan <mzfan@tamu.edu>, Xiaoning Qian <xqian@tamu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Bayesian learning is a powerful approach for distribution-based model predictions, naturally equipped with uncertainty quantification and calibration powers (Murphy, 2022). The key of Bayesian learning – computing the posterior by Bayes' rule, however, is well-known to be challenging due to the intractable partition function (a.k.a. the normalizing constant) (Andrieu et al., 2003).

To circumvent this difficulty, approaches based on sampling according to the (target) posterior distribution without computing the partition function have been considered; e.g., Markov Chain Monte-Carlo (MCMC) sampling (Andrieu et al., 2003) and its gradient-based variants (e.g., Langevin dynamics) generate samples (or *particles*) that follow the target distribution asymptotically using a partition-free function. Such particle-based Bayesian inference methods, which essentially transform a set of initial samples/particles along certain dynamics (e.g., an ordinary differential equation (ODE) or a stochastic differential equation (SDE)) governed by a vector field, have witnessed great successes (Liu, 2017). Most of these methods, e.g. Stein variational gradient descent (SVGD) (Liu & Wang, 2016) and preconditioned functional gradient flow (PFG) (Dong et al., 2022), fall into the category of *gradient-flow* particle-based sampling, where the vector field is a gradient function of the Kullback-Leibler (KL) divergence of the current distribution to the target distribution, such that the dynamics would drive the particles to the minimum of KL-divergence solution, i.e., the target distribution.

Although *gradient-flow* particle-based sampling methods are shown to be flexible and efficient in some applications (Dong et al., 2022), they may not achieve the ideal Bayesian inference performance due to not effectively capturing the posterior distribution. Specifically, as a realization of the KL-Wasserstein gradient-flow method, Langevin Dynamic (LD) is known to suffer from slow mixing, and in turn tends to result in mode missing or misplaced mode weights (Song & Ermon, 2019). It is believed that the posterior for complicated models, especially Bayesian Neural Networks (BNNs) (Goan & Fookes, 2020), contain multiple modes of different weights, and mode missing would impact its generalization, uncertainty quantification, and calibration abilities. More detailed discussions can be found

in Sections 3.1 and 5.1.

In this work, we propose a new family of Bayesian inference methods termed **Path-Guided Particle-based Sampling (PGPS)**. The particles follow an ODE defined by a learned vector field, so that the distribution of the particles is directed by a carefully designed *partition-free* path connecting the initial and the target distributions, instead of evolving along the direction that minimizes some functional. The performance of the PGPS approach is clearly determined by the path being followed, and we propose to rely on a *Log-weighted Shrinkage* path that is more efficient and accurate. The intuition for this choice is that logarithmic weights admit linear mixture of the score function and the shrinkage allows effective coverage of the target distribution along the path.

The contributions of this work are threefold:

1. We propose PGPS as a novel framework of flow-based sampling methods and derive a tractable criterion for any differentiable partition-free path in Proposition 3.1;

2. We theoretically show that the Wasserstein distance between the target distribution and the PGPS generated distribution following the NN-learned vector field with approximation error δ and discretization error by step-size h is bounded by $\mathcal{O}(\delta) + \mathcal{O}(\sqrt{h})$ in Theorem 4.2;

3. We experimentally verify the superior performance of the proposed approach over the state-of-the-art benchmarks, in terms of the sampling quality of faster mode seeking and more accurate weight estimating, and the inference quality with *higher testing accuracy* and *stronger calibration ability* in Bayesian inference, in Section 5.

2. Background

Given an inference model parameterized by parameters $\mathbf{x}$, e.g., a neural network with parameters $\mathbf{x}$, Bayesian inference updates the distribution of the parameters by Bayes' theorem, and performs statistical inference according to the posterior distribution. Specifically, suppose parameters $\mathbf{x} \in \mathbb{R}^d$ has prior density[1] $p_0(\mathbf{x})$, and given a data set $\mathcal{D}$, the posterior $p^*(\mathbf{x})$ is updated by $p^*(\mathbf{x}) = \frac{\hat{p}(\mathbf{x})}{Z}$ with $\hat{p}(\mathbf{x}) = p_0(\mathbf{x})L(\mathcal{D}|\mathbf{x})$, where $L(\mathcal{D}|\mathbf{x})$ is the likelihood function of the data $\mathcal{D}$ and $Z = \int p_0(\mathbf{x})L(\mathcal{D}|\mathbf{x})\,\mathrm{d}\mathbf{x}$ is the partition function. The partition function Z is usually computationally intractable. Many inference methods, including broadly applied Monte Carlo methods (Liu & Liu, 2001), have been proposed to (approximately) draw samples from the posterior/target distribution $p^*(\mathbf{x})$ using the more tractable partition-free function $\hat{p}(\mathbf{x})$.

[1]We assume density of the parameters (random variables) exists and do not differentiate their distribution and density.

Particle-based (particularly flow-based) Bayesian inference methods direct a set of random samples/particles $\{\mathbf{x}_0^{(i)}\}_{i=1}^n \subset \mathbb{R}^d$ drawn i.i.d. from an initial distribution p_0 (e.g., the prior or other distributions from which samples can be drawn directly) along certain ODE dynamics

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \boldsymbol{\phi}_t(\mathbf{x}_t), \quad \mathbf{x}_0 \sim p_0,$$

defined by a vector field $\boldsymbol{\phi}_t : \mathbb{R}^d \to \mathbb{R}^d$. The corresponding evolution of the density functions is characterized by the continuity equation (Jordan et al., 1998):

$$\frac{\partial}{\partial t} p_t(\mathbf{x}) = -\nabla \cdot (p_t(\mathbf{x})\boldsymbol{\phi}_t(\mathbf{x})), \tag{1}$$

where $p_t(\mathbf{x})$ denotes the density of $\mathbf{x}_t$, ∇ is the vector differential operator w.r.t. $\mathbf{x}$ (we omit $\mathbf{x}$ for simplicity throughout the paper), and $\nabla \cdot \boldsymbol{f}$ denotes the divergence of the vector function $\boldsymbol{f}$.

The critical point of the particle-flow-based methods is the design of the vector field $\boldsymbol{\phi}_t$. A typical choice is the gradient of some objective function under a certain metric, and the dynamic is thus a *gradient flow*. An example of the gradient-flow particle-based method is the Wasserstein gradient flow (Ambrosio et al., 2005), which has drawn considerable interest. It is motivated by minimizing a functional $L(p_t) \in \mathbb{R}$ in the Wasserstein space, which is a space of distributions equipped with the Wasserstein metric

$$W_q(p_1, p_2) = \left(\inf_{\gamma \in \Gamma(p_1, p_2)} \mathbb{E}_{(\mathbf{x}_1, \mathbf{x}_2) \sim \gamma} \|\mathbf{x}_1 - \mathbf{x}_2\|_q^q \right)^{1/q},$$

where $\Gamma(p_1, p_2)$ is the set of all the coupling of p_1 and p_2. When the functional is the KL divergence $\mathrm{KL}(p_t \| p^*) = \mathbb{E}_{\mathbf{x}_t \sim p_t}[-\ln p^*(\mathbf{x}_t) + \ln p_t(\mathbf{x}_t)]$ and under the 2-Wasserstein metric, i.e. $q = 2$, the resulting gradient has a closed form

$$\boldsymbol{\phi}_t(\mathbf{x}) = \nabla \ln p^*(\mathbf{x}) - \nabla \ln p_t(\mathbf{x}). \tag{2}$$

Under mild assumptions, the gradient flow converges to the optimal solution, i.e., $\lim_{t\to\infty} p_t = p^*$, which implies that with sufficiently large t, $\mathbf{x}_t$ approximately follows the target distribution p^*. Computing $\nabla \ln p_t(\mathbf{x})$ in Equation (2) is however not feasible in most practical cases. Methods such as learning the current density $p_t(\mathbf{x})$ (Wang et al., 2022), or transforming the problem of finding $\boldsymbol{\phi}_t(\mathbf{x})$ to a tractable learning/optimization problem (Dong et al., 2022; di Langosco et al., 2021) by a swarm of particles at step t, have been developed to implement the gradient flow. The vector field learning in our work is also based on a swarm of particles in the same manner, though the training loss function and the desired vector field are significantly different.

Evidently, the dynamics of the particles are not unique given the evolution of the distributions. Langevin dynamics (LD) $\mathrm{d}\mathbf{x}_t = \nabla \ln \hat{p}(\mathbf{x}_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\mathbf{B}_t$, where $\mathbf{B}_t$ is a standard Brownian motion, is a realization of the KL Wasserstein gradient flow (Jordan et al., 1998). In other words, its distribution satisfies the same Fokker-Planck equation as that of the ODE with vector field $\boldsymbol{\phi}_t(\mathbf{x})$ in (2). LD and its variants, such as Metropolis-adjusted Langevin Algorithm (MALA or LMC) and Stochastic Gradient Langevin Dynamics (SGLD), have been shown to be effective since they do not require learning $\nabla \ln p_t(\mathbf{x})$ as in (2). However, these methods lack a stopping criterion due to their stochastic nature (Dong et al., 2022), and can suffer from slow convergence for some target distributions.

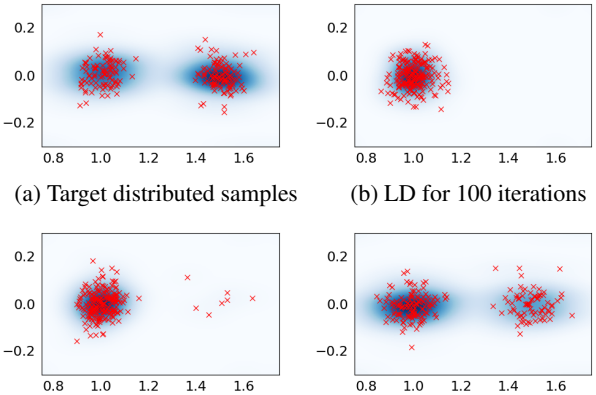
2.1. Motivation of the Proposed PGPS Method

We first pinpoint the cause of the slow convergence of KL Wasserstein gradient flow (e.g., LD), and provide the intuition for the proposed method as a remedy. Consider the experiment setup with a target distribution being a mixture of two Gaussian distributions, as shown in Figure 1 (a). Taking a zero mean isometric Gaussian as initial distribution, the convergence of LD to the target distribution is extremely slow as shown in Figure 1 (b-c), where the particles "stuck" at the left-hand-side mode of the Gaussian mixture and it takes many iterations to reach the right-hand-side mode. The reason for this behavior is that LD and similar gradient-flow-based methods rely heavily on the target distribution, which is an asymptotic target. Such an asymptotic target does not reflect the short-term need to escape from the current mode; i.e., the convergence to the target distribution can be extremely slow. To solve this issue, we propose PGPS which specifies a density evolution path directly connecting the initial and target distribution, and let the distribution of the particles evolve along such a path. At each time step, a short-term intermediate target on the path is set for the particles; more details are given in Section 3. As shown in Figure 1 (d), PGPS indeed finds both modes and converges to the target distribution with considerably fewer iterations.

In the following, we denote the unnormalized density function with a hat as $(\hat{\cdot})$, i.e. $\hat{p}_t(\mathbf{x}) \propto p_t(\mathbf{x})$ with $\int p_t(\mathbf{x})\,\mathrm{d}x = 1$ but $\int \hat{p}_t(\mathbf{x})\,\mathrm{d}\mathbf{x}$ being an unknown positive number.

2.2. Related Works

Gradient-flow particle-based sampling usually aims at finding tractable estimations for the KL-gradient flows in the Wasserstein space. One track of works relies on the universal approximation theorem of neural networks (Hornik et al., 1989) to approximate the gradient-flow and maximize certain discrepancies (di Langosco et al., 2021; Grathwohl et al., 2020; Hu et al., 2018; Dong et al., 2022), among which preconditioned functional gradient flow (PFG) (Dong et al., 2022) was proposed to learn the Wasserstein gradient by a neural network with preconditioning for better approximation. Probability flow ODE (Maoutsa et al., 2020) can also be applied to learn the Wasserstein gradient flow aiming at learning the vector field for a given probability flow. Aside from focusing on the flow approximation, works focusing on the discretization adopt the Jordan, Kinderlehrer, and Otto (JKO) scheme (Jordan et al., 1998), aiming at finding a JKO operator that minimizes the target functional as well as the movement of the particles in each step, has also achieved good performance in arbitrary gradient flow other than KL-gradient flow estimation tasks (Alvarez-Melis et al., 2021; Mokrov et al., 2021).



(a) Target distributed samples (b) LD for 100 iterations

(c) LD for 4,000 iterations (d) PGPS for 650 iterations

Figure 1: An illustration of the effectiveness of PGPS over LD in handling mode-missing.

