PATH INTEGRAL SAMPLER: A STOCHASTIC CONTROL APPROACH FOR SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

We present Path Integral Sampler (PIS), a novel algorithm to draw samples from unnormalized probability density functions. The PIS is built on the Schrödinger bridge problem which aims to recover the most likely evolution of a diffusion process given its initial distribution and terminal distribution. The PIS draws samples from the initial distribution and then propagates the samples through the Schrödinger bridge to reach the terminal distribution. Applying the Girsanov theorem, with a simple prior diffusion, we formulate the PIS as a stochastic optimal control problem whose running cost is the control energy and terminal cost is chosen according to the target distribution. By modeling the control as a neural network, we establish a sampling algorithm that can be trained end-to-end. We provide theoretical justification of the sampling quality of PIS in terms of Wasserstein distance when sub-optimal control is used. Moreover, the path integrals theory is used to compute importance weights of the samples to compensate for the bias induced by the sub-optimality of the controller and time-discretization. We experimentally demonstrate the advantages of PIS compared with other start-of-the-art sampling methods on a variety of tasks.

1 INTRODUCTION

We are interested in drawing samples from a target density $\hat{\mu} = Z\mu$ known up to a normalizing constant Z. Although it has been widely studied in machine learning and statistics, generating asymptotically unbiased samples from such unnormalized distribution can still be challenging (Talwar, 2019). In practice, variational inference (VI) and Monte Carlo (MC) methods are two popular frameworks for sampling.

Variational inference employs a density model q, from which samples are easy and efficient to draw, to approximate the target density (Rezende & Mohamed, 2015; Wu et al., 2020). Two important ingredients for variational inference sampling include a distance metric between q and $\hat{\mu}$ to identify good q and the importance weight to account for the mismatch between the two distributions. Thus, in variational inference, one needs to access the explicit density of q, which restricts the possible parameterization of q. Indeed, explicit density models that provide samples and probability density such as Autoregressive models and normalizing flow are widely used in density estimation (Gao et al., 2020a; Nicoli et al., 2020). However, such models impose special structural constraints on the representation of q. For instance, the expressive power of normalizing flows (Rezende & Mohamed, 2015) is constrained by the requirements that the induced map has to be bijective and its Jacobian needs to be easy-to-compute (Wu et al., 2020; Cornish et al., 2020; Grathwohl et al., 2018).

Most MC methods generate samples by iteratively simulating a well-designed Markov chain (MCMC) or sampling ancestrally (MacKay, 2003). Among them, Sequential Monte Carlo and its variants augmented with annealing trick are regarded as state-of-the-art in certain sampling tasks (Chopin & Papaspiliopoulos, 2020). Despite its popularity, MCMC methods may suffer from long mixing time. The short-run performance of MCMC can be difficult to analyze and samples often get stuck in local minima (Nijkamp et al., 2019; Gao et al., 2020b). There are some recent works exploring the possibility of incorporating neural networks to improve MCMC (Spanbauer et al., 2020; Li et al., 2020b). However, evaluating existing MCMC empirically, not to say designing an objective loss function to train network-powered MCMC, is difficult (Liu et al., 2016; Gorham & Mackey, 2017). Most existing works in this direction focus only on designing data-aware propos-

als (Song et al., 2017; Titsias & Dellaportas, 2019) and training such networks can be challenging without expertise knowledge in sampling.

In this work, we propose an efficient sampler termed Path Integral Sampler (PIS) to generate samples by simulating a stochastic differential equation (SDE) in finite steps. Our algorithm is built on the Schrödinger bridge problem (Pavon, 1989; Dai Pra, 1991; Léonard, 2014; Chen et al., 2021) whose original goal was to infer the most likely evolution of a diffusion given its marginal distributions at two time points. With a proper prior diffusion model, this Schrödinger bridge framework can be adopted for the sampling task. Moreover, it can be reformulated as a stochastic control problem (Chen et al., 2016) whose terminal cost depends on the target density $\hat{\mu}$ so that the diffusion under optimal control has terminal distribution $\hat{\mu}$. We model the control policy with a network and develop a method to train it gradually and efficiently. The discrepancy of the learned policy from the optimal policy also provides an evaluation metric for sampling performance. Furthermore, PIS can be made unbiased even with sub-optimal control policy via the path integral theorem to compute the importance weights of samples. Compared with VI that uses explicit density models, PIS uses an implicit model and has the advantage of free-form network design. The explicit density models have weaker expressive power and flexibility compared with implicit models, both theoretically and empirically (Cornish et al., 2020; Chen et al., 2019; Kingma & Welling, 2013; Mohamed & Lakshminarayanan, 2016). Compared with MCMC, PIS is more efficient and is able to generate high-quality samples with fewer steps. Besides, the behavior of MCMC over finite steps can be analyzed and quantified. We show guaranteed sampling quality in terms of Wasserstein distance from the target density for any given sub-optimal policy.

Our algorithm is based on Tzen & Raginsky (2019), where the authors establish the connections between generative models with latent diffusion and stochastic control and justify the expressiveness of such models theoretically. How to realize this model with networks and how the method performs on real datasets are unclear in Tzen & Raginsky (2019). Another closely related work is Wu et al. (2020); Arbel et al. (2021), which extends Sequential Monte Carlo (SMC) by combining deterministic normalizing flow blocks with stochastic MCMC blocks. To be able to evaluate the importance weights efficiently, MCMC blocks need to be chosen based on annealed target distributions carefully. In contrast, in PIS one can design expressive architecture freely and train the model end-to-end without the burden of tuning MCMC kernels, resampling, annealing scheduling. An illustration of the advantages of PIS is presented in Fig 1. We summarize our contributions as follows.

1. We propose Path Integral Sampler (PIS), a generic sampler that generates samples through simulating a target-dependent SDE which can be trained with free-form architecture network design. We derive performance guarantee in terms of the Wasserstein distance to the target density based on the optimality of the learned SDE. 2. An evaluation metric is provided to quantify the performance of learned PIS. By minimizing such evaluation metric, PIS can be trained end-to-end. This metric also provides an estimation of the normalization constants of target distributions. 3. PIS can generate samples without bias even with sub-optimal SDEs by assigning importance weights using path integral theory. 4. Empirically, PIS achieves the state-of-the-art sampling performance in several sampling tasks.

2 SAMPLING AND STOCHASTIC CONTROL PROBLEMS

We begin with a brief introduction to the sampling problem and stochastic control problem. Throughout, we denote by $\tau = \{\mathbf{x}_t, 0 \leq t \leq T\}$ a continuous-time stochastic trajectory.

2.1 SAMPLING PROBLEM

We are interested in drawing samples from a target distribution $\mu(\mathbf{x}) = \hat{\mu}(\mathbf{x})/Z$ in $\boldsymbol{R}^d$ where Z is the normalization constant. Many sampling algorithms rely on constructing a stochastic process that drives the random particles from an initial distribution ν that is easy to sample from, to the target distribution μ.

In the variational inference framework, one seeks to construct a parameterized stochastic process to achieve this goal. Denote by $\Omega = C([0, T]; \boldsymbol{R}^d)$ the path space consisting of all possible trajectories and by $\mathcal{P}$ the measure over Ω induced by a stochastic process with terminal distribution μ at time T. Let $\mathcal{Q}$ be the measure induced by a parameterized stochastic and denote its marginal distribution at

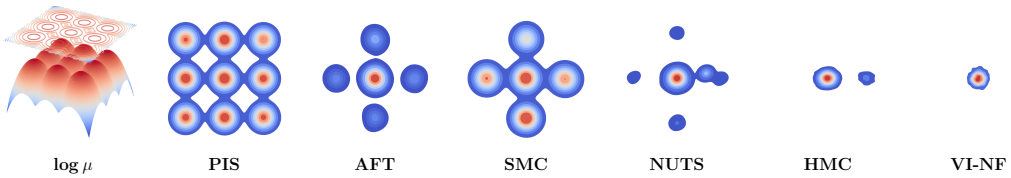


Figure 1: Sampling performance on a challenging 2D unnormalized density model with well-separated modes. Kernel density estimation plots are compared with 2k samples. AFT and SMC use annealing trick with 10 decreasing temperate levels and HMC kernel following (Arbel et al., 2021). Even without annealing trick and resampling, Path Integral Sampler (PIS) generates visually indistinguishable samples from target density with 100 steps. PIS starts $\mathbf{x}_0$ from origin point while others start from a standard Gaussian. We include more details of various approaches in Section 4.

T by $\mu^{\mathcal{Q}}$. Then, by the data processing inequality, the Kullback-Leibler divergence (KL) between marginal distributions $\mu^{\mathcal{Q}}$ and μ can be bounded by

$$D_{\mathrm{KL}}(\mu^{\mathcal{Q}}\|\mu) \leq D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) := \int_{\Omega} d\mathcal{Q} \log \frac{d\mathcal{Q}}{d\mathcal{P}}. \tag{1}$$

Thus, $D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P})$ serves as a performance metric for the sampler, and a small $D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P})$ value corresponds to a good sampler.

2.2 Stochastic control

Consider a model characterized by the stochastic differential equation (SDE) (Särkkä & Solin, 2019)

$$d\mathbf{x}_t = \mathbf{f}(t, \mathbf{x}_t)dt + \mathbf{g}(t, \mathbf{x}_t)(\mathbf{u}_t dt + d\mathbf{w}_t), \ \mathbf{x}_0 \sim \nu \tag{2}$$

where the *drift* term $\mathbf{f} : \boldsymbol{R}^d \to \boldsymbol{R}^d$ is a vector-valued function, and the *diffusion* coeffient $\mathbf{g}$ is a matrix-valued function, $\mathbf{x}_t, \mathbf{u}_t$ denote state and control input respectively, and $\mathbf{w}_t$ denotes standard Brownian motion. In stochastic control, the goal is to find an feedback control strategy that minimizes a certain given cost function.

The standard stochastic control problem can be associated with any cost. In this paper, we only consider the cost of the form

$$\mathbb{E}[\int_0^T \frac{1}{2} \|\mathbf{u}_t\|^2 dt + \Psi(\mathbf{x}_T) \mid \mathbf{x}_0 \sim \nu], \tag{3}$$

where Ψ represents the terminal cost. The corresponding optimal control problem can be solved via dynamic programming (Bertsekas et al., 2000), which amounts to solving the Hamilton-Jacobi-Bellman (HJB) equation (Evans, 1998)

$$\frac{\partial V_t}{\partial t} + \mathbf{f} \cdot \nabla V_t - \frac{1}{2}\nabla V_t' \mathbf{g}\mathbf{g}' \nabla V_t + \frac{1}{2} \mathrm{Tr}(\mathbf{g}\mathbf{g}' \nabla^2 V_t) = 0, \ V_T(\cdot) = \Psi(\cdot). \tag{4}$$

The space-time function $V_t(\mathbf{x})$ is known as *cost-to-go* function or *value function*. The optimal policy can be inferred from $V_t(\mathbf{x})$ as (Pavon, 1989)

$$\mathbf{u}_t^*(\mathbf{x}) = -\mathbf{g}(t, \mathbf{x})' \nabla V_t(\mathbf{x}). \tag{5}$$

3 Path Integral Sampler

It turns out that, with a proper choice of initial distribution ν and terminal loss function Ψ, the stochastic control problem coincides with sampling problem, and the optimal policy drives samples from ν to μ perfectly. The process under optimal control can be viewed as the posterior of uncontrolled dynamics conditioned on target distribution as illustrated in Fig 2. Throughout, we denote by $\mathcal{Q}^u$ the path measure associated with control policy $\mathbf{u}$. We also denote by μ^0 the terminal distribution of the uncontrolled process $\mathcal{Q}^0$. For the ease of presentation, we begin with sampling from a normalized density μ, and then generalize the results to unnormalized $\hat{\mu}$ in Section 3.4.

