Sampling in High-Dimensions using Stochastic Interpolants and Forward-Backward Stochastic Differential Equations

Anand Jerry George **Nicolas Macris**
École Polytechnique Fédérale de Lausanne (EPFL),
Lab for Statistical Mechanics of Inference in Large Systems (SMILS),
CH-1015 Lausanne,
Switzerland

Abstract

We present a class of diffusion-based algorithms to draw samples from high-dimensional probability distributions given their unnormalized densities. Ideally, our methods can transport samples from a Gaussian distribution to a specified target distribution in finite time. Our approach relies on the stochastic interpolants framework to define a time-indexed collection of probability densities that bridge a Gaussian distribution to the target distribution. Subsequently, we derive a diffusion process that obeys the aforementioned probability density at each time instant. Obtaining such a diffusion process involves solving certain Hamilton-Jacobi-Bellman PDEs. We solve these PDEs using the theory of forward-backward stochastic differential equations (FBSDE) together with machine learning-based methods. Through numerical experiments, we demonstrate that our algorithm can effectively draw samples from distributions that conventional methods struggle to handle.

1 INTRODUCTION

Probabilistic modeling is at the heart of modern machine learning, where data is often conceptualized as samples drawn from a probability distribution in a high-dimensional space. In this context, the field of generative modeling, which focuses on drawing new samples from probability distributions given a few samples from the distribution, has witnessed spectacular advancements in recent years, empowering researchers to create synthetic images, videos, and other data with astonishing quality. A concept that has catalyzed significant progress in generative modeling is score-based diffusion modeling [Song and Ermon, 2019, Song et al., 2021, Ho et al., 2020]. This technique incrementally introduces noise to the data until it reaches a state of pure noise, all while learning how to reverse this process effectively.

These new ideas have found applications in the classical sampling problem, wherein the objective is to generate samples from a probability distribution given only its unnormalized density [Zhang and Chen, 2022, Berner et al., 2023, Vargas et al., 2022, Richter et al., 2023, Grenioux et al., 2024, Huang et al., 2023]. In the realm of computational science and statistics, sampling techniques play a pivotal role in various applications ranging from Bayesian inference to machine learning algorithms. The ability to efficiently sample from complex probability distributions underpins the success of numerous computational methodologies, and traditional sampling methods, such as Markov Chain Monte Carlo (MCMC), have been widely employed for this purpose. However, these methods often encounter challenges in high-dimensional spaces or distributions with intricate geometries. Moreover, MCMC methods, which construct a Markov chain with a stationary distribution aligned with the target distribution, often suffer from slow convergence due to long mixing times.

In light of the effectiveness of diffusion processes in generative modeling, there is significant interest in leveraging these processes for sampling. The aim is to find diffusion processes such that starting with the samples from a tractable distribution such as Gaussian, the diffusion process should produce a sample from the desired distribution at the final time. Unlike conventional MCMC methods, diffusion-based approaches do not

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

require tuning of proposal distributions or acceptance probabilities. Furthermore, diffusion-based methods seem to mitigate the slow convergence of MCMC methods.

Traditional score-based generative modeling generates samples from a target distribution by learning how to reverse a forward diffusion process that maps the target distribution to a prior distribution. Ideally, these diffusion processes require an infinite time horizon for convergence. In this work, we design computationally efficient diffusion-based samplers using ideas from the *stochastic interpolants* framework, and *forward-backward SDEs*, to generate exact samples from the target distribution within a finite time.

Problem Statement: We are interested in sampling from a probability distribution π with support on $\mathbb{R}^d$ by constructing diffusion processes which run for a finite time. We are given an unnormalized probability density function $\hat{\pi}$, such that $\pi(x) = \frac{\hat{\pi}(x)}{\int \hat{\pi}(x)dx}$.

Notations: We denote a Gaussian distribution (and its density) with mean μ and covariance Σ by $\mathcal{N}(\mu, \Sigma)$. A uniform random variable in the interval $[0, T]$ is denoted by $U([0, T])$. A derivative of a scalar function f with respect to time $t \in \mathbb{R}_+$ is denoted by $\dot{f}$. A gradient with respect to the space variables $x \in \mathbb{R}^d$ is denoted with ∇, and Δ denotes the Laplacian. We use $\|\cdot\|$ for the Euclidean norm in $\mathbb{R}^d$.

1.1 Our Contributions

There are infinitely many diffusion processes that can drive samples from a prior distribution to a target distribution in a finite time. However, computationally tractable ways to find and simulate diffusions that have distributions other than Dirac distribution at the initial time and given target distribution at finite time are still unknown to the best of our knowledge. We take a step towards this by providing a principled approach for generating such diffusions when the prior distribution is Gaussian. *Specifically, we propose a class of diffusion-based sampling methods that, starting with samples from the Gaussian distribution, can produce samples from a target distribution in finite time given its unnormalized density.* We achieve this by taking an approach based on the stochastic interpolants framework [Albergo et al., 2023]. To the best of our knowledge, this is the first time that stochastic interpolants have been used for classical sampling. Our approach reduces the sampling problem to solving Hamilton-Jacobi-Bellman (HJB) equations; a class of well-studied partial differential equations (PDEs) that arise frequently in the field of optimal control. Traditionally, HJB PDEs are solved by minimizing the corresponding control costs [Zhang and Chen, 2022]. Instead, for important reasons that will become clear later (see the discussion in Section 3.1.1), *our approach uses the theory of forward-backward stochastic differential equations (FBSDE) that connects solutions of HJB PDEs (more generally, nonlinear parabolic PDEs) to solutions of a certain set of stochastic differential equations called FBSDEs.* Moreover, we solve these FBSDEs using machine learning-based methods [Han et al., 2018, E et al., 2022, Raissi, 2018]. One of the advantages of our methods is that they allow solving HJB PDEs without the need to compute computationally expensive Neural SDE gradients [Zhang and Chen, 2022, Berner et al., 2023]. The techniques that we develop to solve HJB PDEs using FBSDEs can be of independent interest.