The Stein Variational Gradient Descent (SVGD) (Liu & Wang, 2016) can be viewed as a specific type of gradient flow w.r.t. the KL-divergence under a metric induced by Stein operator, i.e., approximating the gradient by a kernel function (Liu, 2017). It inspires later works on kernel methods following different flows, e.g., Fisher–Rao Flow (Maurais & Marzouk, 2024) and the flow introduced by minimizing first and second moments (Wang & Nüsken, 2024). However, the curse of dimensionality for the kernel-based methods leads to the particle collapse in SVGD (Ba et al., 2021), i.e., variance collapse. Projecting the inference space to a lower dimension can naturally avoid high-dimensional variational inference (Chen & Ghattas, 2020; Gong et al., 2020; Liu et al., 2022).

Another area of related work is the annealing-based methods, e.g., parallel tempering (Earl & Deem, 2005), annealed importance sampling (Neal, 2001), and sequential Monte Carlo (Doucet et al., 2001). Annealing-based methods utilize intermediate distributions, usually following a log-weighted schedule where the weights are usually interpreted as temperature, to help achieve better performance. Utilizing intermediate distributions (path) has witnessed benefits in both Monte-Carlo estimators (Grosse et al., 2013; Chehab

et al., 2024) and sampling tasks (Heng et al., 2020).

Learning vector fields to update the particles has been broadly adopted in generative models that consider a different task aiming at generating new samples based on existing ones. The backward process of diffusion models (Song et al., 2020; Albergo et al., 2023) is indeed learning the vector fields that can drive the particles inverting the path introduced by the forward process. Flow matching (Lipman et al., 2022), built on Continuous Normalizing Flows (CNF) (Kobyzev et al., 2021) to learn a vector field following some specifically designed path, has demonstrated its empirical effectiveness followed by justification from Benton et al. (2023) theoretically.

In this work, we use SVGD, PFG and LD as benchmarks and defer more detailed discussions on related works to Appendix A.

3. Path-Guided Particle-based Sampling

We propose Path-Guided Particle-based Sampling (PGPS) methods based on a continuous density path linking initial distribution p_0 to target distribution $p_1 = p^*$, while only accessing the partition-free version of the target distribution $\hat{p}_1 = \hat{p}$. Compared to the annealing methods that also utilize intermediate distributions (path), PGPS learns a vector field that would drive the particles to the next intermediate distribution based on a predefined path in each step, which has not been studied previously to the best of our knowledge. In this section, we first derive a condition for viable guiding paths and present a novel class of log-weighted shrinkage paths. We then propose a learning algorithm to effectively approximate the path-guided flow.

Given a partition-free density process $\{\hat{p}_t\}_{t\in[0,1]}$ and its normalized densities $\{p_t\}_{t\in[0,1]}$, with $\hat{p}_0 = p_0$ being the initial distribution and p_1 being the target, assume that $\frac{\partial}{\partial t}\hat{p}_t$ and $\nabla_{\boldsymbol{x}}\hat{p}_t(\boldsymbol{x})$ exist for any $t \in [0,1]$ and $\boldsymbol{x}$ on the support. We wish to construct a vector field $\boldsymbol{\phi}_t : \mathbb{R}^d \to \mathbb{R}^d$ such that the process

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \boldsymbol{\phi}_t(\mathbf{x}_t), \quad \mathbf{x}_0 \sim p_0 \tag{3}$$

satisfies $\mathbf{x}_t \sim p_t$ for any $t \in [0,1]$. The following proposition establishes that determining $\boldsymbol{\phi}_t(\boldsymbol{x})$ does not require the partition function.

Proposition 3.1. *For a given partition-free density path $\{\hat{p}_t\}$, the gradient flow guided by the vector field $\boldsymbol{\phi}_t(\mathbf{x})$ following the continuity equation (1) satisfies:*

$$r(\boldsymbol{x}, \boldsymbol{\phi}_t) - \mathbb{E}_{\mathbf{x}\sim p_t}\left[\frac{\partial \ln \hat{p}_t(\mathbf{x})}{\partial t}\right] = 0, \tag{4}$$

where $r(\boldsymbol{x}, \boldsymbol{\phi}_t) = \frac{\partial \ln \hat{p}_t(\boldsymbol{x})}{\partial t} + (\nabla \ln \hat{p}_t(\boldsymbol{x}) + \nabla) \cdot \boldsymbol{\phi}_t(\boldsymbol{x})$.

The proof of Proposition 3.1 can be found in Appendix D.

Proposition 3.1 indicates that once a vector field $\boldsymbol{\phi}_t$ satisfying Equation (4) is obtained, we can generate samples following the distribution on the density path $\{p_t\}$ when particles evolve according to the vector field in Equation (3). Furthermore, note that Equation (4) is free of the intractable partition function, and we can thus learn the vector field $\boldsymbol{\phi}_t(\boldsymbol{x})$ by approximating it via a neural network that solves for Equation (4).

3.1. Selection of Path

One of the most important components of the proposed approach is the selection of partition-free guiding path $\{\hat{p}_t\}_{t\in[0,1]}$. Although any reasonable path linking the initial and target distributions is valid to direct particles according to Equation (3) as long as the corresponding vector field follows the condition in Proposition 3.1, certain paths that are more robust against democratization and more tractable for training are preferred and may have better performance in practice.

We propose a class of *Log-weighted Shrinkage* paths $\{\hat{p}_t^{\mathrm{LwS}}\}$ as follows

$$\begin{aligned}\ln \hat{p}_t^{\mathrm{LwS}}(\boldsymbol{x}) := (1-t)\ln p_0\left((1-\alpha t)\boldsymbol{x}\right) \\ + t \ln \hat{p}_1\left(\frac{\boldsymbol{x}}{\beta + (1-\beta)t}\right),\end{aligned} \tag{5}$$

where $\alpha \in [0,1]$ and $\beta \in (0,1]$ are controlling parameters.

It is straightforward to check that LwS paths are valid with $\ln \hat{p}_0^{\mathrm{LwS}}(\boldsymbol{x}) = \ln p_0(\boldsymbol{x})$ and $\ln \hat{p}_1^{\mathrm{LwS}}(\boldsymbol{x}) = \ln \hat{p}_1(\boldsymbol{x})$. Moreover, $\frac{\partial}{\partial t}\ln \hat{p}_t^{\mathrm{LwS}}$ and $\nabla \ln \hat{p}_t^{\mathrm{LwS}}$ both exist, when $\nabla \ln \hat{p}_1$ and $\nabla \ln p_0$ exist; see Appendix B.

As its name suggested, LwS paths (5) have two components – Log-weights and Shrinkage. The log-weights enable representing the log-distribution on the path by a linear mixture of the log-initial-distribution and log-target-distribution terms in Equation (5) weighted by $(1-t)$ and t. The linear mixture allows efficient computation of $r(\boldsymbol{x}, \boldsymbol{\phi}_t)$ in Proposition 3.1 when training $\boldsymbol{\phi}_t$ by a neural network. The Shrinkage operates on the initial-distribution term by α and the target-distribution term by β in Equation (5). The first term spreads the initial distribution by a factor $1/(1-\alpha t)$ to cover larger ranges as the factor increases along t; and the second term shrinks the target distribution $\hat{p}_1$ towards zero (i.e., the distribution $\hat{p}_1(\frac{\boldsymbol{x}}{\beta+(1-\beta)t})$ is thinner than $\hat{p}_1(\boldsymbol{x})$) by a factor $\beta + (1-\beta)t$. Since a typical choice of p_0 is zero-mean Gaussian, the shrinkage allows better coverage of the target distribution, and the coverage enables better mode seeking. It is illustrated in Figure 2 for different choices of the hyperparameters α, β. We can observe that with appropriate choices of hyperparameters (e.g., (B) and (C)), the right mode of the target distribution is detected at an early stage

(e.g., $t = 0.2$) compared to the log-weighted path without shrinkage (e.g., (A)). We further discuss the influence of the choice of the hyperparameters in Appendix C.

3.2. Learning Vector Field $\boldsymbol{\phi}_t(\boldsymbol{x})$

Given a viable path $\{p_t\}$, we aim to find a corresponding vector field $\boldsymbol{\phi}_t(\boldsymbol{x})$ as in Proposition 3.1 to direct the particles as in Equation (3). However, solving Equation (4) for $\boldsymbol{\phi}_t(\boldsymbol{x})$ in closed form is intractable. We use a parameterized vector field model $\boldsymbol{\phi}_t^\theta(\boldsymbol{x}) \in \mathbb{R}^d$ – a neural network parameterized by θ – to approximately solve for Equation (4).

Specifically, at each time step t starting with $t = 0$, we have N particles $\{\boldsymbol{x}_{t,(i)}\}_{i=1,\dots,N}$ and minimize the training loss

$$L_t(\theta) = \sum_{i=1\dots N} \left| r(\boldsymbol{x}_{t,(i)}, \boldsymbol{\phi}_t^\theta) - \frac{1}{N} \sum_{j=1\dots N} \frac{\partial \ln \hat{p}_t(\boldsymbol{x}_{t,(j)})}{\partial t} \right|^2 \tag{6}$$

resembling the squared value of the LHS of Equation (4). When particles $\{\boldsymbol{x}_{t,(i)}\}$ following distribution p_t, $\frac{1}{N}\sum_{j=1\dots N} \frac{\partial \ln \hat{p}_t(\boldsymbol{x}_{t,(j)})}{\partial t}$ is an unbiased estimate of $\mathbb{E}_{\mathbf{x}\sim p_t}\left[\frac{\partial \ln \hat{p}_t(\mathbf{x})}{\partial t}\right]$.

The training algorithm is presented in Algorithm 3, where the loss (6) is minimized by gradient descent. It is an iterative algorithm starting from time step $t = 0$ and is increased by Δt after the training for time t. The time step increment Δt is adaptively determined by Algorithm 1, which leads to a smaller increment for larger vector field $\boldsymbol{\phi}_t^\theta$ to control the movement of the particles. Since we have an intermediate target distribution $\hat{p}_t$ on the path to follow, an optional Langevin adjustment (Langevin dynamics w.r.t. the intermediate target $\hat{p}_t$) in Algorithm 2 can be applied to adjust the particles' distribution closer to p_t to reduce the biasedness in the loss function (6). We further discuss the Langevin Adjustment in the experiment section.

Algorithm 1 Adaptive Time Step

Input: Time t, Current particles $\{\boldsymbol{x}_{t,(i)}\}_{i=1\dots N}$, Flow $\boldsymbol{\phi}_t^\theta(\boldsymbol{x})$, Particle step-size ψ, Maximum time step $\Delta t'$;
$\Delta t \leftarrow (N\psi)/\sum_{i=1\dots N} \left\|\boldsymbol{\phi}_t^\theta(\boldsymbol{x}_{t,(i)})\right\|$;
$\Delta t \leftarrow \min\{\Delta t, 1-t, \Delta t'\}$;
Output: Time step Δt;

Training-free deployment of PGPS Many efficient algorithms such as LD or SVGD, are training-free, i.e., learning is not required during the evolution of the particles. We can also implement PGPS in a training-free manner, where at each time step t without training a neural network we update the particles by Langevin adjustment solely. In other words, we iteratively apply Langevin dynamics for sampling from an intermediate target distribution $\hat{p}_t$. A similar approach has been proposed under the name Annealed Langevin Dynamics (ALD) (Song & Ermon, 2019), where a path is given by changing the temperature of the target distribution. In Section 5.2.3, we experimentally compare the standard PGPS and the training-free PGPS and demonstrate the benefits of learning the vector field.

Algorithm 2 Langevin Adjustment

`// Langevin dynamics //`
Input: Particles $\{\boldsymbol{x}_{(i)}\}_{i=1\dots N}$, density $\hat{p}$;
Coefficients: Adjustment step-size δ, LD steps M';
for $k = 1 \dots M'$ **do**
  Sample $\{\xi_{(i)}^k\} \sim \mathcal{N}(0, I)$;
  Adjust $\{\boldsymbol{x}_{(i)}\} \leftarrow \{\boldsymbol{x}_{(i)} + \delta \nabla \ln \hat{p}(\boldsymbol{x}_{(i)}) + \sqrt{2\delta}\xi_{(i)}^k\}$;
end for
Output: Adjusted $\{\boldsymbol{x}_{(i)}\}$;

4. Theoretical Analysis

In this section, we study the distribution of the PGPS-generated particles compared to the target distribution. Note that the target distribution $p^* \propto \hat{p}$ equals to $p_{\mathbf{x}_1}$, where $\mathbf{x}_1 = \mathbf{x}_0 + \int_0^1 \boldsymbol{\phi}_t(\mathbf{x}_t)\,\mathrm{d}t$ with $\mathbf{x}_0 \sim p_0$ by Proposition 3.1. The PGPS method without Langevin adjustment simulates the integration by

$$\hat{\mathbf{x}}_{th+h}^\theta = \hat{\mathbf{x}}_{th}^\theta + \boldsymbol{\phi}_{nh}^\theta(\hat{\mathbf{x}}_{th}^\theta), \quad t = 0, \dots, n-1, \tag{7}$$

where $h = 1/n$ is the step size for some $n \in \mathbb{N}$ capturing the discretization error, and $\hat{\mathbf{x}}_0^\theta \sim p_0$.