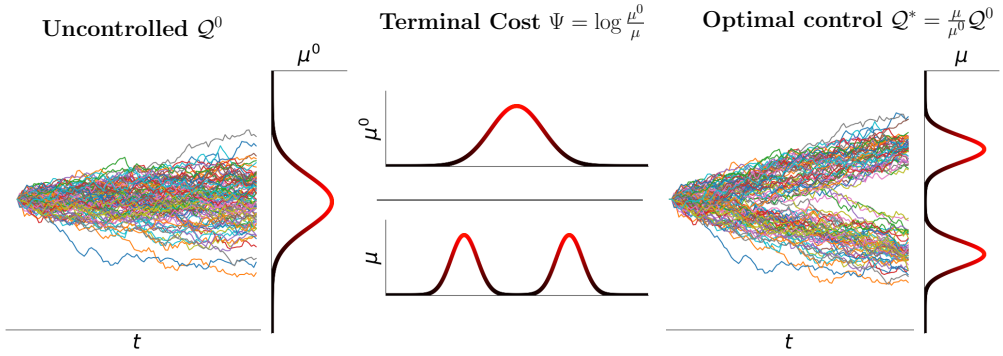


Figure 2: Illustration of Path Integral Sampler (PIS). The optimal policy of a specific stochastic control problem where a terminal cost function is chosen according to the given target density μ, can generate unbiased samples within a finite time horizon.

3.1 Path Integral and Value Function

Due to the special cost structure, the nonlinear HJB eq (4) can be transformed into a linear partial differential equation (PDE)

$$\frac{\partial \phi_t}{\partial t} + \mathbf{f} \cdot \nabla \phi_t + \frac{1}{2} \mathrm{Tr}(\mathbf{g}\mathbf{g}' \nabla^2 \phi_t) = 0, \ \phi_T(\cdot) = \exp\{-\Psi(\cdot)\} \tag{6}$$

by logarithmic transformation (Särkkä & Solin, 2019) $V_t(\mathbf{x}) = -\log \phi_t(\mathbf{x})$. By the celebrated Feynman-Kac formula (Øksendal, 2003), the above has solution

$$\phi_t(\mathbf{x}) = \mathbb{E}_{\mathcal{Q}^0}[\exp(-\Psi(\mathbf{x}_T))|\mathbf{x}_t = \mathbf{x}]. \tag{7}$$

We remark that eq (7) implies that the optimal value function can be evaluated without knowing the optimal policy since the above expectation is with respect to the uncontrolled process $\mathcal{Q}^0$. This is exactly the Path Integral control theory (Thijssen & Kappen, 2015). Furthermore, the optimal control at $(t, \mathbf{x})$ is

$$\mathbf{u}_t^*(\mathbf{x}) = \mathbf{g}(t, \mathbf{x})' \nabla \log \phi_t(\mathbf{x}) = \lim_{s \searrow t} \frac{\mathbb{E}_{\mathcal{Q}^0}\{\exp\{-\Psi(\mathbf{x}_T)\} \int_t^s d\mathbf{w}_t \mid \mathbf{x}_t = \mathbf{x}\}}{(s-t)\mathbb{E}_{\mathcal{Q}^0}\{\exp\{-\Psi(\mathbf{x}_T)\} \mid \mathbf{x}_t = \mathbf{x}\}}, \tag{8}$$

meaning that $\mathbf{u}_t^*(\mathbf{x})$ can also be estimated by uncontrolled trajectories.

3.2 Sampling as a Stochastic Optimal Control Problem

For a given $\mathbf{f}, \mathbf{g}$, there are infinite choices of $\mathbf{u}$ such that eq (2) has terminal distribution μ. We are interested in the one that minimizes the KL divergence to the prior uncontrolled process. This is exactly the Schrödinger bridge problem (Pavon, 1989; Dai Pra, 1991; Chen et al., 2016; 2021), which has been shown to have a stochastic control formulation with cost being control efforts. In cases where ν is a Dirac distribution, it is the same as the stochastic control problem in Section 2.2 with a proper terminal cost as characterized in the following result (Tzen & Raginsky, 2019).

Theorem 1 (Proof in appendix A)**.** *When ν is a Dirac distribution and terminal loss is chosen as $\Psi(\mathbf{x}_T) = \log \frac{\mu^0(\mathbf{x}_T)}{\mu(\mathbf{x}_T)}$, the distribution $\mathcal{Q}^*$ induced by the optimal control policy is*

$$\mathcal{Q}^*(\tau) = \mathcal{Q}^0(\tau|\mathbf{x}_T)\mu(\mathbf{x}_T). \tag{9}$$

Moreover, $\mathcal{Q}^*(\mathbf{x}_T) = \mu(\mathbf{x}_T)$.

To gain more insight, consider the KL divergence

$$D_{\mathrm{KL}}(\mathcal{Q}^u(\tau)\|\mathcal{Q}^0(\tau|\mathbf{x}_T)\mu(\mathbf{x}_T)) = D_{\mathrm{KL}}(\mathcal{Q}^u(\tau)\|\mathcal{Q}^0(\tau)\frac{\mu(\mathbf{x}_T)}{\mu^0(\mathbf{x}_T)}) = D_{\mathrm{KL}}(\mathcal{Q}^u\|\mathcal{Q}^0) + \mathbb{E}_{\mathcal{Q}^u}[\log \frac{\mu^0}{\mu}]. \tag{10}$$

Thanks to the Girsanov theorem (Särkkä & Solin, 2019),

$$\frac{d\mathcal{Q}^u}{d\mathcal{Q}^0} = \exp(\int_0^T \frac{1}{2}\left\|\mathbf{u}_t\right\|^2 dt + \mathbf{u}_t' d\mathbf{w}_t). \tag{11}$$

It follows that

$$D_{\mathrm{KL}}(\mathcal{Q}^u \| \mathcal{Q}^0) = \mathbb{E}_{\mathcal{Q}^u}[\int_0^T \frac{1}{2}\left\|\mathbf{u}_t\right\|^2 dt]. \tag{12}$$

Plugging eq (12) into eq (10) yields

$$D_{\mathrm{KL}}(\mathcal{Q}^u(\tau) \| \mathcal{Q}^0(\tau|\mathbf{x}_T)\mu(\mathbf{x}_T)) = \mathbb{E}_{\mathcal{Q}^u}[\int_0^T \frac{1}{2}\left\|\mathbf{u}_t\right\|^2 dt + \log\frac{\mu^0(\mathbf{x}_T)}{\mu(\mathbf{x}_T)}], \tag{13}$$

which is exactly the cost defined in eq (3) with $\Psi = \log\frac{\mu^0}{\mu}$. Theorem 1 implies that once the optimal control policy that minimizes this cost is found, it can also drive particles from $\mathbf{x}_0 \sim \nu$ to $\mathbf{x}_T \sim \mu$.

3.3 Optimal control policy and sampler

Optimal Policy Representation: Consider the sampling strategy from a given target density by simulating SDE in eq (2) under optimal control. Even though the optimal policy is given in eq (8), only in rare case it has an analytic closed-form.

For more general target distributions, we can instead evaluate the value function eq (7) via empirical samples using Monte Carlo. The approach is essentially importance sampling whose proposal distribution is the uncontrolled dynamics. However, this approach has two drawbacks. First, it is known that the estimation variance can be intolerably high when the proposal distribution is not close enough to the target distribution (MacKay, 2003). Second, even if the variance is acceptable, without a good proposal, the required samples size increases exponentially with dimension, which prevents the algorithm from being used in high or even medium dimension problem (Au & Beck, 2003).

To overcome the above shortcomings, we parameterize the control policy with a neural network $\mathbf{u}_\theta$. We seek a control policy that minimizes the cost

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \mathbb{E}_{\mathcal{Q}^u}[\int_0^T \frac{1}{2}\left\|\mathbf{u}_t\right\|^2 dt + \log\frac{\mu^0(\mathbf{x}_T)}{\mu(\mathbf{x}_T)}]. \tag{14}$$

In the space of policies, eq (14) also serves as distance metric between $\mathbf{u}_\theta$ and $\mathbf{u}^*$ as in eq (13).

Gradient-informed Policy Representation: It is believed that proper prior information can significantly boost the performance of neural network (Goodfellow et al., 2016). The score $\nabla \log \mu(\mathbf{x})$ has been used widely to improve the proposal distribution in MCMC (Li et al., 2020b; Hoffman & Gelman, 2014) and often leads to better results compared with proposals without gradient information. In the same spirit, we incorporate $\nabla \log \mu(\mathbf{x})$ and parameterize the policy as

$$\mathbf{u}_t(\mathbf{x}) = \mathrm{NN}_1(t, \mathbf{x}) - \mathrm{NN}_2(t) \times \nabla \log \mu(\mathbf{x}), \tag{15}$$

where NN_1 and NN_2 are two neural networks. Empirically, we also found that the gradient information leads to faster convergence and smaller discrepancy $D_{\mathrm{KL}}(\mathcal{Q}^u \| \mathcal{Q}^*)$. We remark that PIS with policy eq (15) can be viewed as a modulated Langevin dynamics (MacKay, 2003) that achieves μ within finite time T instead of infinite time.

Evaluating Loss: To optimize $\mathbf{u}_\theta$ with objective loss eq (14), we need to evaluate $\mu^0(\mathbf{x}_T)$ numerically. For simplicity, we use $\mathbf{x}_0 \sim \delta_0$ and $\mathbf{f} = 0, \mathbf{g} = I$ so that $\mu^0(\mathbf{x}_T)$ has a simple closed-form. Empirically it works well. We include more discussion of $\mathbf{f}, \mathbf{g}$ and implementation in appendix F.1.