1.2 Our Techniques

We draw inspiration from the stochastic interpolants framework, which begins with a family of time-indexed random variables defining the density of the diffusion process at each time instant t, and then explores methods to realize such a diffusion process. Such a collection of random variables that have desired probability distribution at the time boundaries are known as stochastic interpolants [Albergo et al., 2023]. To illustrate our techniques, in this section we restrict our focus to one of our methods based on *half interpolants*. A half interpolant is a collection of random variables $\{x_t\}_{t\in[0,T]}$ given by $x_t = g(t)x^* + r(t)z$, for $0 \le t \le T$, where $g, r : [0, T] \to \mathbb{R}_+$ are functions such that $\frac{g}{r}$ is a non-decreasing. Furthermore, g satisfies the boundary condition $g(0) = 0$. Here, $x^* \sim \nu$ and $z \sim \mathcal{N}(0, I_d)$ are independent random variables.

Our aim is to implement a diffusion process $\{S_t\}_{t\in[0,T]}$ such that the distribution of S_t is same as x_t for all $t \in [0, T]$ and also enforce the constraint that $x_T, S_T \sim \pi$ (thereby implicitly choosing ν). If realized, such a diffusion process can drive samples $S_0 \sim \mathcal{N}\left(0, r^2(0)I_d\right)$ to $S_T \sim \pi$.

Let $\rho(t, \cdot)$ denote the probability density of x_t and let $s(t, x) := \nabla \log \rho(t, x)$ denote the so-called score function of density ρ. Lemma 1 shows that ρ satisfies a "Fokker-Planck" PDE given by

$$\partial_t \rho - \frac{\varepsilon^2(t)}{2}\Delta\rho + \nabla \cdot \left(\left(b(t, x) + \frac{\varepsilon^2(t)}{2}s(t, x)\right)\rho\right) = 0, \tag{1}$$

with initial condition $\rho(0, \cdot) \equiv \mathcal{N}\left(0, r^2(0)I_d\right)$, where $b(t, x) = \dot{g}(t)\mathbb{E}[\, x^* \mid x_t = x\,] - r(t)\dot{r}(t)s(t, x)$ and $\varepsilon : [0, T] \to \mathbb{R}_+$ is an arbitrary function.

Equation 1 suggests that we can realize a process S_t with density ρ by simulating a stochastic differential

equation (SDE) given by:

$$dS_t = \left(b(t, S_t) + \frac{\varepsilon^2(t)}{2} s(t, S_t) \right) dt + \varepsilon(t) dW_t, \quad (2)$$

with $S_0 \sim \mathcal{N}\left(0, r^2(0) I_d\right)$, where $\{W_t\}_{t \in 0,T}$ is a standard Brownian motion. Therefore, it suffices to have access to functions b and s (responsible for the drift term) to implement a process S_t that has the same marginal distribution as x_t. We will show (Lemma 2) that both b and s can be expressed in terms of $\mathbb{E}[\, x^* \mid x_t = x\,]$. Thus, it is sufficient to learn the function $\mathbb{E}[\, x^* \mid x_t = x\,]$. Towards this, for some $\beta : [0, T] \to \mathbb{R}_+$, we consider a function $u : [0, T] \times \mathbb{R}^d \to \mathbb{R}$ given by

$$\begin{aligned} u(t, x) &= \log \frac{\rho(t, \beta(t)x)}{\psi(t, \beta(t)x)} \quad (3) \\ &= \log \int_{\mathbb{R}^d} \nu(x^*) e^{\frac{\beta(t) g(t)}{r^2(t)} < x, x^* > - \frac{g^2(t)}{2 r^2(t)} \|x^*\|^2} dx^*, \end{aligned}$$

where $\psi(t, \cdot)$ is the density of the isotropic Gaussian with variance $r^2(t)$. Taking the gradient of (3), we note that $\mathbb{E}[\, x^* \mid x_t = x\,] = \frac{r^2(t)}{\beta(t) g(t)} \nabla u(t, \frac{x}{\beta(t)})$. Hence, a feasible way to obtain $\mathbb{E}[\, x^* \mid x_t = x\,]$ is to compute ∇u. A direct calculation (Lemma 3) shows that u satisfies the following HJB equation:

$$\partial_t u + \frac{\sigma^2}{2} \Delta u + \frac{\sigma^2}{2} \|\nabla u\|^2 - \partial_t \log \left(\frac{\beta(t) g(t)}{r^2(t)} \right) x^T \nabla u = 0, \quad (4)$$

where $\sigma^2(t) = 2 \frac{r^2(t)}{\beta^2(t)} \partial_t \log \frac{g(t)}{r(t)}$. The condition that $\frac{g}{r}$ is non-decreasing assures that σ^2 is a positive function. Observe that (4) is a backward Kolmogorov PDE and can be solved given a terminal condition. To satisfy the constraint $x_T \sim \pi$, we want $\rho(T, \cdot) = \pi(\cdot)$, which gives the terminal condition $u(T, x) = \varphi(x) \equiv \log \frac{\pi(\beta(T)x)}{\psi(T, \beta(T)x)}$. The function β is a design parameter, which we can choose such that the coefficients of the PDE are well-defined for $t \in [0, T]$.

There are several ways to obtain the solution u (more importantly ∇u) of HJB PDE (4) under the terminal condition φ. In the optimal control literature, a prominent approach for solving HJB PDEs involves minimizing the sum of control costs and the terminal cost (see Lemma 6). Instead, we exploit the connections between the solutions of non-linear PDEs and solutions to the corresponding FBSDE to solve (4). The remainder of our approach then relies on machine learning-based techniques to solve an FBSDE associated with the PDE (4). Solving the FBSDE gives us access to the function ∇u on an appropriate domain. Once we have the ∇u, subsequently we obtain functions b and s. We then can realize the process S_t using (2). Figure 1 shows sample trajectories of the diffusion process thus obtained when the target distribution is a mixture of Gaussians.

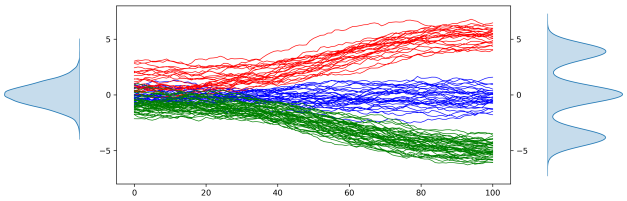

Figure 1: Sample trajectories of diffusion process for sampling from Gaussian mixture.

1.3 Related Works

MCMC methods: For decades, MCMC has stood as the primary method for sampling from unnormalized densities. Techniques that integrate MCMC with annealing and importance sampling methods have consistently yielded superior results in this domain. Among these, Annealed Importance Sampling (AIS) [Neal, 2001] and its Sequential Monte Carlo (SMC) [Del Moral et al., 2006] extensions are widely regarded as state-of-the-art in numerous sampling tasks. Nonetheless, in many practical scenarios, the convergence of these methods can be notably slow. Additionally, analyzing their performance can pose significant challenges, further complicating their application in real-world settings.