We analyze the performance of PGPS using the 2-Wasserstein distance between the generated distribution $p_{\hat{\mathbf{x}}_1}$ and the target distribution $p_{\mathbf{x}_1}$ under the approximation error $\delta^2 := \int_0^1 \mathbb{E}_{\mathbf{x}\sim p_t}[\|\boldsymbol{\phi}_t^\theta(\mathbf{x}) - \boldsymbol{\phi}_t(\mathbf{x})\|^2]\mathrm{d}t$ and discretization error due to step size h in Theorem 4.2. The following assumptions are taken in the analysis.

Assumption 4.1.

(1) **Lipschitzness of $\boldsymbol{\phi}_t$ and $\boldsymbol{\phi}_t^\theta$ on $\boldsymbol{x}$ space**: There exists $K_1 < \infty$, such that $\|\boldsymbol{\phi}_t(\boldsymbol{x}_1) - \boldsymbol{\phi}_t(\boldsymbol{x}_2)\| \le K_1\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$ and $\|\boldsymbol{\phi}_t^\theta(\boldsymbol{x}_1) - \boldsymbol{\phi}_t^\theta(\boldsymbol{x}_2)\| \le K_1\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$ for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d$, $t \in [0,1]$;

(2) **Lipschitzness of $\boldsymbol{\phi}_t^\theta$ on t space**: There exists $K_2 < \infty$, such that $\|\boldsymbol{\phi}_{t_1}^\theta(\boldsymbol{x}) - \boldsymbol{\phi}_{t_2}^\theta(\boldsymbol{x})\| \le K_2|t_2 - t_1|$ for any $\boldsymbol{x} \in \mathbb{R}^d$, $t_1, t_2 \in [0,1]$;

(3) **Finite vector field**: There exists $K_3 < \infty$, such that $\|\boldsymbol{\phi}_t^\theta(\boldsymbol{x})\| \le K_3$ for any $\boldsymbol{x} \in \mathbb{R}^d$, $t \in [0,1]$

Theorem 4.2. *For two flows $\boldsymbol{\phi}_t^\theta(\mathbf{x})$ and $\boldsymbol{\phi}_t(\mathbf{x})$ under Assumption 4.1, the Wasserstein distance between the distribution $p_{\hat{\mathbf{x}}_1^\theta}$ of PGPS generated samples according to dynamics*

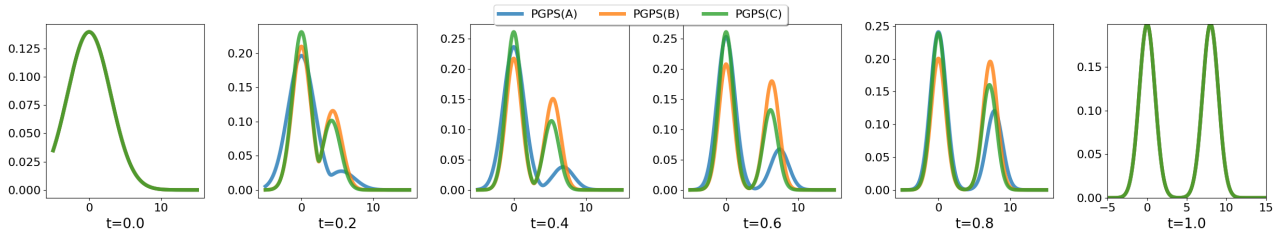


Figure 2: Different *Log-weighted Shrinkage paths* from the initial (left) to target (right) distribution with different hyper-parameters. (A):$\alpha = 0, \beta = 1$ (blue); (B):$\alpha = 1, \beta = 0.5$ (orange); (C):$\alpha = 0.2, \beta = 0.5$ (green).

Algorithm 3 PGPS

Input: Parameterized vector field $\boldsymbol{\phi}_t^\theta(\boldsymbol{x})$, Valid unnormalized path $\{\hat{p}_t\}_{t\in[0,1]}$, Particles from initial distribution $\{\boldsymbol{x}_{0,(i)}\}_{i=1\dots N}$, Maximum training steps M, Training threshold ϵ, Learning rate η, Maximum time step $\Delta t'$, Particle step-size ψ;
Initialize $t \leftarrow 0$;
repeat
  for $k = 1 \dots M$ **do**
    Gradient descent $\theta \leftarrow \theta - \eta \nabla_\theta L_{\mathbf{t}}(\theta)$;
    if $L_t(\theta) < \epsilon$ **then**
      Break;
    end if
  end for
  $\Delta t \leftarrow$ `Adaptive Time Step`$[t, \boldsymbol{\phi}_t^\theta(\boldsymbol{x}), \psi, \Delta t']$;
    `// Algorithm 1 //`
  Update $\{\boldsymbol{x}_{t+\Delta t,(i)}\} \leftarrow \{\boldsymbol{x}_{t,(i)} + \Delta t \boldsymbol{\phi}_t^\theta(\boldsymbol{x}_{t,(i)})\}$;
  Update $t \leftarrow t + \Delta t$;
  (Optional) $\{\boldsymbol{x}_{t,(i)}\} \leftarrow$
  `Langevin Adjustment`$[\{\boldsymbol{x}_{t,(i)}\}, \hat{p}_t]$;
    `// Algorithm 2 //`
until t = 1;
Output: Evolved particles $\{\boldsymbol{x}_{1,(i)}\}_{i=1\dots N}$;

(7) *and the target distribution* $p_{\mathbf{x}_1}$ *is bounded as*

$$W_2(p_{\hat{\mathbf{x}}_1^\theta}, p_{\mathbf{x}_1}) \leq \delta\sqrt{\exp(1+2K_1)} + \sqrt{h}\sqrt{\frac{C(\exp(1+K_1^2)-1)}{1+K_1^2}}, \quad (8)$$

where $C = \frac{1}{2}K_2^2 + \frac{17}{2}K_1^2K_3^2 + 5K_1K_2K_3$.

There are two terms in the upper bound Equation (8) by the approximation error and the discretization error, respectively. The first term is related to the Lipschitzness assumption on $\boldsymbol{\phi}_t(\boldsymbol{x}), \boldsymbol{\phi}_t^\theta(\boldsymbol{x})$ over $\boldsymbol{x}$ space (Assumption 4.1(1)). It characterizes the error introduced due to the approximation of the vector field $\boldsymbol{\phi}_t$ (Lemma D.2). The second term represents the error introduced by discretization, which is related to the Lipschitzness property with respect to t and the finiteness of the vector field (Assumption 4.1(2)-(3)). The proof of Theorem 4.2 can be found in Appendix D.

Theorem 4.2 indicates that with trained vector field of maximum error δ and discretized with uniform step h and the generated distribution is close to the target distribution with W_2-distance bounded by $\mathcal{O}(\delta) + \mathcal{O}(\sqrt{h})$. Therefore, we can improve the performance of the evolved particles by reducing the approximation error and/or refining the discretization. In the following, we illustrate that the training objective of minimizing loss function $L_t(\theta)$ in Equation (6) is aligned with reducing the approximation error.

Note that minimizing $L_t(\theta)$ is to solve the partial differential equation (PDE) in (4), which requires specifying the function space. Let $L^4(p_t)$ be the function space with norm $\|f\|_{L^4(p_t)} = (\int (f(\boldsymbol{x}))^4 p_t(\boldsymbol{x}) \mathrm{d}\boldsymbol{x})^{1/4}$ and $W^{1,4}(p_t) = \{f \in L^4(p_t) : \frac{\partial}{\partial x_i} f(\boldsymbol{x}) \in L^4(p_t)\}$ be a weighted Sobelov space. Denote by $\Psi_t = [W^{1,4}(p_t)]^d$ a product space that contains the vector-valued functions of interest. Specifically, we made mild assumptions below

Assumption 4.3. (a) $\boldsymbol{\phi}_t^\theta, \nabla \ln p_t \in \Psi_t$ for any $t \in [0,1]$; and (b) $\sup_{t\in[0,1]} \mathbb{E}_{\mathbf{x}\sim p_t}[\|\nabla \ln p_t(\mathbf{x})\|^4] < \infty$.

Proposition 4.4. *Under Assumption 4.3, for any* $\boldsymbol{\phi}_t^\theta$*, there exists a vector-field* $\boldsymbol{\phi}_t$ *solution to PDE* (4) *that*

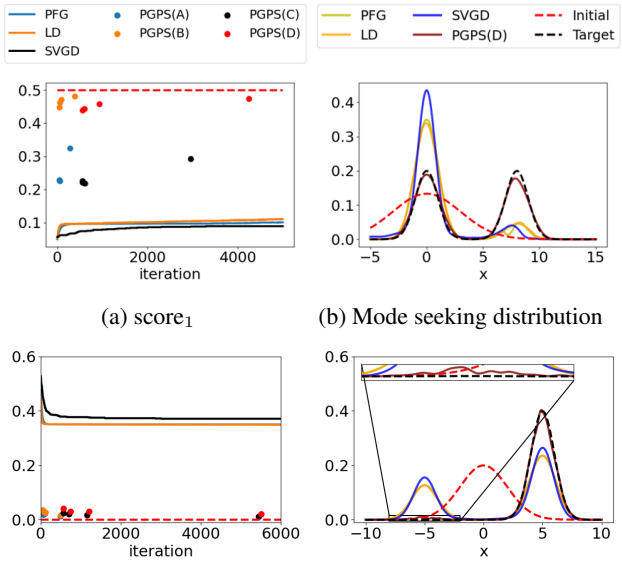
$$\mathbb{E}_{\mathbf{x}\sim p_t}[\|\boldsymbol{\phi}_t^\theta(\mathbf{x}) - \boldsymbol{\phi}_t(\mathbf{x})\|^2] \leq K L_t(\theta), \quad (9)$$

where $K > 0$ *is a universal constant factor and* $L_t(\theta)$ *is in Equation* (6) *with infinite many particles following* p_t.

Proposition justifies the consistency of the proposed method, i.e., $\boldsymbol{\phi}_t$ can be well-approximated by minimizing the loss function *under the infinite particles regime*. The impact of finite particles relies on the generalization analysis and is beyond the scope of the paper.

5. Experiments

We demonstrate the effectiveness of the proposed PGPS methods compared to LD, SVGD (Liu & Wang, 2016), PFG (Dong et al., 2022) baselines. The number of iterations for each method is the same, where the Langevin Adjustment steps in PGPS are counted. The code to reproduce the



(a) score$_1$ (b) Mode seeking distribution

(c) score$_2$ (d) Sensitivity distribution

Figure 3: The performances of different methods: (a, c) score$_1$ and score$_2$ indicating the mode capture ability with the true score illustrated by the red dashed line; (b, d) KDE estimated probability distributions for different methods. The letter following PGPS indicates different hyperparameters. (A): $\alpha = 0$, $\beta = 1$, steps $= 0$ (B): $\alpha = 1$, $\beta = 0.8$, steps $= 0$ (C): $\alpha = 0$, $\beta = 1$, steps $= 10$ (D): $\alpha = 1$, $\beta = 0.8$, steps $= 10$, where 'steps' indicates the number of performed Langevin Adjustment steps. We report the performance of PGPS with $\psi \in \{0.5, 0.1, 0.05, 0.01\}$.

experimetnal results can be found in our Github repository: https://github.com/MingzhouFan97/PGPS.

5.1. Gaussian Mixture Target Distribution

We study the mode-seeking and weight-estimation capabilities of the proposed PGPS for Gaussian mixture target distributions, compared to LD, SVGD, and PFG gradient-flow particle-based benchmarks.

5.1.1. Mode Discovery Missing

Given initial distribution $\mathcal{N}(0, 3^2)$ and target distribution of a mixture of two Gaussian distributions $\mathcal{N}(0, 1)$ and $\mathcal{N}(8, 1)$ with equal weights, we investigate whether the methods can effectively discover both modes.