Optimizing $\mathbf{u}_\theta$ requires the gradient of loss in eq (14), which involves $\mathbf{u}_t$ and the terminal state $\mathbf{x}_T$. To calculate gradients, we rely on backpropagation through trajectory. We train the control policy with recent techniques of Neural SDEs (Li et al., 2020a), which greatly reduce memory consumption during training. We augment the origin SDE with state $\int_0^t \frac{1}{2}\left\|\mathbf{u}\right\|^2 ds$ such that the whole training can be conducted end to end. The full training procedure in provided in Algorithm 1.

Algorithm 1 Training

Define: $\mathbf{f}_{aug}(t, [\mathbf{x}_t, y_t]) = [\mathbf{f}(t, \mathbf{x}_t) + \mathbf{g}(t, \mathbf{x}_t)\mathbf{u}_{\theta t}(\mathbf{x}_t), \frac{1}{2}\left\|\mathbf{u}_{\theta t}(\mathbf{x}_t)\right\|^2]$
Define: $\mathbf{g}_{aug}(t, [\mathbf{x}_t, y_t]) = [\mathbf{g}(t, \mathbf{x}_t), 0]$
repeat
  Vector: $\mathbf{x}_0 = 0$, Scalar: $y_0 = 0$
  $\mathbf{x}_T, y_T = \text{sdeint}(\mathbf{f}_{aug}, \mathbf{g}_{aug}, [\mathbf{x}_0, y_0], [0, T])$
  Gradient descent step $\nabla_\theta[y_T + \log \frac{\mu^0(\mathbf{x}_T)}{\mu(\mathbf{x}_T)}]$
until converged

Wasserstein distance bound: The PIS trained by Algorithm 1 can not generate unbiased samples from the target distribution μ for two reasons. First, due to the non-convexity of networks and randomness of stochastic gradient descent, there is no guarantee that the learned policy is optimal. Second, even if the learned policy is optimal, the time-discretization error in simulating SDEs is inevitable. Fortunately, the following theorem quantifies the Wasserstein distance between the sampler and the target density. (More details can be found in appendix C)

Theorem 2. *Under Condition 1, with sampling step size Δt, if $\left\|\mathbf{u}_t^* - \mathbf{u}_t\right\|^2 \leq d\epsilon$ for any t, then*

$$W_2(\mathcal{Q}^u(\mathbf{x}_T), \mu(\mathbf{x}_T)) = \mathcal{O}(\sqrt{Td(\Delta t + \epsilon)}). \tag{16}$$

3.4 Importance Sampling

The training procedure for PIS does not guarantee its optimality. To compensate for the mismatch between the trained policy and the optimal policy, we introduce importance weight to calibrate generated samples. The importance weight can be calculated by (more details in appendix B)

$$w^u(\tau) = \frac{d\mathcal{Q}^u(\tau)}{d\mathcal{Q}^*(\tau)} = \exp(\int_0^T \frac{1}{2}\left\|\mathbf{u}_t\right\|^2 dt + \mathbf{u}_t' d\mathbf{w}_t + \Psi(\mathbf{x}_T)). \tag{17}$$

We note eq (17) resembles training objective eq (14). Indeed, eq (14) is the average of logarithm of eq (17). If the trained policy is optimal, that is, $\mathcal{Q}^u = \mathcal{Q}^*$, all the particles share the same weight. We summarize the sampling algorithm in Algorithm 2.

Algorithm 2 Sampling

Vector: $\mathbf{x}_0 = 0$, Scalar: $w_0 = 0$
for $i \leftarrow 1$ to N **do**
  $\Delta t = t_i - t_{i-1}, \Delta\mathbf{w} \sim \mathcal{N}(0, \Delta t\mathcal{I})$,
  $\mathbf{x}_i = \mathbf{x}_{i-1} + \mathbf{f}\Delta t + \mathbf{g}(\mathbf{u} + \Delta\mathbf{w})$
  $w_i = w_{i-1} + \mathbf{u}'\Delta\mathbf{w} + \frac{1}{2}\left\|\mathbf{u}\right\|^2$
end for
Outputs: $\mathbf{x}_N, \exp(w_N + \log \frac{\mu^0(\mathbf{x}_N)}{\mu(\mathbf{x}_N)})$

Effective Sample Size: The Effective Sample Size (ESS), $\text{ESS}^u = \frac{1}{\mathbb{E}_{\mathcal{Q}^u}[(w^u)^2]}$, is a popular metric to measure the variance of importance weights. ESS is often accompanied by resampling trick (Tokdar & Kass, 2010) to mitigate deterioration of sample quality. ESS is also regarded as a metric for quantifying goodness of sampler based on importance sampling. Low ESS means that estimation or downstream tasks based on such sampling methods may suffer from a high variance. ESS of most importance samplers is decreasing along the time. However, thanks to the adaptive control policy in PIS, we can quantify the ESS of PIS based on the optimality of learned policy with proof in appendix D.

Theorem 3 (Corollary 7 (Thijssen & Kappen, 2015))**.** *If $\max_{t,\mathbf{x}}\left\|\mathbf{u}_t(\mathbf{x}) - \mathbf{u}_t^*(\mathbf{x})\right\|^2 \leq \frac{\epsilon}{T}$, then*

$$\frac{1}{\mathbb{E}_{\mathcal{Q}^u}[(w^u)^2]} \geq 1 - \epsilon.$$

Estimation of normalization constants: In most sampling problems we only have access to the target density up to a normalization constant, denoted by $\hat{\mu} = Z\mu$. PIS can still generate samples following the same protocol with new terminal cost $\hat{\Psi} = \log \frac{\mu^0}{\hat{\mu}} = \Psi - \log Z$. The additional constant $-\log Z$ does not affect the optimal policy and the optimization of $\mathbf{u}_\theta$. As a byproduct, we can estimate the normalization constants as follows (more details in appendix E).

Theorem 4. *For any given policy $\mathbf{u}$, the logarithm of normalization constant is bounded below by*

$$\mathbb{E}_{\tau\sim\mathcal{Q}^u}[-\hat{S}^u(\tau)] \leq \log Z \tag{18}$$

Figure 3: Sampling performance on rings-shape density function with 100 steps. The gradient information can help PIS-Grad and MCMC algorithm get out of local minima.

where $\hat{S}^u(\tau) = \int_0^T \frac{1}{2} \|\mathbf{u}_t(\mathbf{x}_t)\|^2 \, dt + \mathbf{u}_t'(\mathbf{x}_t) d\mathbf{w}_t + \hat{\Psi}(\mathbf{x}_T)$. *The equality holds only when* $\mathbf{u} = \mathbf{u}^*$. *For any sub-optimal policy, an unbiased estimation of* Z *based on importance sampling is*

$$Z = \mathbb{E}_{\tau \sim \mathcal{Q}^u}[\exp(-\hat{S}^u(\tau))]. \tag{19}$$

4 Experiments

We present empirical evaluations of PIS and several baselines as well as details of practical implementation in this section. Inspired by Arbel et al. (2021), we conduct experiments for tasks of estimating normalization constant and Bayesian inference.

We consider three types of relevant methods. The first category is gradient-guided MCMC methods without the annealing trick. It includes the Hamiltonian Monte Carlo (HMC) (MacKay, 2003) and No-U-Turn Sampler (NUTS) (Hoffman & Gelman, 2014). The second is Sequential Monte Carlo with annealing trick (SMC), which is regarded as state-of-the-art sampling algorithm (Chopin & Papaspiliopoulos, 2020). We consider a standard instance of SMC samplers and the recently proposed Annealed Flow Transport Monte Carlo (AFT) (Arbel et al., 2021). Both use a default 10 temperature levels with a linear annealing scheme. Lastly, variational normalizing flow (VI-NF) (Rezende & Mohamed, 2015) is also included for comparison.

The number of steps N of MCMC algorithm and the number of SDE time-discretization steps for PIS work as a proxy for benchmarking computation times. The time complexity of evaluating drift and control function in PIS is comparable with proposal calculation in MCMC. Since we focus on evaluating policy after training, training consumption is not included in the comparison. The training time of PIS highly depends on efficiency of training NeuralSDEs. One future direction is to investigate additional regularizations and structured SDE to speed up the training.

We also investigate the effects of two different network architectures for Path Integral Sampler. The first one is a time-conditioned neural network without any prior information, which we denote as *PIS-NN*, while the second one incorporates the gradient information of the given energy function, denoted as *PIS-Grad*. When we have an analytical form for the ground truth optimal policy, the policy is denoted as *PIS-GT*. We remark that *PIS-GT* serves the purpose of validating the connections between the optimality of the policy and the effectiveness of Path Integral Sampler. The subscript *RW* is to distinguish PIS with path integral importance weights eq (17) that uses eq (19) to estimate normalization constants from the ones without importance weights that use the bound in eq (18) to estimate Z. For approaches without the annealing trick, we take default $N = 100$ unless otherwise stated. With annealing, N steps are the default for each temperature level, thus AFT and SMC use 10 times more steps compared with HMC and PIS. We include more details about hyperparameters, training, discussions of NUTS, and experiments with large N in appendices F and G.

4.1 PIS-Grad vs PIS-NN: Importance of gradient guidance

We observed that the advantage of PIS-Grad over PIS-NN is more clearer when the target density has multiple modes as it in a toy example shown in Fig 3. The objective $D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{Q}^*)$ is known to have *zero forcing*. When the modes of the density are well separated and $\mathcal{Q}$ is not expressive enough, minimizing $D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{Q}^*)$ can drive $\mathcal{Q}(\tau)$ to zero on some area, even if $\mathcal{Q}^*(\tau) > 0$ (Fox & Roberts, 2012). PIS-NN and VI-NF generate very similar samples that almost cover half the inner ring. The training objective function of VI-NF can also be viewed as minimizing KL divergence between two trajectory distributions (Wu et al., 2020). The added noise during the process can encourage

exploration but it is unlikely such noise only can overcome the local minima. On the other hand, the gradient information can help avoid local minima and provide hopeful exploring directions.

	MG(d=2)			Funnel(d=10)			LGCP(d=1600)		
	B	S	A	B	S	A	B	S	A
PIS$_{RW}$-GT	**-0.012**	**0.013**	**0.018**	-	-	-	-	-	-
PIS-NN	-1.691	0.370	1.731	-0.098	**5e-3**	0.098	-92.4	6.4	92.62
PIS-Grad	-0.440	0.024	0.441	-0.103	9e-3	0.104	-13.2	3.21	13.58
PIS$_{RW}$-NN	-1.192	0.482	1.285	-0.018	7e-3	0.02	-60.8	4.81	60.99
PIS$_{RW}$-Grad	-0.021	0.030	0.037	**-0.008**	9e-3	**0.012**	**-1.94**	**0.91**	**2.14**
AFT	-0.509	0.24	0.562	-0.208	0.193	0.284	-3.08	1.59	3.46
SMC	-0.362	0.293	0.466	-0.216	0.157	0.267	-435	14.7	436
NUTS	-1.871	0.527	1.943	-0.835	0.257	0.874	-1.3e3	8.01	1.3e3
HMC	-1.876	0.527	1.948	-0.835	0.257	0.874	-1.3e3	8.01	1.3e3
VI-NF	-1.632	0.965	1.896	-0.236	0.0591	0.243	-77.9	5.6	78.2

Table 1: Benchmarking on mode separated mixture of Gaussian (MG), Funnel distribution and Log Gaussian Cox Process (LGCP) for estimation log normalization constants. *B* and *S* stand for estimation bias and standard deviation among 100 runs and $A^2 = B^2 + S^2$

4.2 BENCHMARKING DATASETS

Mode-separated mixture of Gaussian: We consider the mixture of Gaussian in 2D dimension. We notice when Gaussian modes are not far from each other, all methods work well. However, when we reduce the variance of Gaussian distribution and separate the modes of Gaussian, the advantage of PIS becomes clear even in the low dimension tasks. We generate 2000 samples from each method and kernel density estimate (KDE) plot are provided in Fig 1. PIS generates samples that are visually indistinguishable from the given density.