Diffusion-based methods: The utilization of diffusions for sampling has been prevalent for a considerable period, with the Langevin diffusion standing out as a prominent example. However, the usage of non-equilibrium dynamics of diffusions for sampling has gained popularity only recently. Noteworthy examples of diffusion-based samplers include Path Integral Sampler (PIS) [Zhang and Chen, 2022], Denoised Diffusion Sampler (DDS) [Vargas et al., 2022], and time reversed Diffusion Sampler (DIS) [Berner et al., 2023], Generalized Bridge Sampler (GBS) [Richter et al., 2023], among others. These samplers leverage advancements in machine learning to address an optimization problem, with the solution being a control function that guides samples from a prior density to samples from the target density. For a detailed comparison of the performance of these methods, we refer the reader to [Blessing et al., 2024]. More recently, the concept of time-reversing diffusion processes has been combined with MCMC techniques to develop samplers that do not require training [Grenioux et al., 2024, Huang et al., 2023].

Stochastic interpolants: The framework of stochastic interpolants was recently introduced in [Albergo et al., 2023]. Despite its conceptual simplicity, this framework provides a unified approach to utilizing diffusions for sampling, particularly in

generative modeling tasks. Stochastic interpolants play a significant role in the derivation of our methods. In particular, our methods strive to learn certain quantities related to the densities defined by the stochastic interpolants. Subsequently, these quantities are used for sampling.

Schrödinger bridge: For arbitrary prior and target distributions, the task of finding a diffusion process that maps one to the other can be formulated as an optimization problem known as a the Schrödinger bridge problem. Concretely, the dynamical formulation of Schrödinger Bridge is the optimization problem $\min_{\mathbb{Q}\in\mathcal{P}(\mathbb{P}_0,\mathbb{P}_T)} D_{\mathrm{KL}}(\mathbb{Q}||\mathbb{P})$, where $\mathcal{P}(\mathbb{P}_0,\mathbb{P}_T)$ is the set of all path measures having density $\mathbb{P}_0$ at $t=0$ and $\mathbb{P}_T$ at $t=T$ and $\mathbb{P}$ is a reference path measure. Solving a Schrödinger bridge problem is generally challenging, as it requires solving a set of coupled partial differential equations (PDEs) [Chen et al., 2021]. However, an instance of the Schrödinger bridge problem that is relatively easier to solve arises when the prior distribution $\mathbb{P}_0$ is a Dirac distribution. In this scenario, the Schrödinger bridge problem reduces to solving a single Hamilton-Jacobi-Bellman (HJB) PDE. The resultant diffusion process is known as a Föllmer process [Föllmer, 1986]. The PIS [Zhang and Chen, 2022] algorithm–a special case of the sampling method we propose–is an implementation of Föllmer process.

2 PRELIMINARIES: RELATION BETWEEN DIFFUSIONS AND PDEs

In this section we introduce the connections between diffusions generated by Stochastic Differential Equations (SDEs) and Partial Differential Equations (PDEs). We first recall the well known relation between diffusions and Fokker-Planck equations, and then we briefly review the connection between FBSDEs and non-linear parabolic PDEs.

Let $\{W_t\}_{t\in[0,T]}$ be a standard Brownian motion. Consider a stochastic process $\{X_t\}_{t\in[0,T]}$ generated by the SDE:

$$dX_t = \mu(t,X_t)dt + \sigma(t)dW_t, \quad X_0 \sim \nu,$$

where $\mu : [0,T]\times\mathbb{R}^d \to \mathbb{R}^d$ is the drift coefficient and $\sigma : [0,T] \to \mathbb{R}$ is the diffusion coefficient. The diffusion coefficient σ can in general take a matrix form and may vary as a function of space. However, for the sake of simplicity in our presentation, we limit our discussion to the scenario where σ is a scalar-valued function solely dependent on t. Let $\rho : [0,T]\times\mathbb{R}^d \to \mathbb{R}_+$ denote the probability density of $\{X_t\}_{t\in[0,T]}$, i.e., $X_t \sim \rho(t,\cdot)$. Then, ρ satisfies a Fokker-Planck equation given by

$$\partial_t\rho - \frac{\sigma^2}{2}\Delta\rho + \nabla\cdot(\mu\rho) = 0, \quad \rho(0,x) = \nu(x).$$

This constitutes a forward Kolmogorov PDE, which is well-defined as an initial value problem. Next, consider the following backward Kolmogorov PDE also known as *quasi-linear parabolic partial differential equation*:

$$\partial_t u + \frac{\sigma^2}{2}\Delta u + \mu^T\nabla u + f(t,x,u,\sigma^T\nabla u) = 0, \quad (5)$$

with terminal condition $u(T,x) = \varphi(x)$, where $u : [0,T]\times\mathbb{R}^d \to \mathbb{R}$ is the solution of the PDE, $\mu : [0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^d \to \mathbb{R}^d$ and $\sigma : [0,T]\to\mathbb{R}$ are coefficient functions, $f : [0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^d \to \mathbb{R}$ is a non-linearity function, and $\varphi : \mathbb{R}^d \to \mathbb{R}$ a given terminal condition. The solution to PDE (5) is related to diffusion processes via the so-called forward-backward stochastic differential equations (FBSDE). Consider the following set of stochastic differential equations:

$$\begin{aligned} dX_t &= \mu(t,X_t,Y_t,Z_t)dt + \sigma(t)dW_t, \\ dY_t &= -f(t,X_t,Y_t,Z_t)dt + Z_t^T dW_t,\ Y_T = \varphi(X_T) \end{aligned} \quad (6)$$

where (X_t,Y_t,Z_t) are stochastic processes adapted to the natural filtration of W_t. Following pioneering work of Bismuth [Bismut, 1973], it was shown by Pardoux and Peng [Pardoux and Peng, 1990, Pardoux, 1998, Pardoux and Tang, 1999] that under certain regularity conditions on μ,σ, and f, quite remarkably, there exists a unique solution (X_t,Y_t,Z_t) to the above set of SDEs. It can be shown that the solution must satisfy $Y_t = u(t,X_t)$ and $Z_t = \sigma(t)\nabla u(t,X_t)$, where u is the solution to PDE (5). Thus we can solve the PDE (5) by solving FBSDE (6).