Note that the left mode $\mathcal{N}(0, 1)$ of the target mixture is automatically discovered by the initial distribution $\mathcal{N}(0, 3^2)$. Define score$_1$ $= \frac{\sum_{i=1}^{N} \mathbb{I}(x_{t,(i)} > 5)}{N}$ to capture the rates of the samples discovering the right mode $\mathcal{N}(8, 1)$ by moving across threshold 5. The score$_1$ is shown in Figure 3a, where the dashed true score is $\mathbb{P}_{\text{target}}(\mathbf{x} > 0.5) \approx 0.499$. Note that the performances of PGPS methods are scattered because the method may require different adaptive iterations for different hyperparameter choices. With the fact that the intermediate state of the PGPS particle is not meaningful, scatter plots are selected rather than lines for LD, SVGD, and PFG. As shown in Figure 3a, PGPS recovers the right mode faster and better with score$_1$ close to the true score 0.49, yet the benchmarks fail. Figure 3b corroborates the finding by visualizing the output distribution of the sample methods, where PGPS-generated distribution is closer to the target.

5.1.2. False Mode Discovery – Sensitivity

The benchmarks not only fail to effectively discover modes but are also sensitive to the target distribution and may lead to false discovery, i.e., they may focus on some negligible mode.

Given initial distribution $\mathcal{N}(0, 2^2)$ and target distribution of a mixture of two Gaussian distributions $\mathcal{N}(-5, 1)$ and $\mathcal{N}(5, 1)$, where the left mode has an extremely small weight 0.001 and the right mode has weight 0.999. As shown in Figure 3d, the left mode is negligible and the target distribution is visually indistinguishable from a Gaussian distribution.

Define score$_2$ $= \frac{\sum_{i=1}^{N} \mathbb{I}(x_{t,(i)} < 0)}{N}$ to capture the rates of the samples focusing on the negligible left mode $\mathcal{N}(-5, 1)$. The score$_2$ is shown in Figure 3c, where the dashed true score is $\mathbb{P}_{\text{target}}(\mathbf{x} < 0) \approx 0.001$. We observe that the benchmarks have a relatively large score$_2$, which indicates they are very sensitive w.r.t. the target distribution. A negligible perturbation from the Gaussian target may lead to these methods focusing on a negligible mode. In contrast, the proposed PGPS is less sensitive with score$_2$ close to the desired value 0.001. Figure 3d corroborates the finding by visualizing the output distribution of the methods.

Compared to the gradient-flow-based benchmarks solely relying on the target distribution and its gradient, the proposed PGPS method follows a smooth LwS path instead, and is indeed less sensitive with better sampling quality.

5.1.3. Weight Recovery

We investigate the capability of the proposed PGPS method in estimating the corresponding weights besides detecting modes. The target distribution is a mixture of four 8-dimensional isometric Gaussian distributions $\{\mathcal{N}(\boldsymbol{\mu}_j, 0.15^2\mathbf{I}_8)\}$ and randomly generated weights; and the initial distribution is $\mathcal{N}(\mathbf{0}, \mathbf{I}_8)$.

For generated samples $\{\boldsymbol{x}_i\}_{i=1}^N$, define the estimated weight $\hat{\omega}_j := \frac{\sum_{i=1}^{N} \mathbb{I}(\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\| < 1)}{N}$. We evaluate the weight mismatch by $e := \sqrt{\sum_{j=1}^{4} (\hat{\omega}_j - \omega_j)^2}$, where $\omega_j := \mathbb{P}_{\text{target}}(\|\mathbf{x} - \boldsymbol{\mu}_j\|) < 1)$ is the ground truth. Smaller error e indicates more accurate weight estimation.

Table 1: Average Expected Calibration Error (ECE) and Accuracy (ACC) on UCI datasets over five independent runs

	Expected Calibration Error (ECE) ↓				Accuracy (ACC) ↑			
	PGPS	SVGD	SGLD	PFG	**PGPS**	SVGD	SGLD	PFG
SONAR	0.2517 ± 0.057	0.1712 ± 0.020	0.3394 ± 0.049	$\mathbf{0.1678} \pm 0.050$	$\mathbf{0.7981} \pm 0.023$	0.7962 ± 0.016	0.7942 ± 0.024	0.7673 ± 0.033
WINEWHITE	$\mathbf{0.0750} \pm 0.011$	0.0988 ± 0.012	0.0935 ± 0.024	0.0876 ± 0.018	0.4520 ± 0.010	0.4520 ± 0.010	$\mathbf{0.4831} \pm 0.049$	0.4520 ± 0.010
WINERED	$\mathbf{0.0366} \pm 0.005$	0.0402 ± 0.004	0.0868 ± 0.029	0.0449 ± 0.005	$\mathbf{0.5938} \pm 0.018$	0.5770 ± 0.018	0.5107 ± 0.096	0.5723 ± 0.019
AUSTRALIAN	0.1703 ± 0.066	0.1713 ± 0.064	0.3517 ± 0.078	$\mathbf{0.1457} \pm 0.047$	0.8620 ± 0.009	0.8626 ± 0.006	0.7362 ± 0.157	$\mathbf{0.8643} \pm 0.006$
HEART	$\mathbf{0.4579} \pm 0.071$	0.5117 ± 0.064	0.5110 ± 0.114	0.4887 ± 0.089	$\mathbf{0.2556} \pm 0.142$	0.1801 ± 0.042	0.2384 ± 0.135	0.1762 ± 0.033
GLASS	$\mathbf{0.1142} \pm 0.008$	0.1155 ± 0.006	0.2157 ± 0.025	0.1289 ± 0.021	$\mathbf{0.5850} \pm 0.080$	0.5383 ± 0.076	0.4561 ± 0.152	0.4505 ± 0.071
COVERTYPE	$\mathbf{0.0743} \pm 0.016$	0.0950 ± 0.012	0.1301 ± 0.038	0.0926 ± 0.078	$\mathbf{0.5899} \pm 0.095$	0.4867 ± 0.006	0.5221 ± 0.084	0.5088 ± 0.053

We take LD (a realization of the Wasserstein gradient flow) as a baseline, and denote its weight mismatch error by e_{LD}. In Figure 4, we demonstrate the distribution of the difference between the weight mismatch error e of a method and the baseline e_{LD} averaged over 10 independent experiments. While the baseline e_{LD} has an average value of 0.3314, the proposed PGPS methods consistently outperform LD with the distributions of $e - e_{LD}$ being significantly less than 0. The performance of PFG is similar to LD because of the same Wasserstein gradient flow nature, while SVGD performs worse than LD for this task. The inferior performance of SVGD is mainly due to the curse of dimensionality, which makes it difficult for the particles to escape from the trapping modes (Liu et al., 2022).

5.2. Bayesian Neural Network Inference

We further test PGPS methods for the Bayesian Neural Network (BNN) inference tasks. BNNs, which model the parameters of NNs as random variables to derive predictive posteriors for prediction, are usually considered to be difficult inference targets because of their non-concave likelihoods (Li et al., 2018). The proposed PGPS methods, with a stronger ability to discover the modes and recover their weights, achieve better inference performance.

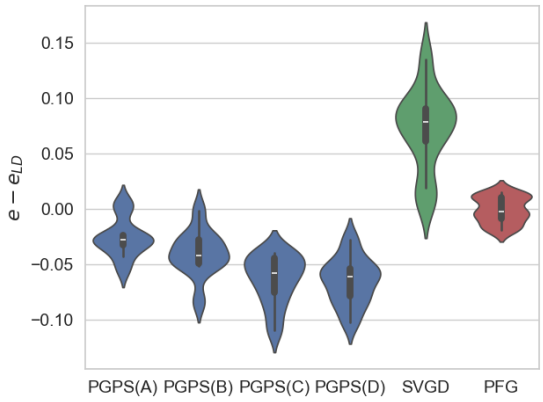


Figure 4: The weight mismatch error. The letter after PGPS indicates different hyperparameters. (A): $\alpha = 0$, $\beta = 1$, steps $= 0$ (B): $\alpha = 0$, $\beta = 0.5$, steps $= 0$ (C): $\alpha = 0$, $\beta = 1$, steps $= 100$ (D): $\alpha = 0$, $\beta = 0.5$, steps $= 100$, where 'steps' is the number the Langevin Adjustment steps.

Table 2: Average negative log-likelihood (NLL), ACC, and ECE on Noisy MNIST data over five independent runs

	PGPS	SVGD	SGLD	PFG
NLL ↓	1.8202 ± 0.019	1.8285 ± 0.040	$\mathbf{1.8184} \pm 0.127$	2.0171 ± 0.014
ACC ↑	$\mathbf{0.8788} \pm 0.017$	0.8282 ± 0.047	0.6419 ± 0.130	0.7119 ± 0.027
ECE ↓	$\mathbf{0.1716} \pm 0.012$	0.1941 ± 0.020	0.2183 ± 0.030	0.1752 ± 0.003

5.2.1. UCI DATASET

We conduct BNN inference for UCI datasets (Dua & Graff, 2017), where the neural network (NN) has one hidden layer with 32 hidden neurons and *Sigmoid* activation. More details of the experimental setup can be found in the Appendix E.4.

We report the averaged testing Expected Calibration Error (ECE) and testing accuracy (ACC) in Table 1, where ECE represents the calibration ability of the uncertain prediction by comparing the difference in prediction accuracy and prediction uncertainty for the test samples. The proposed PGPS methods achieve the best performance across most of the benchmark datasets with lower ECE and higher ACC, compared with SVGD, SGLD, and PFG baselines.

5.2.2. NOISY MNIST DATASET

Robustness is another desired property of learning Bayesian models. It is expected that Bayesian models would give more reasonable predictions with uncertainty quantification (UQ) when facing out-of-distribution data. We benchmark the prediction and UQ performance of the proposed PGPS methods for learning BNNs on the MNIST dataset (Deng, 2012).

To test the robustness of inferred models, we create perturbation by injecting additive Gaussian noise into the test MNIST images. Ensembles of 10 learned BNNs (i.e., 10 particles) are considered for evaluating competing inference methods. The performances are evaluated by negative log-likelihood (NLL), ACC, and ECE in Table 2. We can observe that the proposed PGPS method is again the best-performing inference method on all the metrics with the perturbed test data. SGLD is slightly better in NLL by 0.02 but with a large standard deviation of 0.127.

5.2.3. TRAINING-FREE PSPG

We compare the standard PGPS and the training-free PGPS as discussed in Section 3.2 using the same *Log-weighted*

Table 3: Averaged NLL, ACC, and ECE on Noisy MNIST data over five independent runs

# particles		PGPS	tf-PGPS
10	NLL ↓	**1.8171** $\pm$ 0.0168	1.8238 $\pm$ 0.0251
	ACC ↑	**0.8683** $\pm$ 0.0193	0.8380 $\pm$ 0.0504
	ECE ↓	0.1680 $\pm$ 0.0132	**0.1659** $\pm$ 0.0083
50	NLL	1.8473 $\pm$ 0.0101	**1.8467** $\pm$ 0.0108
	ACC	**0.9006** $\pm$ 0.0071	0.8672 $\pm$ 0.0298
	ECE	**0.1807** $\pm$ 0.0034	0.1890 $\pm$ 0.0071
100	NLL	**1.8763** $\pm$ 0.0087	1.9010 $\pm$ 0.0135
	ACC	**0.9182** $\pm$ 0.0036	0.8956 $\pm$ 0.0259
	ECE	**0.1959** $\pm$ 0.0024	0.1986 $\pm$ 0.0068

Shrinkage Path on the noisy MNIST data as in Section 5.2.2. The performance of standard PGPS and training-free PGPS (tf-PGPS) is reported in Table 3). We can observe that standard PGPS achieves better performance among almost all metrics and the number of particles than tf-PGPS. For the cases where tf-PGPS is better, their performances are almost indistinguishable. Interestingly, NLL increases as the number of particles goes up. We reason this by the fact that when using more particles for estimation, the predictions tend to fit the target posterior distribution better and lead to higher ACC but higher NLL as well.

6. Conclusion

In this paper, we proposed a novel path-guided particle-based sampling (PGPS) method and a *Log-weighted Shrinkage* path as a partition-function-free path that guides the particles moving from an initial distribution to the target distribution. We theoretically analyzed the performance of PGPS under the Wasserstein distance and characterized the impact of approximation error and discretization error on the quality of the generated samples. We conduct extensive experiments to test the PGPS methods in seeking the modes of the target distribution in sampling tasks, and the inference performance in terms of testing accuracy and calibration/uncertainty quantification in Bayesian learning tasks. The proposed PGPS methods perform consistently and considerably better than LD, SVGD, and PFG benchmarks in the experiments.

A limitation of the standard PGPS method is the requirement of training neural networks, similar to the PFG and other learning-required benchmarks. We propose training-free PGPS as an immediate solution, which is slightly worse than the training-based PGPS but more efficient.