Funnel distribution: We consider the popular testing distribution in MCMC literature (Hoffman & Gelman, 2014; Hoffman et al., 2019), 10-dimensional Funnel distribution:

$$x_0 \sim \mathcal{N}(0, 9), \quad x_{1:9}|x_0 \sim \mathcal{N}(0, \exp(x_0)\mathbf{I}).$$

The distribution can be pictured like a funnel - with x_0 wide at the mouth of funnel, getting smaller as the funnel narrows.

Log Gaussian Cox Process: We further investigate estimating the normalization constant of the challenging log Gaussian Cox process (LGCP), which is designed for modeling the positions of Finland pine saplings. In LGCP (Salvatier et al., 2016), an underlying field λ of positive real values is modeled using an exponentially-transformed Gaussian process. Then λ is used to parameterized Poisson points process to model locations of pine saplings. The posterior density of λ is

$$\lambda(\mathbf{x}) \sim \exp(-\frac{(\mathbf{x}-\mu)^T K^{-1}(\mathbf{x}-\mu)}{2}) \prod_{i \in d} \exp(x_i y_i - \alpha \exp x_i), \tag{20}$$

where d denotes the size of discretized grid and y_i denotes observation information. Modeling parameters, including normal distribution and α follow Arbel et al. (2021) (See appendix F).

From the Tab 1, it clearly shows the advantages of PIS among the above three datasets. And importance weight also helps improve the estimation of log normalization constants based on the comparison between PIS$_{RW}$ and PIS. We found PIS-Grad that trained with gradient information outperforms PIS-NN. The difference is more obvious in datasets that have well-separated modes, such as MG and LGCP, and less obvious on one mode tasks like Funnel.

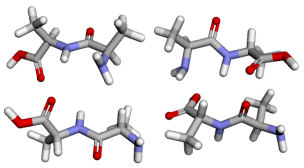
In all cases, PIS$_{RW}$-Grad is better than AFT and SMC. Interestingly, even *without* annealing and gradient information of target density, PIS$_{RW}$-NN can outperform SMC with annealing trick and HMC kernel for the Funnel distribution.

4.3 ALANINE DIPEPTIDE

KL.	μ	ϕ	η_1	ψ	η_2	η_3
VI-NF	175.6 ±4.5	24.2± 4.1	3.1 ± 0.05	14.6± 6.4	7e-2±5e-3	**8.5e-2±3.5e-3**
SMC	183.3 ±2.3	18.3± 2.1	0.32 ± 0.08	9.6 ± 1.2	0.12±0.05	0.15 ± 9e-3
SNF	181.8 ±0.75	6.3± 0.71	0.17±0.05	1.58 ± 0.36	0.11± 0.03	8.8e-2 ±8e-3
AFT	176.4 ±0.98	6.7± 0.58	**0.16 ± 0.05**	1.54 ± 0.09	**5e-2±7e-3**	9.3e-2±8e-3
PIS-NN	**171.3 ±0.61**	**5.2± 0.35**	0.32±0.03	**1.03 ± 0.23**	**5e-2±5e-3**	8.7e-2±3e-3

Table 2: KL-divergences comparison among variational approaches of generated density with target density in overall atom states distribution and five multimodal torsion angles. Mean and standard deviation are conducted with five different random seeds.

Figure 4: Sampled Alanine dipeptide molecules



Thanks to the great success achieved by flow models in the generation of asymptotically unbiased samples from physics models (LeCun, 1998), we explore the applications in the sampling of molecular structure from a simulation of Alanine dipeptide as introduced in Wu et al. (2020). The target density of molecule is $\hat{\mu} = \exp(-E(\mathbf{x}_{[0:65]}) - \frac{1}{2}\left\|\mathbf{x}_{[66:131]}\right\|^2)$. More details about the energy function E are included in appendix F.

We compare PIS with popular variational approaches used in generating samples from the above model. More specifically, we consider VI-NF, and Stochastic Normalizing Flow (SNF) (Wu et al., 2020). SNF is very close to AFT (Arbel et al., 2021). Both of them couple deterministic normalizing flow layers and MCMC blocks except SNF uses an amortized structure. We include more details of MCMC kernel and modification in appendix F. We show the generated molecular in Fig 4 and quantitative comparison of KL divergence in Tab 2, including overall atom states distribution and five multimodal torsion angles (backbone angles ϕ, ψ and methyl rotation angles η_1, η_2, η_3). We remark that unweighted samples are used to approximate the density of torsion angles and all approaches do not use gradient information. Clearly, PIS gives lower divergence.

4.4 Sampling in Variational Autoencoder latent space

In this experiment we investigate sampling in the latent space of a trained Variational Autoencoder (VAE). VAE aims to minimize

$$D_{\mathrm{KL}}(q(\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})),$$

where $q_\phi(\mathbf{z}|\mathbf{x})$ represents encoder and p_θ for a decoder with latent variable $\mathbf{z}$ and data $\mathbf{x}$. We are interested in posterior distribution

$$\mathbf{z} \sim p(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z}). \tag{21}$$

Table 3: Estimation of $\log p_\theta(x)$ of a trained VAE.

	B	S	$\sqrt{\mathrm{B}^2 + \mathrm{S}^2}$
VI-NF	-2.3	0.76	2.42
AFT	-1.7	0.95	1.96
SMC	-10.6	2.01	10.79
PIS$_{RW}$-NN	-1.9	0.81	2.06
PIS$_{RW}$-Grad	**-0.87**	**0.31**	**0.92**

The normalization constant of such target unnormalized density function $p(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$ is exactly the likelihood of data points $p_\theta(\mathbf{x})$, which serves as an evaluation metric for the trained VAE.

We investigate a vanilla VAE model trained with plateau loss on the binary MNIST (LeCun, 1998) dataset. For each distribution, we regard the average estimation from 10 long-run SMC with 1000 temperature levels as the ground truth normalization constant. We choose 100 images randomly and run the various approaches on estimating normalization of those posterior distributions in eq (21) and report the average performance in Tab 3. PIS has a lower bias and variance.

5 Conclusion

In this work, we proposed a new sampling algorithm, Path Integral Sampler, based on the connections between sampling and stochastic control. The control can drive particles from an initial distribution to a target density perfectly when the policy is optimal for a target-dependent optimal control problem. Furthermore, we provide a calibration based on importance weights, ensuring sampling quality even with sub-optimal policies. In the future, we plan to apply this new sampling algorithm to energy-based models learning and higher dimension tasks.

6 REPRODUCIBILITY STATEMENT

The detailed discussion on assumptions and proof of theorems presented in the main paper is included in appendices A and C to E. For empirical experiments, the training settings and implementation tips are included in appendices F and G. We also include an implementation based on PyTorch (Paszke et al., 2019) in the supplementary material.

REFERENCES

Michael Arbel, Alexander GDG Matthews, and Arnaud Doucet. Annealed flow transport monte carlo. arXiv preprint arXiv:2102.07501, 2021.

Siu-Kui Au and JL Beck. Important sampling in high dimensions. Structural safety, 25(2):139–163, 2003.

Dimitri P Bertsekas et al. Dynamic programming and optimal control: Vol. 1. Athena scientific Belmont, 2000.

Ricky T. Q. Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In Advances in Neural Information Processing Systems, 2019.

Yongxin Chen, Tryphon T Georgiou, and Michele Pavon. On the relation between optimal transport and Schrödinger bridges: A stochastic control viewpoint. Journal of Optimization Theory and Applications, 169(2):671–691, 2016.

Yongxin Chen, Tryphon T Georgiou, and Michele Pavon. Stochastic control liaisons: Richard Sinkhorn meets Gaspard Monge on a Schrödinger bridge. SIAM Review, 63(2):249–313, 2021.

Nicolas Chopin and Omiros Papaspiliopoulos. An introduction to sequential Monte Carlo. Springer, 2020.

Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In International Conference on Machine Learning, pp. 2133–2143. PMLR, 2020.

Paolo Dai Pra. A stochastic control approach to reciprocal diffusion processes. Applied mathematics and Optimization, 23(1):313–329, 1991.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. arXiv preprint arXiv:1605.08803, 2016.

Peter Eastman, Jason Swails, John D Chodera, Robert T McGibbon, Yutong Zhao, Kyle A Beauchamp, Lee-Ping Wang, Andrew C Simmonett, Matthew P Harrigan, Chaya D Stern, et al. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. PLoS computational biology, 13(7):e1005659, 2017.

Bradley Efron. Tweedie's formula and selection bias. Journal of the American Statistical Association, 106(496):1602–1614, 2011.

Ronen Eldan, Joseph Lehec, and Yair Shenfeld. Stability of the logarithmic sobolev inequality via the föllmer process. In Annales de l'Institut Henri Poincaré, Probabilités et Statistiques, volume 56, pp. 2253–2269. Institut Henri Poincaré, 2020.

Lawrence C Evans. Partial differential equations. Graduate studies in mathematics, 19(4):7, 1998.

Charles W Fox and Stephen J Roberts. A tutorial on variational bayesian inference. Artificial intelligence review, 38(2):85–95, 2012.

Christina Gao, Joshua Isaacson, and Claudius Krause. i-flow: High-dimensional integration and sampling with normalizing flows. Machine Learning: Science and Technology, 1(4):045023, 2020a.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. arXiv preprint arXiv:2012.08125, 2020b.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning, volume 1. MIT Press, 2016.

Jackson Gorham and Lester Mackey. Measuring sample quality with kernels. In International Conference on Machine Learning, pp. 1292–1301. PMLR, 2017.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. arXiv preprint arXiv:1810.01367, 2018.

Matthew Hoffman, Pavel Sountsov, Joshua V Dillon, Ian Langmore, Dustin Tran, and Srinivas Vasudevan. Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. arXiv preprint arXiv:1903.03704, 2019.

Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. Journal of Machine Learning Research, 15(1):1593–1623, 2014.