3 MAIN IDEAS

In this section, we present our methods in detail. We introduce two algorithms for sampling from probability distributions–both based on the stochastic interpolants framework and the FBSDE formulation for solving partial differential equations. First, we review the definition of a linear interpolant (similar to the one-sided linear stochastic interpolant in [Albergo et al., 2023]).

Definition 1 (Linear interpolants)**.** For some $T>0$, let $g,r : [0,T]\to\mathbb{R}_+$ be such that $\frac{g}{r}$ is a non-deceasing function. Let $x^*\sim\nu$ and $z\sim\mathcal{N}(0,I_d)$ be independent random variables. A *Linear interpolant* is a collection of random variables $\{x_t\}_{t\in[0,T]}$ given by

$$x_t = g(t)x^* + r(t)z, \quad 0\le t\le T. \quad (7)$$

We call x_t a *half interpolant* if we include the boundary condition $g(0)=0$. Further, we call x_t a *full interpolant*

if g and r satisfies the boundary conditions $g(0) = r(T) = 0, g(T) = 1$. We don't enforce any condition on $r(0)$.

Figure 3 in Appendix D shows some examples of linear interpolants. Observe that, since g/r is a non-decreasing function, x_t becomes more informative about x^* as time progresses. For a full interpolant, x_T is fully informative about x^* while for a half interpolant, x_T is still a noisy version of x^*. We will demonstrate that we can utilize either of these to develop sampling methods. When using half interpolants for sampling, the distribution ν will be set implicitly by the constraint that we want $x_T \sim \pi$ whereas, for full interpolants, we take ν equal to the target density π. First, we describe a sampling method using half interpolants and later extend it to the case of full interpolants.

3.1 Sampling using Half Interpolants

Consider a half interpolant x_t for $t \in [0, T]$. Note that it is not possible to obtain x_t using (7), since it requires knowledge about ν, and samples x^* from ν. Instead, if we can implement a diffusion process $\{S_t\}_{t\in[0,T]}$ such that the distribution of S_t is same as x_t for all $t \in [0, T]$ and also enforce the condition that $S_T \sim \pi$, then we have a method that drives $S_0 \sim \mathcal{N}\left(0, r^2(0)I_d\right)$ to $S_T \sim \pi$. Constructing such processes is our aim here. Let $\rho(t, \cdot)$ denote the probability density of x_t and $s(t, x) := \nabla \log \rho(t, x)$ denote the score funciton of the density ρ. The score function s holds significant importance in generative modeling through diffusion processes. Notably, the score function is a pivotal component in our formulation as well. First, we present Lemma 1 that characterizes the density function ρ as a solution to certain PDEs [Albergo et al., 2023] (see Lemma 7 in Appendix E for a proof).

Lemma 1. *The probability density function of x_t defined in (7) satisfies a PDE given by*

$$\partial_t \rho + \nabla \cdot (b\rho) = 0, \quad \rho(0, \cdot) = \mathcal{N}\left(0, r^2(0)I_d\right), \tag{8}$$

where $b(t, x) = \dot{g}(t)\mathbb{E}[\, x^* \mid x_t = x\,] - r(t)\dot{r}(t)s(t, x)$. Equivalently, ρ satisfies the following Fokker-Planck equation:

$$\partial_t \rho - \frac{\varepsilon^2(t)}{2}\Delta\rho + \nabla \cdot \left(\left(b(t, x) + \frac{\varepsilon^2(t)}{2}s(t, x)\right)\rho\right) = 0, \tag{9}$$

with initial condition $\rho(0, \cdot) = \mathcal{N}\left(0, r^2(0)I_d\right)$.

Equation 8 is the continuity equation for the density of particles in a velocity field b. This implies that a process S_t with probability density $\rho(t, \cdot)$ can be obtained by solving an ODE with a random initial condition, given by

$$dS_t = b(t, S_t)dt, \quad S_0 \sim \mathcal{N}\left(0, r^2(0)I_d\right). \tag{10}$$

Similarly, equation 9 governs the evolution of the density of particles under a drift and diffusion component. That is, we can realize process S_t by simulating the following SDE:

$$dS_t = \left(b(t, S_t) + \frac{\varepsilon^2(t)}{2}s(t, S_t)\right)dt + \varepsilon(t)dW_t, \tag{11}$$

with $S_0 \sim \mathcal{N}\left(0, r^2(0)I_d\right)$, where $\{W_t\}_{t\in[0,T]}$ is a standard Brownian motion. Hence, it suffices to have access to functions b and s to implement a diffusion process S_t that has the same marginal distribution as x_t. Moreover, Lemma 2 shows that having either $\mathbb{E}[\, x^* \mid x_t = x\,]$ or s is sufficient to derive functions b and s (see Lemma 8 in Appendix E for a proof).

Lemma 2. *Let x_t be a linear or half interpolant. Let $s(t, x) = \nabla\rho(t, x)$ and $b(t, x) = \dot{g}(t)\mathbb{E}[\, x^* \mid x_t = x\,] - r(t)\dot{r}(t)s(t, x)$. Then, we have the following expressions for b and s:*

$$b(t, x) = \begin{cases} \frac{\dot{r}(t)}{r(t)}x + \left(\dot{g}(t) - \frac{g(t)\dot{r}(t)}{r(t)}\right)\mathbb{E}[\, x^* \mid x_t = x\,] \\ \frac{\dot{g}(t)}{g(t)}x + \left(r^2(t)\frac{\dot{g}(t)}{g(t)} - \dot{r}(t)r(t)\right)s(t, x) \end{cases},$$
$$s(t, x) = \frac{g(t)\mathbb{E}[\, x^* \mid x_t = x\,] - x}{r^2(t)}. \tag{12}$$