A better density path design in PGPS that leverages the structure of the target distribution and analysis of training-free PGPS of its convergence to the target distribution are interesting future directions with both theoretical and practical importance.

Acknowledgements

R. Zhou would like to thank Chunyang Liao for the helpful discussion on Proposition 4.4. This work was supported in part by the U.S. National Science Foundation (NSF) grants SHF-2215573, IIS-2212419, and DMS-2312173; and by the U.S. Department of Engergy (DOE) Office of Science, Advanced Scientific Computing Research (ASCR) M2DT Mathematical Multifaceted Integrated Capability Center (MMICC) under Award B&R# KJ0401010/FWP# CC130, program manager W. Spotz. Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

Impact Statement

This paper presents a new Bayesian inference algorithm for efficient Bayesian learning, which would be critical in small-data scenarios that require uncertainty quantification. It can motivate advances in science, engineering, and biomedicine, for example in bioinformatics, materials science, and high-energy physics.

References

Albergo, M. and Vanden-Eijnden, E. Building normalizing flows with stochastic interpolants. In *ICLR 2023 Conference*, 2023.

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Alvarez-Melis, D., Schiff, Y., and Mroueh, Y. Optimizing functionals on the space of probabilities with input convex neural networks. *arXiv preprint arXiv:2106.00774*, 2021.

Ambrosio, L., Gigli, N., and Savaré, G. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2005.

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. In *International Conference on Machine Learning*, pp. 146–155. PMLR, 2017.

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. An introduction to mcmc for machine learning. *Machine learning*, 50:5–43, 2003.

Ba, J., Erdogdu, M. A., Ghassemi, M., Sun, S., Suzuki, T., Wu, D., and Zhang, T. Understanding the variance collapse of svgd in high dimensions. In *International Conference on Learning Representations*, 2021.

Benton, J., Deligiannidis, G., and Doucet, A. Error bounds for flow matching methods. *arXiv preprint arXiv:2305.16860*, 2023.

Chehab, O., Hyvarinen, A., and Risteski, A. Provable benefits of annealing for estimating normalizing constants: Importance sampling, noise-contrastive estimation, and beyond. *Advances in Neural Information Processing Systems*, 36, 2024.

Chen, P. and Ghattas, O. Projected stein variational gradient descent. *Advances in Neural Information Processing Systems*, 33:1947–1958, 2020.

Chwialkowski, K., Strathmann, H., and Gretton, A. A kernel test of goodness of fit. In *International conference on machine learning*, pp. 2606–2615. PMLR, 2016.

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

di Langosco, L. L., Fortuin, V., and Strathmann, H. Neural variational gradient descent. *arXiv preprint arXiv:2107.10731*, 2021.

Dong, H., Wang, X., Lin, Y., and Zhang, T. Particle-based variational inference with preconditioned functional gradient flow. *arXiv preprint arXiv:2211.13954*, 2022.

Doucet, A., De Freitas, N., and Gordon, N. An introduction to sequential monte carlo methods. *Sequential Monte Carlo methods in practice*, pp. 3–14, 2001.

Dua, D. and Graff, C. Uci machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Earl, D. J. and Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

Goan, E. and Fookes, C. Bayesian neural networks: An introduction and survey. *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018*, pp. 45–87, 2020.

Gong, W., Li, Y., and Hernández-Lobato, J. M. Sliced kernelized stein discrepancy. *arXiv preprint arXiv:2006.16531*, 2020.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., and Zemel, R. Learning the stein discrepancy for training and evaluating energy-based models without sampling. In *International Conference on Machine Learning*, pp. 3732–3747. PMLR, 2020.

Grosse, R. B., Maddison, C. J., and Salakhutdinov, R. R. Annealing between distributions by averaging moments. *Advances in Neural Information Processing Systems*, 26, 2013.

Heng, J., Bishop, A. N., Deligiannidis, G., and Doucet, A. Controlled sequential monte carlo. *The Annals of Statistics*, 48(5):2904–2929, 2020.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Hu, T., Chen, Z., Sun, H., Bai, J., Ye, M., and Cheng, G. Stein neural sampler. *arXiv preprint arXiv:1810.03545*, 2018.

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, 2021. doi: 10.1109/TPAMI.2020.2992934.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Liu, J. S. and Liu, J. S. *Monte Carlo strategies in scientific computing*, volume 10. Springer, 2001.

Liu, Q. Stein variational gradient descent as gradient flow. *Advances in neural information processing systems*, 30, 2017.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

Liu, X., Zhu, H., Ton, J.-F., Wynne, G., and Duncan, A. Grassmann stein variational gradient descent. *arXiv preprint arXiv:2202.03297*, 2022.

Maoutsa, D., Reich, S., and Opper, M. Interacting particle solutions of fokker–planck equations through gradient–log–density estimation. *Entropy*, 22(8):802, 2020.

Maurais, A. and Marzouk, Y. Sampling in unit time with kernel fisher-rao flow. *arXiv preprint arXiv:2401.03892*, 2024.

Mokrov, P., Korotin, A., Li, L., Genevay, A., Solomon, J. M., and Burnaev, E. Large-scale wasserstein gradient flows. *Advances in Neural Information Processing Systems*, 34: 15243–15256, 2021.

Murphy, K. P. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL probml.ai.

Neal, R. M. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

Treves, F. *Topological Vector Spaces, Distributions and Kernels: Pure and Applied Mathematics, Vol. 25*, volume 25. Elsevier, 2016.

Wang, L. and Nüsken, N. Measure transport with kernel mean embeddings. *arXiv preprint arXiv:2401.12967*, 2024.

Wang, Y., Chen, P., and Li, W. Projected wasserstein gradient descent for high-dimensional bayesian inference. *SIAM/ASA Journal on Uncertainty Quantification*, 10(4): 1513–1532, 2022.

A. Related Works

Wasserstein gradient flow aims at building gradient flow where the density follows the steepest descent path of some objective functional of density function under the Wasserstein metric. Sampling is fulfilled once the descent path converges to the target distribution, which is the minimizer of the objective function. A popular objective function of density is the KL-divergence between the density and the target distribution, and the flow is thus called KL Wasserstein gradient flow. While it is intractable in general to derive the KL Wasserstein gradient flow, i.e., the flow does not have a closed-form, Wang et al. (2022) resorted to Kernel Density Estimation (KDE) to estimate the gradient flow and uses Euler discretization to update the samples. However, it suffers from the curse of dimensionality, i.e. the kernel matrix would tend to be diagonal as dimensionality increases, due to the nature of kernels, which leads to inaccurate density estimation.

Aside from Euler discretization, the Jordan, Kinderlehrer, and Otto (JKO) scheme, aiming at finding a JKO operator that minimizes the target functional as well as the movement of the particles in each step, has been broadly applied to discretize the Wasserstein gradient flow. Alvarez-Melis et al. (2021) and Mokrov et al. (2021) applied a series of Input Convex Neural Networks (ICNN, Amos et al. (2017)) to model the gradient flow to ensure the convexity of the potential function in JKO scheme.

While the popularity of Stein Variational Gradient Descent (SVGD) (Liu & Wang, 2016) arises as a particle-based VI method, it can also be viewed as a specific type of gradient flow w.r.t. KL-divergence by determining the gradient $\boldsymbol{\phi}_t(\mathbf{x})$ that is the steepest descent direction under the kernelized Stein's Discrepancy (Chwialkowski et al., 2016) by the reproducing kernel Hilbert space (RKHS) (Liu, 2017). However, the curse of dimensionality for the kernel-based methods leads to the particle collapse in SVGD (Ba et al., 2021), i.e., variance collapse. Currently, there are two major methods to tackle the curse of dimensionality. Projecting the inference space to a lower dimension can naturally avoid high-dimensional VI. While Chen & Ghattas (2020) projected the dynamics into a lower dimensional subspace and theoretically proved the asymptotically converging performance of the projected SVGD, Gong et al. (2020) proposed sliced kernel Stein discrepancy that projects the particle dynamics into a single dimensional subspace. More recently, Liu et al. (2022) proposed Grassmann SVGD that also considers a low-dimensional projection and is claimed to be more efficient than projected SVGD (Chen & Ghattas, 2020) without the need for costly eigenvector decomposition. Another type of popular method leverages the Universal Approximation Theorem of Neural Networks (NNs) (Hornik et al., 1989) and defines more general discrepancy. di Langosco et al. (2021) proposed to minimize Stein's discrepancy based on the NNs, instead of functions drawn from RKHS like SVGD. Grathwohl et al. (2020) proposed to learn a single energy function based on Stein's Discrepancy for energy-based models, while Hu et al. (2018) tried to learn a transport plan based on Stein's Discrepancy or more general f-divergence. Dong et al. (2022) modified the regularization term of the loss function to a preconditioned version but it needs to calculate the Jacobian of the target density and in turn time-consuming.

B. Implementation of LwS Path

Though it is possible to fully depend on the AutoGrad functionality of the machine learning packages, a relatively closed form of the gradient and derivatives of our *Log-Shrinkage Path*, $\ln \hat{p}_t^{\mathrm{LwS}}(\boldsymbol{x}) = (1-t)\ln p_0\left((1-\alpha t)\boldsymbol{x}\right) + t\ln \hat{p}_1\left(\frac{\boldsymbol{x}}{\beta+(1-\beta)t}\right)$, would lead to better calculation quality and faster computational speed.

Denote $\boldsymbol{x}_a = (1-\alpha t)\boldsymbol{x}$, $\boldsymbol{x}_b = \frac{\boldsymbol{x}}{\beta+(1-\beta)t}$. The gradient of our path $\hat{p}_t^{\mathrm{LwS}}(\boldsymbol{x})$ at time t would be

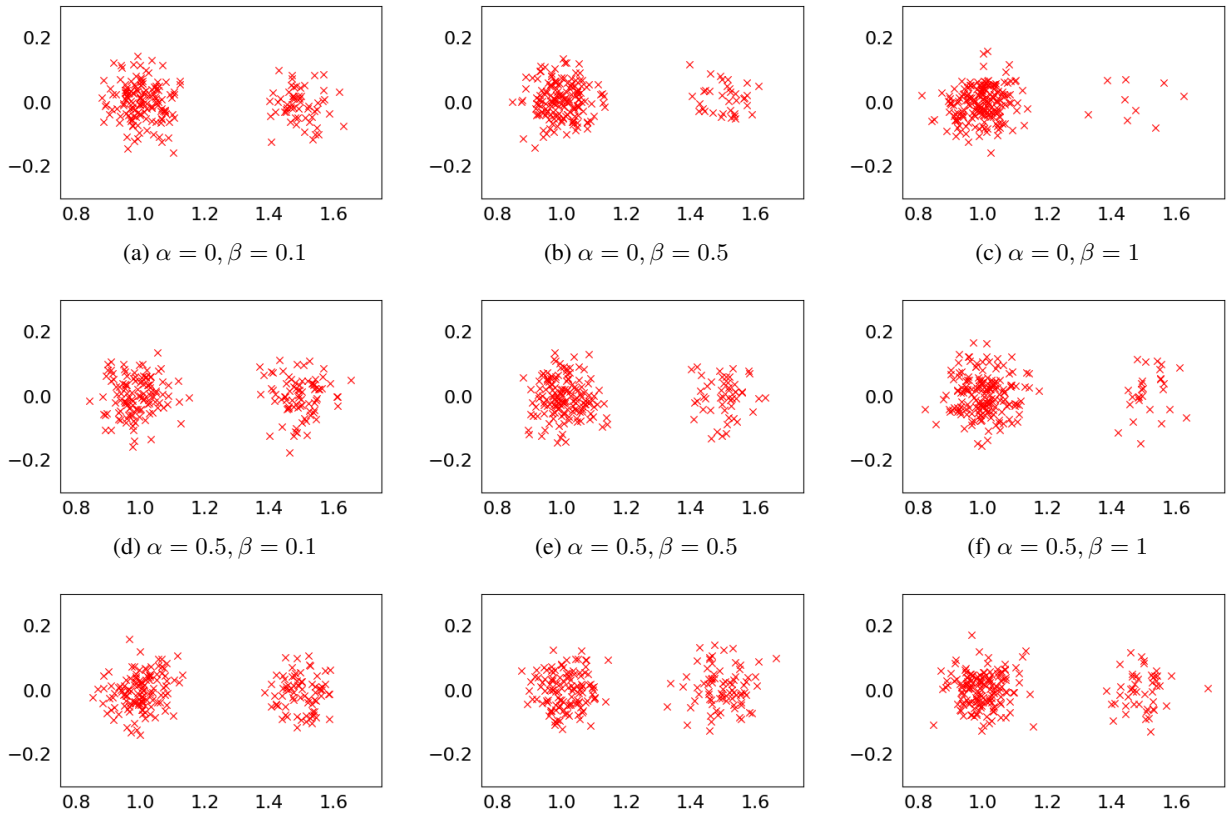
$$\nabla \ln \hat{p}_t^{\mathrm{LwS}}(\boldsymbol{x}) = (1-t)(1-\alpha t)\nabla \ln p_0(\boldsymbol{x}_a) + \frac{t}{\beta+(1-\beta)t}\nabla \ln \hat{p}_1(\boldsymbol{x}_b), \tag{10}$$

and the derivative would be

$$\frac{\mathrm{d}}{\mathrm{d}t}\ln \hat{p}_t^{\mathrm{LwS}}(\boldsymbol{x}) = -\ln p_0(\boldsymbol{x}_a) + \ln \hat{p}_1(\boldsymbol{x}_b) - \alpha(1-t)\boldsymbol{x}\cdot\nabla \ln p_0(\boldsymbol{x}_a) - \frac{(1-\beta)t\boldsymbol{x}\cdot\nabla \ln \hat{p}_1(\boldsymbol{x}_b)}{(\beta+(1-\beta)t)^2}, \tag{11}$$

where $(\cdot)$ denotes inner product.