Jian Huang, Yuling Jiao, Lican Kang, Xu Liao, Jin Liu, and Yanyan Liu. Schr schrödinger-föllmer sampler: Sampling without ergodicity. arXiv preprint arXiv:2106.10880, 2021.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. arXiv preprint arXiv:1806.07572, 2018.

Patrick Kidger, James Foster, Xuechen Li, and Terry Lyons. Efficient and accurate gradients for neural sdes. arXiv preprint arXiv:2105.13493, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

Yann LeCun. The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/, 1998.

Christian Léonard. A survey of the schrödinger problem and some of its connections with optimal transport. Discrete & Continuous Dynamical Systems, 34(4):1533, 2014.

Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In International Conference on Artificial Intelligence and Statistics, pp. 3870–3882. PMLR, 2020a.

Zengyi Li, Yubei Chen, and Friedrich T Sommer. A neural network mcmc sampler that maximizes proposal entropy. arXiv preprint arXiv:2010.03587, 2020b.

Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In International conference on machine learning, pp. 276–284. PMLR, 2016.

David JC MacKay. Information theory, inference and learning algorithms. Cambridge university press, 2003.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. arXiv preprint arXiv:1610.03483, 2016.

Jesper Møller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log gaussian cox processes. Scandinavian journal of statistics, 25(3):451–482, 1998.

Kim A Nicoli, Shinichi Nakajima, Nils Strodthoff, Wojciech Samek, Klaus-Robert Müller, and Pan Kessel. Asymptotically unbiased estimation of physical observables with neural samplers. Physical Review E, 101(2):023304, 2020.

Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. arXiv preprint arXiv:1904.09770, 2019.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. Science, 365(6457), 2019.

Bernt Øksendal. Stochastic differential equations. In Stochastic differential equations, pp. 65–84. Springer, 2003.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32: 8026–8037, 2019.

Michele Pavon. Stochastic control and nonequilibrium thermodynamical systems. Applied Mathematics and Optimization, 19(1):187–202, 1989.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In International Conference on Machine Learning, pp. 1530–1538. PMLR, 2015.

John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in python using pymc3. PeerJ Computer Science, 2:e55, 2016.

Simo Särkkä and Arno Solin. Applied stochastic differential equations, volume 10. Cambridge University Press, 2019.

Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-nice-mc: Adversarial training for mcmc. arXiv preprint arXiv:1706.07561, 2017.

Span Spanbauer, Cameron Freer, and Vikash Mansinghka. Deep involutive generative models for neural mcmc. arXiv preprint arXiv:2006.15167, 2020.

Kunal Talwar. Computational separations between sampling and optimization. arXiv preprint arXiv:1911.02074, 2019.

Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. arXiv preprint arXiv:2006.10739, 2020.

Sep Thijssen and HJ Kappen. Path integral control and state-dependent feedback. Physical Review E, 91(3):032104, 2015.

Michalis Titsias and Petros Dellaportas. Gradient-based adaptive markov chain monte carlo. Advances in Neural Information Processing Systems, 32:15730–15739, 2019.

Surya T Tokdar and Robert E Kass. Importance sampling: a review. Wiley Interdisciplinary Reviews: Computational Statistics, 2(1):54–60, 2010.

Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In Conference on Learning Theory, pp. 3084–3114. PMLR, 2019.

Hao Wu, Jonas Köhler, and Frank Noe. Stochastic normalizing flows. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 5933–5944. Curran Associates, Inc., 2020.

A PROOF OF THEOREM 1

Before proving our main theorem, we introduce the following important lemma.

Lemma 4.1 (Dai Pra (1991); Pavon (1989))**.** *The transition density associated with optimal control policy* $\mathbf{u}^*$ *for eq* (2) *and eq* (3) *follows*

$$Q^*_{s,t}(\mathbf{x},\mathbf{y}) = \mathcal{Q}^0_{s,t}(\mathbf{x},\mathbf{y})\frac{\phi_t(\mathbf{y})}{\phi_s(\mathbf{x})}, \tag{22}$$

where $\mathcal{Q}^u_{s,t}(\mathbf{x},\mathbf{y})$ *denote the transition probability from state* $\mathbf{x}$ *at time* s *to state* $\mathbf{y}$ *at time* t*.*

Proof of Theorem 1: We denote initial Dirac distribution by $\nu = \delta_{\bar{\mathbf{x}}_0}$. Combining eq (7) and $V_t(\mathbf{x}) = -\log\phi_t(\mathbf{x})$ we obtain

$$V_0(\bar{\mathbf{x}}_0) = -\log E_{\mathcal{Q}^0}[\exp(-\Psi(\mathbf{x}_T))] = -\log(\int \frac{\mu}{\mu^0}d\mu^0) = 0.$$

Therefore, we can evaluate the KL divergence between $\mathcal{Q}^*$ and $\mathcal{Q}^0(\tau|\mathbf{x}_T)\mu(\mathbf{x}_T)$ as

$$D_{\mathrm{KL}}(\mathcal{Q}^*(\tau)\|\mathcal{Q}^0(\tau|\mathbf{x}_T)\mu(\mathbf{x}_T)) = \mathbb{E}_{\tau\sim\mathcal{Q}^*}[\int_0^T \frac{1}{2}\left\|\mathbf{u}^*\right\|^2 dt + \Psi(\mathbf{x}_T)] = V_0(\bar{\mathbf{x}}_0) = 0.$$

The first equality is based on eq (13). Next, we show that $\mathbf{x}^*_T \sim \mu$. The above equations imply $\mathcal{Q}^*_{0,T}(\bar{\mathbf{x}}_0, y)dy = \exp\left(-\Psi(y)\right)\mu^0(dy)$. It follows that

$$\mathbb{P}[\mathbf{x}^*_T \in A] = \int_A \mathcal{Q}^*_{0,T}(\bar{\mathbf{x}}_0, y)dy = \int_A \exp(-\Psi(y))\mu^0(dy) = \mu(A).$$

B PROOF OF IMPORTANCE WEIGHTS

By definition

$$w^u(\tau) = \frac{d\mathcal{Q}^u(\tau)}{d\mathcal{Q}^*(\tau)} = \frac{d\mathcal{Q}^u(\tau)}{d\mathcal{Q}^0(\tau)}\frac{d\mathcal{Q}^0(\tau)}{d\mathcal{Q}^*(\tau)}. \tag{23}$$

Plugging eq (11) and eq (22) into the above we obtain

$$w^u(\tau) = \exp(\int_0^T \frac{1}{2}\left\|\mathbf{u}_t\right\|^2 dt + \mathbf{u}_t' d\mathbf{w}_t + \log\frac{\mu^0(\mathbf{x}_T)}{\mu(\mathbf{x}_T)}) = \exp(\int_0^T \frac{1}{2}\left\|\mathbf{u}_t\right\|^2 dt + \mathbf{u}_t' d\mathbf{w}_t + \Psi(\mathbf{x}_T)).$$

C PROOF OF THEOREM 2

C.1 LIPCHITZ CONDITON AND PRELIMINARY LEMMA

To ease the burden of notations, we assume $\mathbf{x}_0 \sim \delta_0$. Our conclusion and proof can be generalized to other Dirac distributions easily. We start by assuming some conditions on the Lipschitz of optimal policy $\mathbf{u}^*$. It promises the existence of a unique strong solution with $\mathbf{x}_T \sim \mu$. The conditions and properties are studied in Schrödinger-Föllmer process, which dates back to the Schrödinger problem. For proof of existence of a unique strong solution and detailed discussion, we refer the reader to Dai Pra (1991); **?**); Eldan et al. (2020); Huang et al. (2021).

Condition 1.

$$||\mathbf{u}^*_t(\mathbf{x})||_2^2 \le C_0(1 + \left\|\mathbf{x}\right\|^2) \tag{24}$$

and

$$\left\|\mathbf{u}^*_{t_1}(\mathbf{x}_1) - \mathbf{u}^*_{t_2}(\mathbf{x}_2)\right\| \le C_1(\left\|\mathbf{x}_1 - \mathbf{x}_2\right\| + |t_1 - t_2|^{\frac{1}{2}}). \tag{25}$$

We introduce the following lemma before stating the proof for Theorem 2.

Lemma 4.2. *(Huang et al., 2021, Lemma A.2.) Assume Condition 1 holds, then the following inequality holds for* $\mathbf{x}_t$ *generated from* $\mathbf{u}^*$*,*

$$\mathbb{E}[\left\|\mathbf{x}_{t_2} - \mathbf{x}_{t_1}\right\|^2] \le 4C_0\exp(2C_0)(C_0 + d)(t_2 - t_1)^2 + 2C_0(t_2 - t_1)^2 + 2d|t_2 - t_1|,\ t_1, t_2 \in [0, T]. \tag{26}$$

C.2 PROOF OF THEOREM 2

We denote by $\mathbf{x}^*_{(0:T)}$ the trajectory controlled by the optimal policy $\mathbf{u}^*$, and by $\{\mathbf{x}_{t_0}, \mathbf{x}_{t_1}, \cdots, \mathbf{x}_{t_N}\}$ the discrete time process with sub-optimal policy $\mathbf{u}$ over discrete time $\{t_k\}$ such that $t_0 = 0, t_N = T$ and $t_k - t_{k-1} = \Delta t$. The process $\{\mathbf{x}_{t_k}\}$ can be extended to continuous time setting as

$$\mathbf{x}_{t_k} = \mathbf{x}_{t_{k-1}} + \int_{t_{k-1}}^{t_k} \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})ds + d\mathbf{w}_s.$$

The key of our proof is the bound of $\left\|\mathbf{x}_{t_k} - \mathbf{x}^*_{t_k}\right\|^2$.

$$\begin{aligned}
\left\|\mathbf{x}_{t_k} - \mathbf{x}^*_{t_k}\right\|^2 &= \left\|\mathbf{x}_{t_{k-1}} + \int_{t_{k-1}}^{t_k} \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})ds + d\mathbf{w}_s - [\mathbf{x}^*_{t_{k-1}} + \int_{t_{k-1}}^{t_k} \mathbf{u}^*_s(\mathbf{x}^*_s)ds + d\mathbf{w}_s]\right\|^2 \\
&\leq \left\|\mathbf{x}_{t_{k-1}} - \mathbf{x}^*_{t_{k-1}}\right\|^2 + \left\|\int_{t_{k-1}}^{t_k} [\mathbf{u}^*_s(\mathbf{x}^*_s) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})]ds\right\|^2 \\
&\quad + 2\left\|\mathbf{x}_{t_{k-1}} - \mathbf{x}^*_{t_{k-1}}\right\| \left\|\int_{t_{k-1}}^{t_k} [\mathbf{u}^*_s(\mathbf{x}^*_s) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})]ds\right\| \\
&\leq \left\|\mathbf{x}_{t_{k-1}} - \mathbf{x}^*_{t_{k-1}}\right\|^2 + (\int_{t_{k-1}}^{t_k} \left\|\mathbf{u}^*_s(\mathbf{x}^*_s) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\| ds)^2 \\
&\quad + 2\left\|\mathbf{x}_{t_{k-1}} - \mathbf{x}^*_{t_{k-1}}\right\| (\int_{t_{k-1}}^{t_k} \left\|\mathbf{u}^*_s(\mathbf{x}^*_s) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\| ds) \\
&\leq (1+\alpha)\left\|\mathbf{x}_{t_{k-1}} - \mathbf{x}^*_{t_{k-1}}\right\|^2 + (1+\frac{1}{\alpha})(\int_{t_{k-1}}^{t_k} \left\|\mathbf{u}^*_s(\mathbf{x}^*_s) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\| ds)^2 \\
&\leq (1+\alpha)\left\|\mathbf{x}_{t_{k-1}} - \mathbf{x}^*_{t_{k-1}}\right\|^2 \\
&\quad + (1+\frac{1}{\alpha})(t_k - t_{k-1})(\int_{t_{k-1}}^{t_k} \left\|\mathbf{u}^*_s(\mathbf{x}^*_s) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 ds), \qquad (27)
\end{aligned}$$

where the first and second inequality is based on the triangle inequality, the third inequality is based $2ab \leq \alpha a^2 + \frac{1}{\alpha}b^2$ for any $\alpha > 0$, and the forth inequality is based on the Cauchy-Schwarz inequality.