Relying on the score s to obtain b presents a challenge due to the condition $g(0) = 0$, causing the first term in the second expression for b to diverge as t approaches 0, resulting in numerical instability. Since there are no boundary conditions on r for a half interpolant, a better strategy is to obtain the function $\mathbb{E}[\, x^* \mid x_t = x\,]$ and use the first and third expressions to estimate b and s respectively. Towards this, consider the expression for ρ:

$$\begin{aligned}\rho(t, x) &= \frac{1}{(2\pi r^2(t))^{d/2}}\int_{\mathbb{R}^d} \nu(dx^*)e^{-\frac{\|x - g(t)x^*\|^2}{2r^2(t)}} \\ &= \psi(t, x)\int_{\mathbb{R}^d} \nu(dx^*)e^{\frac{g(t)}{r^2(t)}<x, x^*> - \frac{g^2(t)}{2r^2(t)}\|x^*\|^2},\end{aligned}$$

where $\psi(t, x) = \frac{1}{(2\pi r^2(t))^{d/2}}e^{-\frac{\|x\|^2}{2r^2(t)}}$. For some $\beta : [0, T] \to \mathbb{R}_+$, we define the function

$$\begin{aligned}u(t, x) &= \log\frac{\rho(t, \beta(t)x)}{\psi(t, \beta(t)x)} \\ &= \log\int_{\mathbb{R}^d} \nu(dx^*)e^{\frac{\beta(t)g(t)}{r^2(t)}<x, x^*> - \frac{g^2(t)}{2r^2(t)}\|x^*\|^2}.\end{aligned} \tag{13}$$

Taking the gradient of u, we see that $\mathbb{E}[\, x^* \mid x_t = x\,] = \frac{r^2(t)}{\beta(t)g(t)}\nabla u(t, \frac{x}{\beta(t)})$. Lemma 3 characterizes u as the solution to an HJB PDE (refer to Lemma 9 in Appendix E for a proof).

Lemma 3. *Let $u : [0,T] \times \mathbb{R}^d :\to \mathbb{R}$ be the function given in (13). Then u satisfies the following Hamilton-Jacobi-Bellman equation*

$$\partial_t u + \frac{\sigma^2}{2}\Delta u + \frac{\sigma^2}{2}\|\nabla u\|^2 + \mu^T \nabla u = 0, \qquad (14)$$

where $\mu(t,x) = -\partial_t \log\left(\frac{\beta(t)g(t)}{r^2(t)}\right) x$ *and* $\sigma^2(t) = 2\frac{r^2(t)}{\beta^2(t)}\left(\frac{\dot{g}(t)}{g(t)} - \frac{\dot{r}(t)}{r(t)}\right) \geq 0$.

Note that (14) is a backward Kolmogorov PDE and can be solved given a terminal condition $u(T,\cdot)$. To constrain that $X_T \sim \pi$, we want $\rho(T,\cdot) = \pi(\cdot)$, which together with (13) yields

$$u(T,x) = \varphi(x) \equiv \log \frac{\pi(\beta(T)x)}{\psi(T,\beta(T)x)}. \qquad (15)$$

Note that this condition also implicitly determines ν. Here, for the sake of simplicity in presentation, we assumed that π is a normalized density. However, it is important to note that it can be replaced by an unnormalized density $\hat{\pi}$. This would merely shift the solution u by a constant, which would not impact the sampling algorithm, as only ∇u is required in the sampling phase. The function β is a design parameter, which we can choose such that the coefficients of the PDE are well defined for $t \in [0,T]$. Note that simply choosing $\beta(t) = 1$ can cause μ and σ to diverge as t goes to 0. One possible choice for β is $\frac{r}{g}$, which makes the coefficients well defined for all $t \in [0,T]$.

We emphasize that our aim is to obtain the function $\mathbb{E}[\, x^* \mid x_t = x\,] = \frac{r^2(t)}{\beta(t)g(t)}\nabla u(t, \frac{x}{\beta(t)})$ along the sample paths S_t given by our sampling equation (11). A prominent strategy in optimal-control literature to obtain ∇u is to formulate an optimization problem whose solution is ∇u (see discussion in Appendix C). However, for reasons explained below, we pursue an alternative approach to solve PDE (14). Given that (14) is a nonlinear parabolic PDE, as previously established, we can derive its solutions by solving the corresponding FBSDE. An FBSDE corresponding to PDE (14) can be formulated as:

$$\begin{aligned} dX_t &= (\mu(t,X_t) + \sigma(t)Z_t)\, dt + \sigma(t) dW_t, \\ dY_t &= \frac{1}{2}\|Z_t\|^2 dt + Z_t^T dW_t, \end{aligned} \qquad (16)$$

with $X_0 = \xi$ an appropriately chosen initial condition and $Y_T = \varphi(X_T)$. Lemma 4 relates the processes in (16) to the solution of the PDE (14) under the terminal condition (15) (refer to Lemma 10 in Appendix E for a proof).

Lemma 4. *Let $u : [0,T] \times \mathbb{R}^d :\to \mathbb{R}$ be the solution to PDE (14). Then the processes Y_t and Z_t in (16) are given by $Y_t = u(t,X_t)$ and $Z_t = \sigma(t)\nabla u(t,X_t)$.*

3.1.1 Solving the FBSDE

We take a machine learning-based approach to solve the FBSDE (16). In particular, we approximate the solution u to the PDE (14) with a neural network u^θ paramterized by θ. Subsequently, we obtain an approximation to ∇u by taking the gradient of u^θ. These approximations enable us to approximate the processes Y_t and Z_t. Thereafter, we can form a loss function based on the fact that (X_t, Y_t, Z_t) satisfies the SDEs given in (16). A subtle point in solving the above FBSDE (16) is that the distribution of the process X_t differs from that of the sampling process S_t at each time instant t. Therefore, we might not obtain the function ∇u on the desired domain. We overcome this difficulty by noting that FBSDE (16) is a local equation in the sense that it should be satisfied at each time instant t. In particular, we maintain a separate process X_t in order to ensure that we learn u and ∇u on the appropriate domain, and use a penalty term in the loss (Equ. (19)) which enforces constraints arising from FBSDE locally in time. This is a major advantage of FBSDE over optimal control based approaches for solving HJB PDEs. Specifically, for a τ chosen uniformly at randomly from $[0,T]$ we set, $X = X_\tau$, $Y = u^\theta(\tau, X_\tau)$, $Z = \sigma(\tau)\nabla u^\theta(\tau, X_\tau)$, where X_τ is the solution to the following ODE (for $t = \tau$):

$$\frac{dX_t}{dt} = \frac{\dot{r}(t)}{r(t)}X_t + \left(\frac{\dot{g}(t)}{g(t)} - \frac{\dot{r}(t)}{r(t)}\right)\frac{r^2(t)}{\beta(t)}\nabla u^\theta\left(t, \frac{X_t}{\beta(t)}\right), \qquad (17)$$