In our training target of Equation (6), one critical part is the divergence $\nabla\cdot\boldsymbol{\phi}^\theta$ of the approximated vector field $\boldsymbol{\phi}^\theta$. While (Dong et al., 2022) proposed to use an efficient computational estimation derived by the integration-by-parts technique, a close form of divergence can be derived for relatively simple NN implementation. Specifically, for the one hidden layer, H hidden neuron, D dimensional input, *sigmoid* activation MLP we used in this work, $\boldsymbol{\phi}^\theta = W_2\sigma(W_1\boldsymbol{x}+b_1)+b_2$, the close



(a) $\alpha = 0, \beta = 0.1$ (b) $\alpha = 0, \beta = 0.5$ (c) $\alpha = 0, \beta = 1$

(d) $\alpha = 0.5, \beta = 0.1$ (e) $\alpha = 0.5, \beta = 0.5$ (f) $\alpha = 0.5, \beta = 1$

(g) $\alpha = 1, \beta = 0.1$ (h) $\alpha = 1, \beta = 0.5$ (i) $\alpha = 1, \beta = 1$

Figure 5: Particles for tf-PGPS following the LwS-path with different hyperparameters discretized with a constant time step of 0.01, 30 LD steps for each intermediate distribution. With the same computational demand, the hyperparameter choices can influence the sample quality. The no-shrinkage setup $(\alpha = 0, \beta = 1)$ leads to the worst performance and the hyperparameter choices that incorporate shrinkage capture much better the mode on the right.

form of divergence would be

$$\nabla_{\boldsymbol{x}} \cdot \boldsymbol{\phi}^{\theta}(\boldsymbol{x}) = \sum_{i=1}^{D} (\nabla_{\boldsymbol{x}} \boldsymbol{\phi}^{\theta}_{(i)}(\boldsymbol{x}))_i = trace(W_2 \text{diag}(\boldsymbol{x}_g) W_1) = \sum W_2 \otimes W_1^T \otimes (\mathbf{1}_D \boldsymbol{x}_g^T), \tag{12}$$

where σ is the *sigmoid* function, $\boldsymbol{x}_h = \sigma(W_1 \boldsymbol{x} + b_1)$ is the output of the first layer, $\otimes$ denotes entry-wise product, $\boldsymbol{x}_g = \boldsymbol{x}_h \otimes (\mathbf{1}_H - \boldsymbol{x}_h)$, $\mathbf{1}_D$ and $\mathbf{1}_H$ denotes all one matrix with size of $D \times 1$ and $H \times 1$, respectively, and the "$\sum$" sign in the last equation denotes summation along both dimensions.

C. Influence of Hyperparameter in LwS Path

To show the impact of hyperparameter choices, we here give an example with the same target as the motivating example in Figure 1(a), a mixture of two Gaussian distributions with equal weights. We apply the training-free version of PGPS discretized with a constant time step of 0.01, i.e. 99 intermediate distributions, and 30 LD steps performed for each intermediate distribution to ensure that different setups consume the same computational resource.

With the results illustrated in Figure 5, it can be observed that the hyperparameter choices can influence the sample quality with the same computational demand. The no-shrinkage setup leads to the worst performance and the hyperparameter choices that incorporate shrinkage capture the mode to the right much better.

D. Proofs

D.1. Proof of Proposition 3.1

Proposition D.1. *For a given partition-free density path $\{\hat{p}_t\}$, the gradient flow guided by the vector field $\boldsymbol{\phi}_t(\mathbf{x})$ following the Fokker-Planck equation (1) satisfies:*

$$r(\boldsymbol{x}, \boldsymbol{\phi}_t) - \mathbb{E}_{\mathbf{x}\sim p_t}\left[\frac{\partial \ln \hat{p}_t(\mathbf{x})}{\partial t}\right] = 0, \tag{13}$$

where $r(\boldsymbol{x}, \boldsymbol{\phi}_t) = \frac{\partial \ln \hat{p}_t(\boldsymbol{x})}{\partial t} + (\nabla \ln \hat{p}_t(\boldsymbol{x}) + \nabla) \cdot \boldsymbol{\phi}_t(\boldsymbol{x})$.

Proof. We start with the Fokker-Planck equation:

$$\frac{\partial}{\partial t} p_t(\boldsymbol{x}) = -\nabla \cdot (p_t(\boldsymbol{x})\boldsymbol{\phi}_t(\boldsymbol{x})), \tag{14}$$

The right-hand side (RHS) of (14) can be derived to be:

$$-\nabla \cdot (p_t(\boldsymbol{x})\boldsymbol{\phi}_t(\boldsymbol{x})) = -p_t(\boldsymbol{x})[\nabla \ln p_t(\boldsymbol{x}) \cdot \boldsymbol{\phi}_t(\boldsymbol{x}) + \nabla \cdot \boldsymbol{\phi}_t(\boldsymbol{x})]. \tag{15}$$

Though it is usually non-trivial to find the derivative of the path with respect to time, it is clear that $\nabla \ln p_t = \nabla \ln \hat{p}_t + \nabla \ln \int \hat{p}_t \, \mathrm{d}\boldsymbol{x} = \nabla \ln \hat{p}_t$. Equation (15) can then be further transformed into:

$$-\nabla \cdot (p_t(\boldsymbol{x})\boldsymbol{\phi}_t(\boldsymbol{x})) = -p_t(\boldsymbol{x})[\nabla \ln \hat{p}_t(\boldsymbol{x}) \cdot \boldsymbol{\phi}_t(\boldsymbol{x}) + \nabla \cdot \boldsymbol{\phi}_t(\boldsymbol{x})]. \tag{16}$$

On the other hand, the left-hand side (LHS) of (14) can be derived as:

$$\begin{aligned}
&\frac{\partial}{\partial t} p_t(\boldsymbol{x}) = \frac{\partial}{\partial t} \frac{\hat{p}_t(\boldsymbol{x})}{\int \hat{p}_t(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}} \\
=&\frac{\int \hat{p}_t(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \frac{\partial}{\partial t}\hat{p}_t(\boldsymbol{x}) - \hat{p}_t(\boldsymbol{x}) \frac{\partial}{\partial t} \int \hat{p}_t(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}}{(\int \hat{p}_t(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x})^2} \\
=&\frac{\frac{\partial}{\partial t}\hat{p}_t(\boldsymbol{x})}{\int \hat{p}_t(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}} - \frac{\hat{p}_t(\boldsymbol{x})}{\int \hat{p}_t(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}} \int \frac{\frac{\partial}{\partial t}\hat{p}_t(\boldsymbol{x})}{\int \hat{p}_t(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}} \, \mathrm{d}\boldsymbol{x} \\
=&\frac{\hat{p}_t(\boldsymbol{x})}{\int \hat{p}_t(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}} \frac{\partial}{\partial t} \ln \hat{p}_t(\boldsymbol{x}) - p_t(\boldsymbol{x}) \int \frac{\hat{p}_t(\boldsymbol{x})}{\int \hat{p}_t(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}} \frac{\partial}{\partial t} \ln \hat{p}_t(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\
=&p_t(\boldsymbol{x})(\frac{\partial \ln \hat{p}_t(\boldsymbol{x})}{\partial t} - \int p_t(\boldsymbol{x}) \frac{\partial \ln \hat{p}_t(\boldsymbol{x})}{\partial t} \, \mathrm{d}\boldsymbol{x}) \\
=&p_t(\boldsymbol{x}) \left(\frac{\partial \ln \hat{p}_t(\boldsymbol{x})}{\partial t} - \mathbb{E}_{\mathbf{x}\sim p_t}\left[\frac{\partial \ln \hat{p}_t(\mathbf{x})}{\partial t}\right]\right).
\end{aligned} \tag{17}$$

Substituting equations (16) and (17) to Equation (14) and $p_t(\boldsymbol{x})$ on both sides and we have the desired result

$$\nabla \ln \hat{p}_t(\boldsymbol{x}) \cdot \boldsymbol{\phi}_t(\boldsymbol{x}) + \nabla \cdot \boldsymbol{\phi}_t(\boldsymbol{x}) = \frac{\partial \ln \hat{p}_t(\boldsymbol{x})}{\partial t} - \mathbb{E}_{\mathbf{x}\sim p_t}\left[\frac{\partial \ln \hat{p}_t(\mathbf{x})}{\partial t}\right]. \tag{18}$$

□

D.2. Proof of Theorem 4.2

Before showing the quality analysis of the evolved particles with the discretized algorithm, we first evaluate the impact of error between the numerical approximation $\boldsymbol{\phi}_t^\theta(\boldsymbol{x})$ and the true $\boldsymbol{\phi}_t(\boldsymbol{x})$ satisfying Equation (4).

Lemma D.2 (Proposition 3 of Albergo & Vanden-Eijnden (2023))**.** *For two flows $\boldsymbol{\phi}_t^\theta(\boldsymbol{x})$ and $\boldsymbol{\phi}_t(\boldsymbol{x})$ under Assumption 4.1, the Wasserstein distance between the distribution $p_{\mathbf{x}_1^\theta}$ of random variable $\mathbf{x}_1^\theta = \mathbf{x}_0 + \int_0^1 \boldsymbol{\phi}_t^\theta(\mathbf{x}_t) \, \mathrm{d}t$, and the distribution $p_{\mathbf{x}_1}$ of $\mathbf{x}_1 = \mathbf{x}_0 + \int_0^1 \boldsymbol{\phi}_t(\mathbf{x}_t) \, \mathrm{d}t$ is bounded:*

$$W^2(p_{\mathbf{x}_1^\theta}, p_{\mathbf{x}_1}) \leq \delta^2 \exp(1 + 2K_1). \tag{19}$$

Due to the necessary discretization involved in particle evolution, another error factor arises. We here give the analysis of the discretization error.

Lemma D.3. *For a trained flow $\boldsymbol{\phi}_t^\theta(\boldsymbol{x})$ under Assumption 4.1, the Wasserstein distance between the distribution $p_{\mathbf{x}_1}$ of random variable $\mathbf{x}_1 = \mathbf{x}_0 + \int_0^1 \boldsymbol{\phi}_t^\theta(\mathbf{x}_t)\,\mathrm{d}t$ and the distribution $p_{\hat{\mathbf{x}}_1}$ of random variable $\hat{\mathbf{x}}_1$ generated by constant discretization $\hat{\mathbf{x}}_{(n+1)h} = \hat{\mathbf{x}}_{nh} + \boldsymbol{\phi}_{nh}^\theta(\hat{\mathbf{x}}_{nh})$ with step-size h is bounded:*

$$W^2(p_{\mathbf{x}_1}, p_{\hat{\mathbf{x}}_1}) \leq h\frac{C(\exp\big(1+K_1^2\big)-1)}{1+K_1^2}, \tag{20}$$

where $C = \frac{1}{2}K_2^2 + \frac{17}{2}K_1^2K_3^2 + 5K_1K_2K_3$ is a constant.