In the following we bound the second term in eq (27) as

$$\begin{aligned}
&\left\|\mathbf{u}^*_s(\mathbf{x}^*_s) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 \\
&= \left\|\mathbf{u}^*_s(\mathbf{x}^*_s) - \mathbf{u}^*_{t_{k-1}}(\mathbf{x}_{t_{k-1}}) + \mathbf{u}^*_{t_{k-1}}(\mathbf{x}_{t_{k-1}}) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 \\
&\leq (1+\beta)\left\|\mathbf{u}^*_s(\mathbf{x}^*_s) - \mathbf{u}^*_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 + (1+\frac{1}{\beta})\left\|\mathbf{u}^*_{t_{k-1}}(\mathbf{x}_{t_{k-1}}) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 \\
&\leq 2C_1^2(1+\beta)[\left\|\mathbf{x}^*_s - \mathbf{x}_{t_{k-1}}\right\|^2 + |s - t_{k-1}|] + (1+\frac{1}{\beta})\left\|\mathbf{u}^*_{t_{k-1}}(\mathbf{x}_{t_{k-1}}) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2,
\end{aligned}$$

where the first inequality uses $2ab \leq \beta a^2 + \frac{1}{\beta}b^2$ for an arbitrary $\beta > 0$ and the second one is based on eq (25). It follows that

$$\begin{aligned}
&\int_{t_{k-1}}^{t_k} \left\|\mathbf{u}^*_s(\mathbf{x}^*_s) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 ds \\
&\leq \int_{t_{k-1}}^{t_k} 2C_1^2(1+\beta)\left\|\mathbf{x}^*_s - \mathbf{x}_{t_{k-1}}\right\|^2 ds + \int_{t_{k-1}}^{t_k} 2C_1^2(1+\beta)(s - t_{k-1})ds \\
&\quad + \int_{t_{k-1}}^{t_k} (1+\frac{1}{\beta})\left\|\mathbf{u}^*_{t_{k-1}}(\mathbf{x}_{t_{k-1}}) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 ds.
\end{aligned}$$

Thus, for stepsize $\Delta t = t_k - t_{k-1}$, we establish

$$\begin{aligned}
&\int_{t_{k-1}}^{t_k} \left\|\mathbf{u}_s^*(\mathbf{x}_s^*) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 ds \\
&\leq \int_{t_{k-1}}^{t_k} 2C_1^2(1+\beta)\left\|\mathbf{x}_s^* - \mathbf{x}_{t_{k-1}}\right\|^2 ds \\
&\quad + C_1^2(1+\beta)\Delta t^2 + (1+\frac{1}{\beta})\left\|\mathbf{u}_{t_{k-1}}^*(\mathbf{x}_{t_{k-1}}) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 \Delta t. \qquad (28)
\end{aligned}$$

Next we bound $\left\|\mathbf{x}_s^* - \mathbf{x}_{t_{k-1}}\right\|^2$ in eq (28) as

$$\left\|\mathbf{x}_s^* - \mathbf{x}_{t_{k-1}}\right\|^2 \leq (1+\eta)\left\|\mathbf{x}_s^* - \mathbf{x}_{t_{k-1}}^*\right\|^2 + (1+\frac{1}{\eta})\left\|\mathbf{x}_{t_{k-1}}^* - \mathbf{x}_{t_{k-1}}\right\|^2, \qquad (29)$$

where the inequality is based on $(a+b)^2 \leq (1+\eta)a^2 + (1+\frac{1}{\eta})b^2$ for an arbitrary $\eta > 0$.

Plugging eq (29) into eq (28) yields

$$\begin{aligned}
&\int_{t_{k-1}}^{t_k} \left\|\mathbf{u}_s^*(\mathbf{x}_s^*) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 ds \\
&\leq (1+\frac{1}{\beta})\left\|\mathbf{u}_{t_{k-1}}^*(\mathbf{x}_{t_{k-1}}) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 \Delta t + C_1^2(1+\beta)\Delta t^2 \\
&\quad + 2C_1^2(1+\beta)(1+\frac{1}{\eta})\left\|\mathbf{x}_{t_{k-1}}^* - \mathbf{x}_{t_{k-1}}\right\|^2 \Delta t + 2C_1^2(1+\beta)(1+\eta)\int_{t_{k-1}}^{t_k}\left\|\mathbf{x}_s^* - \mathbf{x}_{t_{k-1}}^*\right\|^2 ds. \qquad (30)
\end{aligned}$$

Plugging eq (28) and (30) into eq (27) yields

$$\begin{aligned}
LHS \leq &[1+\alpha+2C_1^2(1+\frac{1}{\alpha})(1+\beta)(1+\frac{1}{\eta})\Delta t^2]\left\|\mathbf{x}_{t_{k-1}}^* - \mathbf{x}_{t_{k-1}}\right\|^2 \\
&+ 2C_1^2(1+\frac{1}{\alpha})(1+\beta)(1+\eta)\Delta t\int_{t_{k-1}}^{t_k}\left\|\mathbf{x}_s^* - \mathbf{x}_{t_{k-1}}^*\right\|^2 ds \\
&+ (1+\frac{1}{\alpha})(1+\frac{1}{\beta})\left\|\mathbf{u}_{t_{k-1}}^*(\mathbf{x}_{t_{k-1}}^*) - \mathbf{u}_{t_{k-1}}(\mathbf{x}_{t_{k-1}})\right\|^2 \Delta t^2 + (1+\frac{1}{\alpha})C_1^2(1+\beta)\Delta t^3. \qquad (31)
\end{aligned}$$

Invoking lemma 4.2, we obtain

$$\mathbb{E}[\int_{t_{k-1}}^{t_k}\left\|\mathbf{x}_s^* - \mathbf{x}_{t_{k-1}}^*\right\|^2 ds] \leq 4C_0\exp(2C_0)(C_0+d)\Delta t^3 + 2C_0\Delta t^3 + 2d\Delta t^2.$$

Taking the expectation of eq (31), in view of the above and the assumption on control, we establish

$$\mathbb{E}[\left\|\mathbf{x}_{t_k} - \mathbf{x}_{t_k}^*\right\|^2] \leq C_3\mathbb{E}[\left\|\mathbf{x}_{t_{k-1}} - \mathbf{x}_{t_{k-1}}^*\right\|^2] + C_4,$$

where $C_3 = [1+\alpha+2C_1^2(1+\frac{1}{\alpha})(1+\beta)(1+\frac{1}{\eta})\Delta t^2]$, and

$$\begin{aligned}
C_4 = &(1+\frac{1}{\alpha})(1+\frac{1}{\beta})d\epsilon\Delta t^2 + (1+\frac{1}{\alpha})C_1^2(1+\beta)\Delta t^3 \\
&+ 2C_1^2(1+\frac{1}{\alpha})(1+\beta)(1+\eta)[4C_0\exp(2C_0)(C_0+d)\Delta t^3 + 2C_0\Delta t^3 + 2d\Delta t^2]\Delta t.
\end{aligned}$$

Finally, in view of the fact $\mathbf{x}_0 = \mathbf{x}_0^*$ and fixed step size Δt, we conclude that by the choice $\alpha = C_1\Delta t, \beta = \eta = 1$

$$\mathbb{E}[\|\mathbf{x}_T - \mathbf{x}_T^*\|^2] \leq \frac{C_3^{\frac{T}{\Delta t}} - 1}{C_3 - 1}C_4 = \mathcal{O}(dT(\Delta t + \epsilon)), \qquad (32)$$

where the last inequality based on ignoring the high order terms, $C_3^{\frac{T}{\Delta t}} - 1 \leq \mathcal{O}(T)$ and $\frac{C_4}{C_3 - 1} \leq \mathcal{O}(d(\Delta t + \epsilon))$.

D PROOF OF THEOREM 3

The proof is a natural extension of Corollary 7 in Thijssen & Kappen (2015).

We define random variable

$$S^u(t) = \int_0^t \frac{\|\mathbf{u}_s\|^2}{2} ds + \mathbf{u}_s' d\mathbf{w}_s + \Psi(\mathbf{x}_T), \tag{33}$$

and

$$\Phi(t) = \exp(-S^u(0) + S^u(t)). \tag{34}$$

Lemma 4.3. *(Thijssen & Kappen, 2015, Lemma 4) For any feasible control policy* $\mathbf{u}$ *for stochastic optimal control problem,*

$$\Phi(T)\phi_t(\mathbf{x}_T) - \Phi(t)\phi_t(\mathbf{x}_t) = \int_t^T \Phi(s)\phi_t(\mathbf{x}_s)(\mathbf{u}_s^* - \mathbf{u}_s)' d\mathbf{w}_s. \tag{35}$$

Corollary 1.

$$\phi_t(\mathbf{x}) = \mathbb{E}_{\mathcal{Q}^0}[\exp(-\Psi(\mathbf{x}_T))|\mathbf{x}_t = \mathbf{x}] = \mathbb{E}_{\mathcal{Q}^u}[\exp(-\Psi(\mathbf{x}_T) - \int_t^T \frac{\|\mathbf{u}_s\|^2}{2} ds)|\mathbf{x}_t = \mathbf{x}] \tag{36}$$