with $X_0 \sim \mathcal{N}\left(0, r^2(0) I_d\right)$ and we form the following δ-step discretization of the FBSDE as:

$$\begin{aligned} \hat{X}_\delta &= X + (\mu(\tau,X) + \sigma(\tau)Z)\,\delta + \sigma(\tau)\sqrt{\delta}w, \\ \hat{Y}_\delta &= Y + \frac{1}{2}\|Z\|^2\delta + \sqrt{\delta}Z^T w, \end{aligned} \qquad (18)$$

where $w \sim \mathcal{N}(0, I_d)$. The ODE in (17) coincides with the ODE in (10) when ∇u^θ equals ∇u. Therefore, at optimum, we ensure that ∇u is learned over an appropriate domain. With an appropriately choosen $\lambda > 0$, we use the equations (18), the quantity $Y_\delta = u^\theta(\tau + \delta, \hat{X}_\delta)$, and the boundary conditions of the FBSDE (16), to form the loss function as follows:

$$\mathcal{L}(\theta) = \mathbb{E}\left\|\nabla u^\theta(T, X_T) - \nabla\varphi(X_T)\right\|^2 + \lambda\mathbb{E}\left(\hat{Y}_\delta - Y_\delta\right)^2. \qquad (19)$$

The expectations above are over the process X_τ (given τ), the random time τ, and w. Note that since we are more interested in the quantity ∇u than u, we formed the loss function in terms of the gradient of the terminal condition. We find the approximate solution to FBSDE (16) and in turn to the PDE (14) by minimizing $\mathcal{L}(\theta)$ over θ. In practice, we discretize the ODE (17) using the Euler-Maruyama scheme. We observe that

the discretization error in this step does not directly contribute to the training error, as it only affects the precise path of X_t. We present the steps to compute loss in Procedure LossHalfInterpolant in Appendix A.

3.1.2 Sampling

Once we have an estimate of ∇u, we can derive the functions b and s by utilizing equations (12). Substituting $\mathbb{E}[\, x^* \mid x_t = x\,] = \frac{r^2(t)}{\beta(t)g(t)}\nabla u(t, \frac{x}{\beta(t)})$, we get

$$b(t,x) = \frac{\dot r(t)}{r(t)}x + \frac{r^2(t)}{\beta(t)}\left(\frac{\dot g(t)}{g(t)} - \frac{\dot r(t)}{r(t)}\right)\nabla u\left(t, \frac{x}{\beta(t)}\right),$$
$$s(t,x) = \frac{1}{\beta(t)}\nabla u\left(t, \frac{x}{\beta(t)}\right) - \frac{x}{r^2(t)}.$$

Subsequently, we can use either ODE (10) or SDE (11) for sampling. We outline the steps for sampling in Procedure SampleHalfInterpolant in Appendix A.

3.2 Sampling using Full Interpolants

In this section, we introduce a method to sample from probability distributions using a full interpolant. Merely employing the previous method is destined to fail, as the coefficients of the PDE tend towards infinity if we impose the condition $r(T) = 0$ required by the full interpolant. Instead, we propose a two-step approach that involves solving two HJB PDEs; one for $t \in [0, T']$ and the other for $t \in [T', T]$ with $T' \in (0, T)$. The idea is to obtain $\mathbb{E}[\, x^* \mid x_t = x\,]$ for $t \in [0, T']$ and the score s for $t \in [T', T]$. This is due to the observation that it is possible to form well-behaved PDEs for a quantity related to $\mathbb{E}[\, x^* \mid x_t = x\,]$ for $t \in [0, T']$ and similarly, for a quantity related to s for $t \in [T', T]$. Since b can be derived from either $\mathbb{E}[\, x^* \mid x_t = x\,]$ or s, we acquire the functions essential for sampling for all t within the whole interval $[0, T]$.

First, let us consider t in the interval $[T', T]$. For some function $\alpha : [T', T] \to R_+$, let $v(t,x) = \log \rho(t, \alpha(t)x) + d \log g(t)$. Taking the gradient of v, we observe that v is related to the score by $s(t,x) = \frac{1}{\alpha(t)}\nabla v(t, \frac{x}{\alpha(t)})$. Lemma 5 shows that v satisfies an HJB PDE (see Lemma 11 in Appendix E for a proof).

Lemma 5. *Let $v(t,x) = \log \rho(t, \alpha(t)x) + d \log g(t)$. Then v satisfies the following Hamilton-Jacobi-Bellman equation*

$$\partial_t v + \frac{\bar\sigma^2}{2}\Delta v + \frac{\bar\sigma^2}{2}\|\nabla v\|^2 + \bar\mu^T \nabla v = 0, \qquad (20)$$

where $\bar\mu(t,x) = \partial_t \log\left(\frac{g(t)}{\alpha(t)}\right) x$ and $\bar\sigma^2 = 2\frac{r^2(t)}{\alpha^2(t)}\left(\frac{\dot g(t)}{g(t)} - \frac{\dot r(t)}{r(t)}\right)$.

For sampling, we need x_T distributed as π. This provides us with the terminal condition

$$v(T,x) = \varphi(x) \equiv \log \pi(\alpha(T)x) + d \log g(T). \qquad (21)$$

Similar to β in the case of half interpolants, we can select α such that the PDE (20) is well defined for all t in $[T', T]$. Here, simply choosing $\alpha = 1$ also suffices.

For t in $[0, T']$, it is natural to employ the function u used in the half interpolant-based sampler. Therefore, we have to solve the PDE (14), this time however, subjected to a terminal condition dictated by the solution to PDE (20). The terminal condition is given by $\varphi'(x) = \log \frac{\rho(T', \beta(T')x)}{\psi(T', \beta(T')x)} = v\left(T', \frac{\beta(T')x}{\alpha(T')}\right) - \log \psi(T', \beta(T')x) - d \log g(T')$. For completeness, we restate the PDE below:

$$\partial_t u + \frac{\sigma^2}{2}\Delta u + \frac{\sigma^2}{2}\|\nabla u\|^2 + \mu^T \nabla u = 0, \quad u(T', x) = \varphi'(x). \qquad (22)$$

To estimate u and v, we need to solve PDEs (22) and (20) subjected to the terminal conditions φ' and φ respectively. We again rely on solving the associated FBSDEs for solving the PDEs. The details of FBSDEs and the loss function for solving them can be found in Appendix B.