Proof. Here we consider the discretization error

$$\begin{aligned}
&W^2(p_{\mathbf{x}_{t+h}}, p_{\hat{\mathbf{x}}_{t+h}}) && (21)\\
\leq&\mathbb{E}_\gamma\|\mathbf{x}_{t+h} - \hat{\mathbf{x}}_{t+h}\|^2 && (22)\\
=&\mathbb{E}_\gamma\|\mathbf{x}_t + (\mathbf{x}_{t+h} - \mathbf{x}_t) - [\hat{\mathbf{x}}_t + h\boldsymbol{\phi}_t(\hat{\mathbf{x}}_t)]\|^2 && (23)\\
\leq&\mathbb{E}_\gamma\|[\mathbf{x}_t - \hat{\mathbf{x}}_t] + [\mathbf{x}_{t+h} - \mathbf{x}_t - h\boldsymbol{\phi}_t(\hat{\mathbf{x}}_t)]\|^2 && (24)\\
\leq&(1+\lambda)\mathbb{E}_\gamma\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|^2 + (1+\frac{1}{\lambda})\mathbb{E}_\gamma\|\int_t^{t+h}\boldsymbol{\phi}(t',\mathbf{x}_{t'})dt' - h\boldsymbol{\phi}_t(\hat{\mathbf{x}}_t)\|^2 && (25)
\end{aligned}$$

Now we bound the second term:

$$\begin{aligned}
&\mathbb{E}_\gamma\|\int_t^{t+h}\boldsymbol{\phi}(t',\mathbf{x}_{t'})dt' - h\boldsymbol{\phi}_t(\hat{\mathbf{x}}_t)\|^2 && (26)\\
\leq&\mathbb{E}_\gamma(\int_t^{t+h}\|\boldsymbol{\phi}(t',\mathbf{x}_{t'}) - \boldsymbol{\phi}_t(\hat{\mathbf{x}}_t)\|dt')^2 \quad \text{(Jensen's)} && (27)\\
\leq&\mathbb{E}_\gamma(\int_t^{t+h}\|\boldsymbol{\phi}(t',\mathbf{x}_{t'}) - \boldsymbol{\phi}_t(\mathbf{x}_{t'}) + \boldsymbol{\phi}_t(\mathbf{x}_{t'}) - \boldsymbol{\phi}_t(\hat{\mathbf{x}}_t)\|dt')^2 && (28)\\
\leq&\mathbb{E}_\gamma(\int_t^{t+h}\|\boldsymbol{\phi}(t',\mathbf{x}_{t'}) - \boldsymbol{\phi}_t(\mathbf{x}_{t'})\| + \|\boldsymbol{\phi}_t(\mathbf{x}_{t'}) - \boldsymbol{\phi}_t(\hat{\mathbf{x}}_t)\|dt')^2\text{(Triangular)} && (29)\\
\leq&\mathbb{E}_\gamma(\int_t^{t+h}K_2(t'-t) + K_1\|\mathbf{x}_{t'} - \hat{\mathbf{x}}_t\|dt')^2 && (30)\\
=&\mathbb{E}_\gamma(\frac{K_2h^2}{2} + \int_t^{t+h}K_1\|\mathbf{x}_{t'} - \hat{\mathbf{x}}_t\|dt')^2 && (31)\\
\leq&\mathbb{E}_\gamma(\frac{K_2h^2}{2} + \int_t^{t+h}K_1\|\mathbf{x}_{t'} - \mathbf{x}_t\| + K_1\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|dt')^2 \qquad \text{(Triangular)} && (32)\\
=&\mathbb{E}_\gamma(\frac{K_2h^2}{2} + hK_1\|\mathbf{x}_t - \hat{\mathbf{x}}_t\| + \int_t^{t+h}K_1\|\mathbf{x}_{t'} - \mathbf{x}_t\|dt')^2 && (33)\\
\leq&\mathbb{E}_\gamma(\frac{K_2h^2}{2} + hK_1\|\mathbf{x}_t - \hat{\mathbf{x}}_t\| + \int_t^{t+h}K_1K_3(t'-t)dt')^2 && (34)\\
=&\mathbb{E}_\gamma(\frac{K_2h^2}{2} + hK_1\|\mathbf{x}_t - \hat{\mathbf{x}}_t\| + \frac{K_1K_3}{2}h^2)^2. && (35)
\end{aligned}$$

Substituting (35) to (25),

$$\begin{aligned}
&W^2(p_{\mathbf{x}_{t+h}}, p_{\hat{\mathbf{x}}_{t+h}}) && (36)\\
\leq&(1+\lambda)\mathbb{E}_\gamma\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|^2 + (1+\frac{1}{\lambda})\mathbb{E}_\gamma(C_1h^2 + hK_1\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|)^2 && (37)\\
=&(1+\lambda)\mathbb{E}_\gamma\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|^2
\end{aligned}$$

$$+(1+\frac{1}{\lambda})\mathbb{E}_\gamma(C_1^2h^4+h^2K_1^2\|\mathbf{x}_t-\hat{\mathbf{x}}_t\|^2+2C_1h^3K_1\|\mathbf{x}_t-\hat{\mathbf{x}}_t\|) \tag{38}$$

$$=(1+\lambda)\mathbb{E}_\gamma\|\mathbf{x}_t-\hat{\mathbf{x}}_t\|^2 + (1+\frac{1}{\lambda})(C_1^2h^4+h^2K_1^2\mathbb{E}_\gamma\|\mathbf{x}_t-\hat{\mathbf{x}}_t\|^2+2C_1h^3K_1\mathbb{E}_\gamma\|\mathbf{x}_t-\hat{\mathbf{x}}_t\|) \tag{39}$$

$$\leq(1+\lambda)\mathbb{E}_\gamma\|\mathbf{x}_t-\hat{\mathbf{x}}_t\|^2 + (1+\frac{1}{\lambda})(C_1^2h^4+h^2K_1^2\mathbb{E}_\gamma\|\mathbf{x}_t-\hat{\mathbf{x}}_t\|^2+2C_1h^3K_1\sqrt{\mathbb{E}_\gamma\|\mathbf{x}_t-\hat{\mathbf{x}}_t\|^2}), \tag{40}$$

where $C_1=\frac{K_2}{2}+\frac{K_1K_3}{2}$. Choosing γ that minimizes $\mathbb{E}_\gamma\|\mathbf{x}_t-\hat{\mathbf{x}}_t\|^2$,

$$W^2(p_{\mathbf{x}_{t+h}},p_{\hat{\mathbf{x}}_{t+h}}) \tag{41}$$

$$\leq(1+\lambda)W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+(1+\frac{1}{\lambda})(C_1^2h^4+h^2K_1^2W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+2C_1h^3K_1W(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})). \tag{42}$$

Choosing $\lambda=h$,

$$W^2(p_{\mathbf{x}_{t+h}},p_{\hat{\mathbf{x}}_{t+h}}) \tag{43}$$

$$\leq(1+h)W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+(1+\frac{1}{h})(C_1^2h^4+h^2K_1^2W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+2C_1h^3K_1W(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})) \tag{44}$$

$$\leq(1+h)W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+(1+\frac{1}{h})(C_1^2h^4+h^2K_1^2W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+4C_1h^3K_1K_3) \tag{45}$$

$$\leq(1+h)W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+hK_1^2W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+Ch^2 \tag{46}$$

$$\leq[1+(1+K_1^2)h]W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+Ch^2, \tag{47}$$

where $C=2C_1^2+4K_1^2K_3^2+8C_1K_1K_3$ is a constant. The fact that $h^4<h^3<h^2$ for $0<h<1$ and $W(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})$ can be bounded by $2K_3$ leads to (47). $W(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})$ is bounded because $W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})\leq\mathbb{E}\|\mathbf{x}_1-\mathbf{x}_0-(\hat{\mathbf{x}}_1-\mathbf{x}_0)\|^2\leq 2\mathbb{E}\|\mathbf{x}_1-\mathbf{x}_0\|^2+2\mathbb{E}\|\hat{\mathbf{x}}_1-\mathbf{x}_0\|^2\leq 4K_3^2$.

We therefore have

$$W^2(p_{\mathbf{x}_{t+h}},p_{\hat{\mathbf{x}}_{t+h}})\leq[1+(1+K_1^2)h]W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+Ch^2, \tag{48}$$

and

$$W^2(p_{\mathbf{x}_{t+h}},p_{\hat{\mathbf{x}}_{t+h}})+\frac{Ch}{1+K_1^2}\leq[1+(1+K_1^2)h][W^2(p_{\mathbf{x}_t},p_{\hat{\mathbf{x}}_t})+\frac{Ch}{1+K_1^2}]. \tag{49}$$

It can then be shown that

$$W^2(p_{\mathbf{x}_{nh}},p_{\hat{\mathbf{x}}_{nh}})+\frac{Ch}{1+K_1^2}\leq[1+(1+K_1^2)h]^n[W^2(p_{\mathbf{x}_0},p_{\hat{\mathbf{x}}_0})+\frac{Ch}{1+K_1^2}]=\frac{[1+(1+K_1^2)h]^nCh}{1+K_1^2}. \tag{50}$$

Hence,

$$W^2(p_{\mathbf{x}_1},p_{\hat{\mathbf{x}}_1})\leq h\frac{C}{1+K_1^2}[(1+(1+K_1^2)h)^{1/h})-1] \tag{51}$$

$$\leq h\frac{C}{1+K_1^2}(\exp\big(1+K_1^2\big)-1). \tag{52}$$

□

Combining Lemma D.2 and Lemma D.3, we have our main result:

Theorem D.4. *For two flows $\phi_t^\theta(\mathbf{x})$ and $\phi_t(\mathbf{x})$ under Assumption 4.1, the Wasserstein distance between the distribution $p_{\hat{\mathbf{x}}_1^\theta}$ of random variable $\hat{\mathbf{x}}_1^\theta$ generated by discretization $\hat{\mathbf{x}}_{(n+1)h}=\hat{\mathbf{x}}_{nh}+\phi_{nh}^\theta(\hat{\mathbf{x}}_{nh})$ with step-size h, and the distribution $p_{\mathbf{x}_1}$ of $\mathbf{x}_1=\mathbf{x}_0+\int_0^1\phi_t^\theta(\mathbf{x}_t)\,\mathrm{d}t$ is bounded:*

$$W(p_{\hat{\mathbf{x}}_1^\theta},p_{\mathbf{x}_1})\leq\delta\sqrt{\exp(1+2K_1)}+\sqrt{h}\sqrt{\frac{C(\exp(1+K_1^2)-1)}{1+K_1^2}}, \tag{53}$$

where $C=\frac{1}{2}K_2^2+\frac{17}{2}K_1^2K_3^2+5K_1K_2K_3$.

Proof. Because Wasserstein distance is a metric,

$$W(p_{\hat{\mathbf{x}}_1^\theta}, p_{\mathbf{x}_1}) \leq W(p_{\hat{\mathbf{x}}_1^\theta}, p_{\mathbf{x}_1^\theta}) + W(p_{\mathbf{x}_1^\theta}, p_{\mathbf{x}_1}), \tag{54}$$

where $p_{\mathbf{x}_1^\theta}$ is the distribution of $\mathbf{x}_1^\theta$, the random variable following the gradient flow $\boldsymbol{\phi}_t^\theta(x)$. With the first term bounded by Theorem D.3 and the second term bounded by Theorem D.2, we complete the proof. □

D.3. Proof of Proposition 4.4

For a given path $\hat{p}_t(\boldsymbol{x})$, we are essentially solving the equation (4) restated below

$$\frac{\partial \ln \hat{p}_t(\boldsymbol{x})}{\partial t} + (\nabla \ln \hat{p}_t(\boldsymbol{x}) + \nabla) \cdot \boldsymbol{\phi}_t(\boldsymbol{x}) - \mathbb{E}_{\mathbf{x}\sim p_t}\left[\frac{\partial \ln \hat{p}_t(\mathbf{x})}{\partial t}\right] = 0 \tag{55}$$

via minimizing a quadratic loss function. Note that the solution $\boldsymbol{\phi}_t$ for the equation above may not be unique. We will next show minimizing the quadratic loss function is consistent with solving the equation. Having an infinite number of samples implies that we are studying the behavior in expectation. Note that the precise impact of finite samples is related to the issue of generalization error, which is beyond the scope of this work. Given infinite number of particles following distribution p_t, the loss function in Equation (6) can be written as

$$L_t(\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_t}\left[\left(\frac{\partial \ln \hat{p}_t(\boldsymbol{x})}{\partial t} + (\nabla \ln \hat{p}_t(\boldsymbol{x}) + \nabla) \cdot \boldsymbol{\phi}_t^\theta(\boldsymbol{x}) - \mathbb{E}_{\mathbf{x}\sim p_t}\left[\frac{\partial \ln \hat{p}_t(\mathbf{x})}{\partial t}\right]\right)^2\right] \tag{56}$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p_t}\left[\left((\nabla \ln \hat{p}_t(\boldsymbol{x}) + \nabla) \cdot (\boldsymbol{\phi}_t^\theta(\boldsymbol{x}) - \boldsymbol{\phi}_t(\boldsymbol{x}))\right)^2\right], \tag{57}$$

where the first relation is by definition and the second relation is by $(\nabla \ln \hat{p}_t(\boldsymbol{x}) + \nabla) \cdot \boldsymbol{\phi}_t(\boldsymbol{x}) = \mathbb{E}_{\mathbf{x}\sim p_t}\left[\frac{\partial \ln \hat{p}_t(\mathbf{x})}{\partial t}\right] - \frac{\partial \ln \hat{p}_t(\boldsymbol{x})}{\partial t}$.