Proof. This follows importance sampling with density ratio from eq (11). □

Proof of Theorem 3: We denote the important weight by w^u; note it is a random variable. It follows that

$$w^u = \frac{\exp(-\Psi(\mathbf{x}_T) - \int_0^T (\frac{\|\mathbf{u}\|^2}{2} dt + \mathbf{u}' d\mathbf{w}))}{\mathbb{E}_{\mathcal{Q}^u}[\exp(-\Psi(\mathbf{x}_T) - \int_0^T (\frac{\|\mathbf{u}\|^2}{2} dt + \mathbf{u}' d\mathbf{w}))]} = \frac{\exp(-S^u(t))}{\mathbb{E}_{\mathcal{Q}^u}[\exp(-S^u(t))]}. \tag{37}$$

Dividing the LHS of eq (35) by $\phi_0(\mathbf{x}_0)$ we obtain

$$\frac{\Phi(T)\phi_t(\mathbf{x}_T) - \Phi(t)\phi_t(\mathbf{x}_t)}{\phi_0(\mathbf{x}_0)} = \frac{\Phi(T)\phi_t(\mathbf{x}_T) - \phi_0(\mathbf{x}_0)}{\mathbb{E}_{\mathcal{Q}^u}[\exp(-S^u(0))]} = w^u - \mathbb{E}_{\mathcal{Q}^u}[w^u]. \tag{38}$$

Therefore, the variance of w^u equals

$$\begin{aligned}
\mathbb{E}_{\mathcal{Q}^u}[(w^u - \mathbb{E}_{\mathcal{Q}^u}[w^u])^2] &= \mathbb{E}_{\mathcal{Q}^u}[(\int_0^T \frac{\Phi(s)\phi_s(\mathbf{x}_s)}{\phi_0(\mathbf{x}_0)} (\mathbf{u}_s^* - \mathbf{u}_s)' d\mathbf{w})^2] \\
&= \mathbb{E}_{\mathcal{Q}^u}[\int_0^T \frac{\Phi^2(s)\phi_s^2(\mathbf{x}_s)}{\phi_0^2(\mathbf{x}_0)} (\mathbf{u}_s^* - \mathbf{u}_s)'(\mathbf{u}_s^* - \mathbf{u}_s) ds] \\
&= \mathbb{E}_{\mathcal{Q}^u}[\int_0^T (w^u \phi_s(\mathbf{x}_s) \exp(S^u(s)))^2 (\mathbf{u}_s^* - \mathbf{u}_s)'(\mathbf{u}_s^* - \mathbf{u}_s) ds].
\end{aligned} \tag{39}$$

By Jensen's inequality

$$\phi(s, \mathbf{x}_s)^2 = (\mathbb{E}_{\mathcal{Q}^u}[\exp(-S^u(s))|\mathbf{x}_s])^2 \leq \mathbb{E}_{\mathcal{Q}^u}[\exp(-2S^u(s))|\mathbf{x}_s].$$

Plugging the above inequality into eq (39), we reach the upper bound of variance

$$\mathbb{E}_{\mathcal{Q}^u}[(w^u - \mathbb{E}_{\mathcal{Q}^u}[w^u])^2] \leq \int_0^T \mathbb{E}_{\mathcal{Q}^u}[(\mathbf{u}_s^* - \mathbf{u}_s)'(\mathbf{u}_s^* - \mathbf{u}_s)(w^u)^2] ds. \tag{40}$$

In view of the fact $\text{Var}(w^u) + 1 = \mathbb{E}_{\mathcal{Q}^u}[(w^u)^2]$, we arrive at

$$1 + \mathbb{E}_{\mathcal{Q}^u}[(w^u)^2] \leq \mathbb{E}_{\mathcal{Q}^u}[(w^u)^2] \int_0^T \mathbb{E}_{\mathcal{Q}^u}[(\mathbf{u}_s^* - \mathbf{u}_s)'(\mathbf{u}_s^* - \mathbf{u}_s)] ds.$$

If we consider the near optimal policy such that $\max_{t,\mathbf{x}} \|\mathbf{u}_t(\mathbf{x}) - \mathbf{u}_t^*(\mathbf{x})\|^2 \leq \frac{\epsilon}{T}$, then it follows that

$$\frac{1}{\mathbb{E}_{\mathcal{Q}^u}[(w^u)^2]} \geq 1 - \epsilon. \tag{41}$$

E PROOF OF THEOREM 4

We consider the KL divergence between trajectory distribution resulted from the policy $\mathbf{u}$ and the one from optimal policy $\mathbf{u}^*$:

$$\begin{aligned}
D_{\mathrm{KL}}(\mathcal{Q}^u(\tau)\|\mathcal{Q}^*(\tau)) &= D_{\mathrm{KL}}(\mathcal{Q}^u(\tau)\|\mathcal{Q}^0(\tau)\frac{\mu(\mathbf{x}_T)}{\mu^0(\mathbf{x}_T)}) \\
&= \mathbb{E}_{\tau\sim\mathcal{Q}^u}[\int_0^T \frac{1}{2}\left\|\mathbf{u}_t\right\|^2 dt + \mathbf{u}_t' d\mathbf{w}_t + \Psi(\mathbf{x}_T)] \\
&= \mathbb{E}_{\tau\sim\mathcal{Q}^u}[\int_0^T \frac{1}{2}\left\|\mathbf{u}_t\right\|^2 dt + \mathbf{u}_t' d\mathbf{w}_t + \hat{\Psi}(\mathbf{x}_T) + \log Z] \\
&= \mathbb{E}_{\tau\sim\mathcal{Q}^u}[\hat{S}^u(\tau) + \log Z] \\
&\geq 0.
\end{aligned}$$

The last inequality is based on the fact $D_{\mathrm{KL}}(\mathcal{Q}^u(\tau)\|\mathcal{Q}^*(\tau)) \geq 0$ and the equality holds only when $\mathbf{u} = \mathbf{u}^*$, pointing to

$$0 = \mathbb{E}_{\tau\sim\mathcal{Q}^*}[\hat{S}^u(\tau) + \log Z].$$

Therefore, we can estimate the normalization constant by

$$Z \geq \exp(-\mathbb{E}_{\tau\sim\mathcal{Q}^u}[\hat{S}^u(\tau)]), \quad Z = \exp(-\mathbb{E}_{\tau\sim\mathcal{Q}^*}[\hat{S}^u(\tau)]). \tag{42}$$

Next we provide an unbiased estimation with sub-optimal policy $\mathbf{u}$ based on importance sampling as

$$\begin{aligned}
1 &= \mathbb{E}_{\tau\sim\mathcal{Q}^u}[\frac{\mathcal{Q}^*(\tau)}{\mathcal{Q}^u(\tau)}] \\
&= \mathbb{E}_{\tau\sim\mathcal{Q}^u}[\exp(\log\frac{\mathcal{Q}^*(\tau)}{\mathcal{Q}^u(\tau)})] \\
&= \mathbb{E}_{\tau\sim\mathcal{Q}^u}[\exp(-\hat{S}^u(\tau) - \log Z)].
\end{aligned}$$

The last equality is based on the fact $\mathbb{E}_{\tau\sim\mathcal{Q}^u}[\frac{\mathcal{Q}^*(\tau)}{\mathcal{Q}^u(\tau)}] = \int_\tau \mathcal{Q}^*(\tau)d\tau = 1$. Hence, we obtain an unbiased estimation of the normalization constant as

$$Z = \mathbb{E}_{\tau\sim\mathcal{Q}^u}[\exp(-\hat{S}^u(\tau))].$$

F EXPERIMENT DETAILS AND DISCUSSIONS

The PIS algorithm is implemented in PyTorch (Paszke et al., 2019). We use Adam optimizer (Kingma & Ba, 2014) in all experiments to learn optimal policy with learning rates 5×10^{-3} and other default hyperparameters. All experiments are trained with 30 epochs and 15000 points datasets. Loss in most experiments plateau after 3 epochs, some event 1 epoch. Experiments are conducted using an NVIDIA A6000 GPU. Training one epoch on 2d example takes around 15 seconds for PIS-NN and 30 seconds for PIS-Grad, 1.6 minutes and 1.8 minutes respectively on Funnel (d=10), and 7 minutes and 9 minutes on LGCP (d=1600).

For all trained PIS and its variants, we use 100 time-discretization steps for the SDEs. Gradient clipping with value 1 is used. A Fourier feature augmentation (Tancik et al., 2020) is employed for time condition. For HMC, we uses 10 iterations of Hamiltonian Monte Carlo with 10 leapfrog steps per iterations, totaling 100 leapfrog steps. For NUTS, we set the maximum depth of the tree built as 5. Note that samples of HMC and NUTS used in our experiments are from separate trajectories instead of from one trajectory at different timestamps. We observed that the latter is more likely to generate samples that concentrate on one single mode. For SMC and AFT, we use 10 transitions with each transition using the same amount computation as HMC. The settings of SMC and AFT follow the official implementation (Arbel et al., 2021) in the released codebase [1].

[1] `https://github.com/deepmind/annealed_flow_transport`

F.1 CHOICE OF $\mathbf{f}, \mathbf{g}$:

As discuss in Section 3.3, we prefer use a linear function for $\mathbf{f}, \mathbf{g}$ to promise a closed-form μ^0. Choice of $\mathbf{f}, \mathbf{g}$ encodes prior knowledge into dynamics without control and $\mathcal{Q}^*$ is determined based on the prior $\mathcal{Q}^0$. Intuitively, the ideal $\mathcal{Q}^0$ should drive particles from ν to $\mathcal{Q}^0(\mathbf{x}_T)$ that is close to μ. In PIS, our training objective is to fit $\mathcal{Q}^*$ with parameterized $\mathcal{Q}^u$. Thus training can be easier and faster if $\mathcal{Q}^*$ and $\mathcal{Q}^0$ are close since we use zero control as initialization for training policy. However, there is no general approach to choose $\mathbf{f}, \mathbf{g}$ such that $\mathcal{Q}^0(\mathbf{x}_T)$ is close to μ and $\mathcal{Q}^0(\mathbf{x}_T)$ has a closed form. In this work, we adopt the general form with $\mathbf{f} = 0, \mathbf{g} = \mathbf{I}$. It is interesting to explore other prior dynamics or data-variant $\mathbf{f}, \mathbf{g}$ in the future work.

F.2 ESTIMATION OF NORMALIZATION CONSTANTS

As discussed in Chopin & Papaspiliopoulos (2020), normalization constants estimation of SMC and its variants AFT can be achieved with incremental importance sampling weights.

In our experiments we treat HMC and NUTS as special cases of SMC with only two different temperature levels. One corresponds to a standard Gaussian distribution and the other one corresponds to the target density. Since the initial distribution ν for SMC and NUTs is chosen as standard Gaussian, we can omit the MCMC steps for it and the total computation efforts required for the specific SMC are for the transitions in HMC and NUTS.

For VI-NF, we use importance sampling

$$\int \hat{\mu}(\mathbf{x}) d\mathbf{x} = \int q(\mathbf{x}) \frac{\hat{\mu}(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} = \mathbb{E}_q[\frac{\hat{\mu}(\mathbf{x})}{q(\mathbf{x})}],$$

where q is the normalized distribution represented by normalizing flows, to provide an unbiased estimation of normalization constants. We use the ELBO in eq (18) for PIS and the unbiased estimation eq (19) for PIS$_{RW}$.