4 NUMERICAL RESULTS

We evaluated our methods on several distributions from the literature, that are considered challenging to sample from. More details of our implementation can be found in Section F. Below, we provide some distributions for which we present our results. Let $\Psi(x; \mu, \Sigma)$ denote the probability density of the d-dimensional Gaussian distribution with mean μ and covariance Σ.

Gaussian mixture model (GMM): The density of a Gaussian mixture model is given by $\pi(x) = \frac{1}{M}\sum_{i=1}^M \Psi(x; \mu_i, \Sigma_i)$. In our experiments, we use $M = 9, \{\mu_i\}_{i=1}^9 = \{-5, 0, 5\}^2$ and $\Sigma_i = 0.3 I_2$ for all i (here $d = 2$).

Neal's Funnel distribution: Probability density of the funnel distribution is given by $\pi(x) = \Psi(x_1; 0, s^2)\Psi(x_2, x_3, \ldots, x_d; 0, e^{x_1} I_{d-1})$. The values of the parameters are $d = 10, s = 3$. Reference [Neal, 2003] introduces this distribution and discusses the behavior of MCMC implementations.

Double Well (DW): The probability density of double well is given by $\pi(x) = \frac{1}{Z}\exp\left(-\sum_{i=1}^w (x_i^2 - \delta)^2 - \frac{1}{2}\sum_{i=w+1}^d x_i^2\right)$.

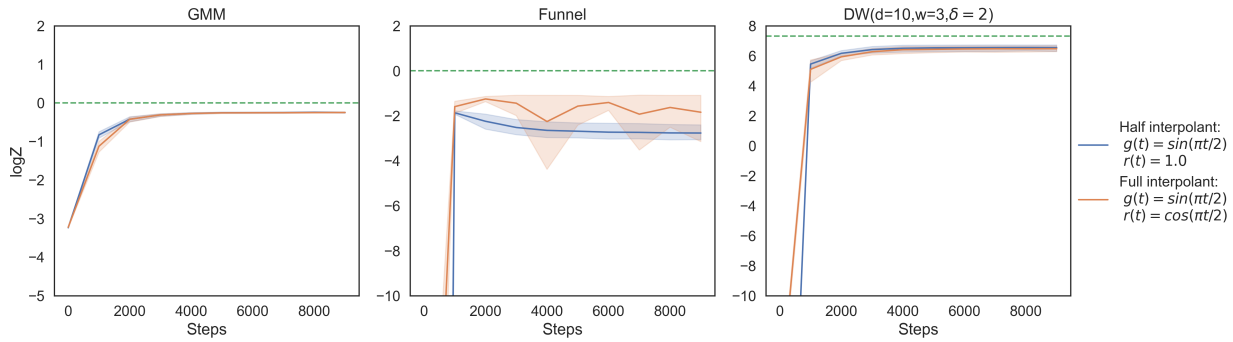

Figure 2: Estimates of $\log Z$ as a function of training steps along with 95% confidence intervals.

Soft Spherical Spin Glass model : We consider a spin glass model whose probability density is given by $\pi(x) \propto \exp\{\frac{\beta}{\sqrt{2d}} x^T A x - \frac{\beta^2}{4d}\|x\|^4 - \frac{1}{2}\|x\|^2\}$, where $A \in \mathbb{R}^{d\times d}$ with i.i.d. Gaussian entries.

We use our sampler for estimating different quantities associated to the above distributions. We estimate the *log normalization constant* ($\log Z$) of the distribution using the method explained in Appendix G. Other quantities we consider are mean L_1-norm, and mean squared L_2-norm, whose estimates are obtained from samples using empirical average. We stress that our estimates are computed without using the importance sampling technique. Hence they reflect the 'true' quality of the samples. In practice, we observe that the full interpolant-based sampling method gives better results compared to the half interpolant-based sampler. We mention that the full interpolant-based sampling method that we presented can in fact work with half interpolant functions as well. Figure 2 shows the estimates of log normalization constant as a function of the training steps, given by the full interpolant-based sampling method. The interpolants used are visualized in Figure 3.

Additional numerical results can be found in Appendix H. In Section H.1, we use our sampler to confirm a predicted phase transition in the soft spherical spin glass model. In Section H.4 we compare our method against another diffusion based sampler called Generalized Bridge Sampler [Richter et al., 2023, Blessing et al., 2024]. In Table 6 we show how different parameters affect the performance of our sampler.

5 DISCUSSION AND FUTURE WORK

We presented a class of diffusion-based algorithms for sampling from unnormalized densities that can obtain samples in finite time. An existing diffusion-based method that can achieve the same is the Path Integral Sampler (PIS) [Zhang and Chen, 2022]. However, PIS is obliged to have a Dirac distribution as its prior. On the other hand, diffusion-based samplers such as DDS [Vargas et al., 2022] and DIS [Berner et al., 2023] are more closely related to score-based generative modeling techniques. They can accommodate non-Dirac distributions as their prior, but ideally require an infinite amount of time for convergence. Our method circumvents these limitations. Our approach which is based on the stochastic interpolants framework permits Gaussian initial distribution on top of producing samples in finite time. We would like to point out that the Föllmer process that the PIS implements is same as the diffusion process that our half interpolant-based method realizes when the interpolant functions take the form $g(t) = f(t)$ and $r(t) = \sqrt{f(t)}$ for some positive function f with $f(0) = 0$.

The training phase of PIS, DDS, and DIS involves computing gradients over the paths generated by an SDE (Neural SDE), which can become computationally expensive as we decrease the discretization step-size of the SDE. However, our method does not suffer from this limitation as we detach the process (17) from the computational graph, and also the discretization step-size of the ODE 17 does not significantly impact the final accuracy. An extension of DIS presented in [Richter et al., 2023] also detaches path gradients from the computational graph, making it computationally less expensive. This is achieved by using log-variance divergence as the loss function.

A limitation of our method currently is that it is not evident if our construction allows for the computation of important weights required to correct the estimates obtained using the samples. This is an important aspect for further investigation. Lastly, we mention that the performance of our sampler can vary significantly based on the values of the hyperparameters used in the implementation. Our primary focus in this work is to present ideas that offer a class of more versatile diffusion-based sampling algorithms, while also offering

a fresh perspective on the existing ones.

Acknowledgments

The work of A. J. G has been supported by Swiss National Science Foundation grant number 200021-204119.

References

[Albergo et al., 2023] Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. (2023). Stochastic Interpolants: A Unifying Framework for Flows and Diffusions. arXiv:2303.08797 [cond-mat].