Discussion of the function space The discussion is the same for any $t \in [0, 1]$ and we omit subscript t in the following. We first need to specify the vector field class (the function space to solve the PDE) that the optimization is performed in. Define the operator $(\mathcal{T}\boldsymbol{\psi})(\boldsymbol{x}) := (\nabla \ln p(x) + \nabla) \cdot \boldsymbol{\psi}(\boldsymbol{x})$, where $\boldsymbol{\psi}$ is a differentiable vector field. Let $L^2(\mu) \subset \{\mathbb{R}^d \to \mathbb{R}\}$ be a weighted L^2 space with measure $d\mu(x) = p(x)dx$. It is required that $\mathcal{T}(\boldsymbol{\phi}^\theta - \boldsymbol{\phi}) \in L^2(\mu)$ so that the loss function exists. This condition is satisfied by choosing the vector field such that $\nabla \ln p(x)$, $\boldsymbol{\phi}^\theta$ and $\boldsymbol{\phi}$ all lie in $\Psi := [W^{1,4}(\mu)]^d$, which is the product space of the weighted Sobelov space $W^{1,4}(\mu) = \{f \in L^4(p_t) : \frac{\partial}{\partial x_i} f(\boldsymbol{x}) \in L^4(p_t)\} \subset L^4(\mu)$.

The following proposition established the desired consistency.

Proposition D.5 (Restatement of Proposition 4.4)**.** *Under Assumption 4.3, for any $\boldsymbol{\phi}_t^\theta$ there exists a vector-field $\boldsymbol{\phi}_t$ solution to PDE (4) that*

$$\mathbb{E}_{\mathbf{x}\sim p_t}[\|\boldsymbol{\phi}_t^\theta(\mathbf{x}) - \boldsymbol{\phi}_t(\mathbf{x})\|^2] \leq K\mathbb{E}_{\mathbf{x}\sim p_t}\left[\left(\frac{\partial \ln \hat{p}_t(\mathbf{x})}{\partial t} + (\nabla \ln \hat{p}_t(\mathbf{x}) + \nabla) \cdot \boldsymbol{\phi}_t^\theta(\mathbf{x}) - \mathbb{E}_{\mathbf{x}\sim p_t}\left[\frac{\partial \ln \hat{p}_t(\mathbf{x})}{\partial t}\right]\right)^2\right] \tag{58}$$

where $K > 0$ is a universal constant factor.

Proof. We omit the subscript t in p_t, $\boldsymbol{\phi}_t$, $\boldsymbol{\phi}_t^\theta$ for simplicity, and we let μ be the measure associated with p_t as $\mathrm{d}\mu(\boldsymbol{x}) = p(\boldsymbol{x})\mathrm{d}t$. The function class $L^2(\mu)$, $L^4(\mu)$ and $W^{1,4}(\mu)$ is defined accordingly.

Note that $(\Psi, \|\cdot\|_\Psi)$ is a Banach space with

$$\|\boldsymbol{\psi}\|_\Psi^2 := \sum_j \int \boldsymbol{\psi}_j(\boldsymbol{x})^2 d\mu(\boldsymbol{x}) + \sum_{i,j} \int [\frac{\partial}{\partial x_i}\boldsymbol{\psi}_j(\boldsymbol{x})]^2 d\mu(\boldsymbol{x}). \tag{59}$$

$L^2(\mu)$ is a naturally a Banach space with $\|g\|_\mu^2 = \int g(\boldsymbol{x})^2 d\mu(\boldsymbol{x})$ for any $g \in L^2(\mu)$.

We next show that operator $\mathcal{T} : \Psi \to L^2(\mu)$ is

$$(\mathcal{T}\boldsymbol{\psi})(\boldsymbol{x}) := (\nabla \ln p(\boldsymbol{x}) + \nabla) \cdot \boldsymbol{\psi}(\boldsymbol{x}), \quad \forall \boldsymbol{\psi} \in \Psi, \tag{60}$$

is a bounded linear operator. The linearity is straightforward. The boundedness is because that for any $\boldsymbol{\psi} \in \Psi$ with $\|\boldsymbol{\psi}\|_\Psi < \infty$,

$$\|\mathcal{T}\boldsymbol{\psi}\|_\mu \leq \|\nabla \ln p \cdot \boldsymbol{\psi}\|_\mu + \|\nabla \cdot \boldsymbol{\psi}\|_\mu < \infty, \tag{61}$$

where the first inequality is by triangle inequality and the second is by the fact that

$$\|\nabla \ln p \cdot \boldsymbol{\psi}\|_\mu^2 = \int (\nabla \ln p(\boldsymbol{x}) \cdot \boldsymbol{\psi}(\boldsymbol{x}))^2 d\mu(\boldsymbol{x}) \tag{62}$$

$$\leq \int \|\nabla \ln p(\boldsymbol{x})\|^2 \|\boldsymbol{\psi}(\boldsymbol{x})\|^2 d\mu(\boldsymbol{x}) \tag{63}$$

$$\leq \sqrt{\int \|\nabla \ln p(\boldsymbol{x})\|^4 d\mu(\boldsymbol{x})} \sqrt{\int \|\boldsymbol{\psi}(\boldsymbol{x})\|^4 d\mu(\boldsymbol{x})} < \infty, \tag{64}$$

and the fact that $\nabla \ln p, \boldsymbol{\psi} \in \Psi = [W^{1,4}(\mu)]^d$.

Denote by $G = \{\mathcal{T}\boldsymbol{\psi} : \boldsymbol{\psi} \in \Psi\}$ the range of the linear operator $\mathcal{T}$ and let $N^{\mathcal{T}} = \{\boldsymbol{\psi} \in \Psi : (\mathcal{T}\boldsymbol{\psi})(x) = 0, \forall x\}$ be the null space of $\mathcal{T}$. It follows that $\mathcal{T} : \Psi/N^{\mathcal{T}} \to G$ is a bijection, where $\Psi/N^{\mathcal{T}}$ is the quotient space. To see this bijection, observe that $\mathcal{T}\boldsymbol{\psi} \neq \mathcal{T}\boldsymbol{\phi}$ if and only if $\boldsymbol{\psi} - \boldsymbol{\phi} \notin N^{\mathcal{T}}$.

By the bounded inverse theorem (Treves, 2016), the invertible mapping $\mathcal{T}^{-1} : G \to \Psi/N^{\mathcal{T}}$ exists and is bounded. Thus there exists a constant $K > 0$ that for any $\boldsymbol{\phi}^\theta$, there is a $\boldsymbol{\phi}$ which solves the PDE and

$$\mathbb{E}_{\mathbf{x}\sim p}[\|\boldsymbol{\phi}^\theta(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x})\|^2] = \inf_{\boldsymbol{\xi} \in N^{\mathcal{T}}} \mathbb{E}_{\mathbf{x}\sim p}[\|\boldsymbol{\phi}^\theta(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\xi}(\mathbf{x})\|^2] \tag{65}$$

$$\leq \inf_{\boldsymbol{\xi} \in N^{\mathcal{T}}} \|\boldsymbol{\phi}^\theta - \boldsymbol{\phi} - \boldsymbol{\xi}\|_\Psi^2 \tag{66}$$

$$\leq K\|\mathcal{T}\boldsymbol{\phi}^\theta - \mathcal{T}\boldsymbol{\phi}\|_\mu^2, \tag{67}$$

which concludes the proof. □

E. Experimental Details

The experiments are performed on Nvidia Tesla T4 GPU and Intel Xeon 8352Y CPU. To reproduce the experimental results, please refer to our code in our GitHub repo: `https://github.com/MingzhouFan97/PGPS`. Here we briefly summarize the setup.

E.1. Illustrative Example

Illustrated as Figure 1a, the target is a mixture of two uncorrelated Gaussian with a standard deviation of 0.05 and mean of $(1, 0)$ and $(1.5, 0)$, respectively. The initial particles are sampled from a two-dimensional uncorrelated Gaussian distribution with zero mean and variance of 0.1. 200 particles are considered in this example.

E.2. Gaussian Mixture Examples

To estimate the vector field ϕ_t for PGPS in both experiments, we use a two-layer perceptron with 64 hidden neurons and `Sigmoid` activation function. The particle step-sizes ψ is set to be $\{0.5, 0.1, 0.05, 0.01\}$, the step size for LD, PFG, SVGD, and PGPS adjustment are all set to be 10^{-2}.

E.3. Weight Recovery

The centers of the four modes are deterministically set to be $\mu_1 = [1, 0, 0, 0, 0, 0, 0, 0]$, $\mu_2 = [0, -1, 0, 0, 0, 0, 0, 0]$, $\mu_3 = [0, 0, 1, 0, 0, 0, 0, 0]$, and $\mu_4 = [0, 0, 0, -1, 0, 0, 0, 0]$. The weights are generated by performing `Softmax` over samples from a 4-dimensional standard Gaussian distribution. The NN to estimate the vector field ϕ_t for PGPS is a two-layer perceptron with 128 hidden neurons and `Sigmoid` activation function. The step size for LD, PFG, SVGD, and PGPS adjustment are all set to be 10^{-4}.

E.4. Bayesian Neural Network Inference

The NN to estimate the vector field ϕ_t for PGPS is a two-layer perceptron with 128 hidden neurons and `Sigmoid` activation function. The step size for LD, PFG, SVGD, and PGPS adjustment are all set to be 10^{-1}. The path hyperparameter α is selected from $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$ and β is selected from $\{0.2, 0.4, 0.6, 0.8, 1\}$.

F. Additional Experimental Results

F.1. BNN inference on UCI Dataset

We report the NLL along with the ACC results for Section 5.2.1 in Table 4. In many datasets, SVGD has the best NLL; while in none of these benchmark experiments, SVGD can achieve the best ACC. We conjecture that this is due to variance collapse that SVGD leads to particles gathering close together on the modes and in turn being 'over-confident' on the prediction so that SVGD would tend to get better NLL on certain datasets but worse on ACC. Our PGPS achieves the best ACC and second-best NLL in many of the datasets.

Table 4: Average negative log-likelihood (NLL) and accuracy (ACC) on UCI datasets over five independent runs

	Negative Log-Likelihood (NLL)				Accuracy (ACC)			
	PGPS	SVGD	SGLD	PFG	**PGPS**	SVGD	SGLD	PFG
SONAR	0.5357 ± 0.014	$\mathbf{0.5059} \pm 0.010$	0.5099 ± 0.017	0.5314 ± 0.011	$\mathbf{0.7981} \pm 0.023$	0.7962 ± 0.016	0.7942 ± 0.024	0.7673 ± 0.033
WINEWHITE	1.9788 ± 0.009	1.9905 ± 0.011	$\mathbf{1.9774} \pm 0.050$	1.9898 ± 0.010	0.4520 ± 0.010	0.4520 ± 0.010	$\mathbf{0.4831} \pm 0.049$	0.4520 ± 0.010
WINERED	1.9642 ± 0.012	1.9566 ± 0.012	1.9502 ± 0.096	$\mathbf{1.9359} \pm 0.018$	$\mathbf{0.5938} \pm 0.018$	0.5770 ± 0.018	0.5107 ± 0.096	0.5723 ± 0.019
AUSTRALIAN	0.5042 ± 0.013	$\mathbf{0.4507} \pm 0.006$	0.5732 ± 0.161	0.4511 ± 0.007	0.8620 ± 0.009	0.8626 ± 0.006	0.7362 ± 0.157	$\mathbf{0.8643} \pm 0.006$
HEART	$\mathbf{0.9428} \pm 0.030$	1.0800 ± 0.027	1.0686 ± 0.131	1.0914 ± 0.033	$\mathbf{0.2556} \pm 0.142$	0.1801 ± 0.042	0.2384 ± 0.135	0.1762 ± 0.033
GLASS	1.6853 ± 0.030	$\mathbf{1.6664} \pm 0.027$	1.7083 ± 0.145	1.7162 ± 0.029	$\mathbf{0.5850} \pm 0.080$	0.5383 ± 0.076	0.4561 ± 0.152	0.4505 ± 0.071
COVERTYPE	1.6016 ± 0.014	$\mathbf{1.5981} \pm 0.018$	1.6439 ± 0.082	1.6241 ± 0.011	$\mathbf{0.5899} \pm 0.095$	0.4867 ± 0.006	0.5221 ± 0.084	0.5088 ± 0.053