F.3 2 DIMENSIONAL RINGS EXAMPLE

The ring-shape density function

$$\log \hat{\mu} = -\frac{\min((\|\mathbf{x}\| - 1)^2, (\|\mathbf{x}\| - 3)^2, (\|\mathbf{x}\| - 5)^2)}{100}.$$

Consider the special case of gradient informed SDE, which can be viewed as PIS-Grad with a specific group of parameters,

$$d\mathbf{x} = -\frac{d \log \hat{\mu}(\mathbf{x})}{d\mathbf{x}} dt + \sqrt{2} d\mathbf{w}.$$

This is exactly the Langevin dynamics used widely in sampling (MacKay, 2003). As a special case of MCMC, Langevin sampling can generate high quality samples given large enough time interval (MacKay, 2003). From this perspective, PIS-Grad can be viewed as a modulated Langevin dynamics that is adjusted and represented by neural networks.

F.4 BENCHMARKING DATASETS

For mixture of Gaussian, we choose nine centers over the grid $\{-5, 0, 5\} \times \{-5, 0, 5\}$, and each Gaussian has variance 0.3. The small variance is selected deliberately to distinguish the performance of the different methods. We use 2000 samples for estimating the log normalization constant Z. We use the standard MLP network to parameterize the control drift $\mathbf{u}_t(\mathbf{x})$, where the time signal is augmented by Fourier feature using 64 different frequencies. We use 2 layer (64 hidden neurons in each layer) MLP to extract features from the augmented time signal and $\mathbf{x}$ separately, and another 2 layer MLP to map the summation of features to the policy command. We note that all these methods for comparison, including HMC, NUTS, SMC, AFT, can reach reasonably good results given large enough iterations. However, with small finite number of steps, PIS achieves the best performance. We include more results for long-run MCMC methods in Tab 4.

	MG(d=2)			Funnel(d=10)			LGCP(d=1600)		
	B	S	A	B	S	A	B	S	A
AFT-10^3	-0.509	0.24	0.562	-0.249	0.0758	0.261	-3.08	1.59	3.46
SMC-10^3	-0.362	0.293	0.466	-0.338	0.136	0.364	-440	14.7	441
AFT-2×10^3	-0.371	0.477	0.604	-0.249	0.0758	0.261	-1.23	0.826	1.48
SMC-2×10^3	-0.398	0.198	0.444	-0.338	0.136	0.364	-197	5.21	197
AFT-3×10^3	-0.316	0.365	0.483	-0.281	0.0839	0.293	-1.05	0.514	1.17
SMC-3×10^3	-0.137	0.62	0.635	-0.323	0.064	0.329	-109	5.58	109
AFT-5×10^3	-0.194	0.319	0.373	-0.253	0.0397	0.256	-0.949	0.439	1.05
SMC-5×10^3	-0.129	0.246	0.278	-0.298	0.0564	0.303	-37.5	5.04	37.8
AFT-10^4	-0.03	0.515	0.515	-0.194	0.0554	0.202	**-0.827**	**0.356**	**0.901**
SMC-10^4	-0.171	0.446	0.477	-0.239	0.0412	0.243	-6.47	1.95	6.76
PIS-10^2	**-0.021**	**0.03**	**0.037**	**-0.008**	**9e-3**	**0.012**	-1.94	0.91	2.14

Table 4: Long-run MCMC on mode separated mixture of Gaussian (MG), Funel distribution and Log Gaussian Cox Process (LGCP) for estimating log normalization constants. The suffix denotes the total number of discrete-time steps for each method, which equals the number of layers multiply steps per layer. We experiments $10, 20, 30, 50, 100$ layers for annealing and 100 leapfrog steps per layer. As the number of steps increases, the performance of AFT and SMC gradually improves. PIS denotes the PIS$_{RW}$-Grad. *B* and *S* stand for estimation bias and standard deviation among 100 runs and $A^2 = B^2 + S^2$.

In the experiment with Funnel distribution, Arbel et al. (2021) suggests to use a slice sampler kernel for AFT and SMC, which includes 1000 steps of slice sampling per temperature. In Tab 1, we still use HMC for comparing performance with the same number of integral steps. We also include the results with slice sampler in Tab 5. We use 6000 particles for the estimation of log normalization constants. The network architecture of PIS is exactly the same as that in the experiments with mixture of Gaussian.

	Funnel(d=10)		
	B	S	A
AFT-10	0.128	0.376	0.398
SMC-10	-0.193	0.067	0.204
AFT-20	0.0134	0.173	0.174
SMC-20	-0.113	0.0878	0.143
AFT-30	0.074	0.309	0.318
SMC-30	**-0.006**	0.188	0.188
PIS$_{RW}$-Grad	-0.008	**0.009**	**0.012**

Table 5: AFT and SMC with slice sampler kernel. The suffix denotes the number of temperature levels for annealing. 1000 slicing sampling steps are used for each temperature. Though there is no annealing and only 100 steps are used, the performance of PIS is competitive.

In the example with Cox process, the covariance K is chosen as

$$K(u, v) = 1.91 \times \exp(-\frac{\|u - v\|_2}{M\beta}),$$

and the mean vector equals $\log(126) - \sigma^2$ and $\alpha = 1/M^2$. We note that this setting follows Arbel et al. (2021); Møller et al. (1998). Totally 2000 samples are used to evaluate the log normalization constant. We treat the mean of estimation results from 100 repetitions of SMC with 1000 temperatures as ground truth normalization constants. In this experiment, we found clipping gradient from target density function help stabilize and speed up the training of PIS-Grad. This example is the most challenging task among the three. One major reason is the high dimensionality of the task; the PIS needs to find optimal policy $\mathbf{u} : (t,\ \mathbb{R}^d) \to \mathbb{R}^d$ in high dimensional space. In addition, there is no prior information that can be used to shrink the search space, which makes the training of PIS

with MLP more difficult. We use 2000 particles for estimation of log normalization constants. We also include more experiment results in Tab 6.

	LGCP(d=1600)		
	B	S	A
AFT-10^4	**-0.827**	0.356	0.901
PIS$_{RW}$-Grad-1 $\times 10^2$	-1.94	0.91	2.14
PIS$_{RW}$-Grad-5 $\times 10^2$	-1.25	0.57	1.373
PIS$_{RW}$-Grad-10 $\times 10^2$	-0.832	**0.214**	**0.859**

Table 6: PIS with large number of integral step. The suffix number is the total integral steps. For AFT-10^4, we use 100 annealing layers and run 100 leapfrog steps per each annealing layer.

F.5 ALANINE DIPEPTIDE

The setup for target density distribution and the comparison method are adopted from Wu et al. (2020). Following Noé et al. (2019), an invertible transformation between Cartesian coordinates and the internal coordinates is deployed before output the final samples. Then we normalize the coordinates by removing means and dividing them by the standard deviation of train data. To setup the target distribution, we simulate Alanine dipeptide in vacuum using OpenMMTools (Eastman et al., 2017) [2]. Total 10^5 atoms data points are generated as training data. Situation parameters, including time-step and temperature setting are the same as Wu et al. (2020). We refer the reader to official codebase for more details of setting target density function [3].

Following the setup in Wu et al. (2020), we use unweighted samples to compute the metrics. KL divergence of VI-NF on μ is calculated based on ELBO instead of importance sampling as in normalizing constants tasks. We use Metropolis random walk MCMC block for SMC, SNF and AFT and RealNVP blocks for SNF and AFT (Dinh et al., 2016). For a fair comparison, we use PIS-NN instead of PIS-Grad since none of the approaches in this example uses the gradient information. We note that SNF is originally trained with maximizing data likelihood where some empirical samples are assumed to be available. We modify the training objective function by reversing the original KL divergence as in eq (1).

F.6 MORE DETAILS ON SAMPLING IN VARIATIONAL AUTOENCODER LATENT SPACE

Figure 5: Origin data images and their reconstructions from trained vanilla VAE. It can be seen that reconstruction images are smoother compared with the original images.

We use a vanilla VAE architecture to train on binary MNIST data. The encoder uses a standard 3 layer MLP networks with 1024 hidden neurons, and maps an image to the mean and standard deviation of 50 dimension diagonal Normal distribution. The decoder employs 3 layer MLP networks to decode images from latent states. ReLU nonlinearity is used for hidden layers. For training, we use the Adam optimizer with learning rate 5×10^{-4}, batch size 128. With reparameterization trick (Kingma & Welling, 2013) and closed-form KL divergence between approximated normal distribution and standard normalization distribution, we train networks for totally 100 epochs. We show performance of vanilla in Fig 5.

[2] https://github.com/choderalab/openmmtools

[3] https://github.com/noegroup/stochastic_normalizing_flows

We parameterize distribution of decoder $p_\theta(\mathbf{x}|\mathbf{z})$ as

$$\log p_\theta(\mathbf{x}|\mathbf{z}) = \log p(\mathbf{x}|D_\theta(\mathbf{z})) = \mathbf{x} \log D_\theta(\mathbf{z}) + (1 - \mathbf{x}) \log(1 - D_\theta(\mathbf{z})).$$

For PIS, we use the same network and training protocol as that in the experiment for mixture of Gaussian and Funnel distributions. We also use gradient clip to prevent the magnitude of control drift from being too large.

G TECHNICAL DETAILS AND TIPS

Here we provide a list of observations and failure cases we encountered when we trained PIS. We found such evidences through some experiments, though there is no way we are certain the following claims are correct and general for different target densities.

- We notice that smaller T may result in control strategy with large Lipchastiz constants, which is within expectation since large control is required to drive particles to destination with less amount of time. It is reported that it is more difficult to approximate large Lipchastiz functions with neural networks (Jacot et al., 2018; Tancik et al., 2020). We thus recommend to increase T or constraint the magnitude of $\mathbf{u}$ to stablize training when encountering numeric issue or when results are not satisfactory.
- We found batch normalization can help stablize and speed up training, and the choice of nonlinear activation (ReLU and its variants) does not make much difference.
- We also notice that if the control $\mathbf{u}_t(\mathbf{x})$ has large Lipchastiz constants in time dimension, the discretized error would also increase. For calculating the weights based on path integral, we suggest to decrease time stepsize and increase N when the number of integral steps is small and discretization error is high.
- We obtained more stable and smaller training loss when training with Tweedie's formula (Efron, 2011), but we found no obvious improvements on testing the trained sampler or estimating normalization constants.
- Regardless the accuracy and memory advantages of Reversible Heun claimed by *torchsde* (Li et al., 2020a; Kidger et al., 2021) , we found this integration approach is less stable compared with simple Euler integration without adjoint and results in numerical issues occasionally. We recommend readers to use Ito Euler integration when memory permits or conduct training with a small Δt.