[Arous et al., 2001] Arous, G. B., Dembo, A., and Guionnet, A. (2001). Aging of spherical spin glasses. *Probability Theory and Related Fields*, 120(1):1–67.

[Barra et al., 2014] Barra, A., Genovese, G., Guerra, F., and Tantari, D. (2014). About a solvable mean field model of a Gaussian spin glass. *Journal of Physics A: Mathematical and Theoretical*, 47(15):155002.

[Berner et al., 2023] Berner, J., Richter, L., and Ullrich, K. (2023). An optimal control perspective on diffusion-based generative modeling. *Transactions on Machine Learning Research*.

[Bismut, 1973] Bismut, J.-M. (1973). Conjugate convex functions in optimal stochastic control. *Journal of Mathematical Analysis and Applications*, 44(2):384–404.

[Blessing et al., 2024] Blessing, D., Jia, X., Esslinger, J., Vargas, F., and Neumann, G. (2024). Beyond ELBOs: A Large-Scale Evaluation of Variational Methods for Sampling. arXiv:2406.07423 [cs, stat].

[Chen et al., 2021] Chen, T., Liu, G.-H., and Theodorou, E. (2021). Likelihood Training of Schrödinger Bridge using Forward-Backward SDEs Theory. In *International Conference on Learning Representations*.

[Del Moral et al., 2006] Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2006.00553.x.

[E et al., 2022] E, W., Han, J., and Jentzen, A. (2022). Algorithms for Solving High Dimensional PDEs: From Nonlinear Monte Carlo to Machine Learning. *Nonlinearity*, 35(1):278–310.

[Föllmer, 1986] Föllmer, H. (1986). Time reversal on Wiener space. In Albeverio, S. A., Blanchard, P., and Streit, L., editors, *Stochastic Processes — Mathematics and Physics*, pages 119–129, Berlin, Heidelberg. Springer.

[Grenioux et al., 2024] Grenioux, L., Noble, M., Gabrié, M., and Durmus, A. O. (2024). Stochastic Localization via Iterative Posterior Sampling.

[Han et al., 2018] Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510. arXiv:1707.02568.

[Hendrycks and Gimpel, 2023] Hendrycks, D. and Gimpel, K. (2023). Gaussian Error Linear Units (GELUs). arXiv:1606.08415 [cs].

[Ho et al., 2020] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.

[Huang et al., 2023] Huang, X., Dong, H., Hao, Y., Ma, Y.-A., and Zhang, T. (2023). Reverse Diffusion Monte Carlo.

[Kingma and Ba, 2017] Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs].

[Kosterlitz et al., 1976] Kosterlitz, J. M., Thouless, D. J., and Jones, R. C. (1976). Spherical Model of a Spin-Glass. *Physical Review Letters*, 36(20):1217–1220. Publisher: American Physical Society.

[Neal, 2001] Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing*, 11(2):125–139.

[Neal, 2003] Neal, R. M. (2003). Slice sampling. *The Annals of Statistics*, 31(3):705–767. Publisher: Institute of Mathematical Statistics.

[Pardoux, 1998] Pardoux, E. (1998). Backward Stochastic Differential Equations and Viscosity Solutions of Systems of Semilinear Parabolic and Elliptic PDEs of Second Order. In Decreusefond, L., Øksendal, B., Gjerde, J., and Üstünel, A. S., editors, *Stochastic Analysis and Related Topics VI*, Progress in Probability, pages 79–127, Boston, MA. Birkhäuser.

[Pardoux and Peng, 1990] Pardoux, E. and Peng, S. G. (1990). Adapted solution of a backward stochastic differential equation. *Systems & Control Letters*, 14(1):55–61.

[Pardoux and Tang, 1999] Pardoux, E. and Tang, S. (1999). Forward-backward stochastic differential equations and quasilinear parabolic PDEs. *Probability Theory and Related Fields*, 114(2):123–150.

[Raissi, 2018] Raissi, M. (2018). Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations. arXiv:1804.07010 [cs, math, stat].

[Richter et al., 2023] Richter, L., Berner, J., and Liu, G.-H. (2023). Improved sampling via learned diffusions. arXiv:2307.01198 [cs, math, stat].

[Song and Ermon, 2019] Song, Y. and Ermon, S. (2019). Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[Song et al., 2021] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-Based Generative Modeling through Stochastic Differential Equations. arXiv:2011.13456 [cs, stat].

[Tancik et al., 2020] Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc.

[Vargas et al., 2022] Vargas, F., Grathwohl, W. S., and Doucet, A. (2022). Denoising Diffusion Samplers. In *International Conference on Learning Representations*.

[Zhang and Chen, 2022] Zhang, Q. and Chen, Y. (2022). Path Integral Sampler: a stochastic control approach for sampling. arXiv:2111.15141 [cs].

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   Some numerical results about the time complexity of the algorithm are described in Appendix H.2. However, a complete analysis of complexity is difficult due to the presence of neural network training.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
   Codes are provided as supplementary materials.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   Proofs are included in Appendix E.
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   Codes are provided as supplementary materials.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   Implementation details are given in Appendix F.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Materials

A Algorithms for Half Interpolant Sampler

We compile the steps required to form the loss for half interpolant sampler into an algorithm. First, we define a sub-routine LossFBSDE that generates the component in the loss due to FBSDE equations. This subroutine will also be used in the full interpolant sampler. Next, we present the procedure LossHalfInterpolant to generate the

Procedure LossFBSDE(t, X, m, δ, f, h)

Input: t, X, m, δ, f, h
Output: L
$w \sim \mathcal{N}(0, I_d)$
$Z \leftarrow h(t)\nabla m(t, X)$
$\hat{Y} \leftarrow m(t, X) + \frac{1}{2}\|Z\|^2\delta + \sqrt{\delta}Z^T w$
$X \leftarrow X + (f(t, X) + h(t)Z)\,\delta + h(t)\sqrt{\delta}w$
$Y \leftarrow m(t + \delta, X)$
$L \leftarrow \left(\hat{Y} - Y\right)^2$
return L

overall loss for the half interpolant sampler. Here, we assume that the batch size equals 1. In practice, we use multiple realizations of X and $\{t_i\}_{i=1}^{N-1}$ to form a single batch. This loss is subsequently minimized to find the optimal θ. Once we have found the optimal θ, we can sample from the target distribution by simulating the SDE

Procedure LossHalfInterpolant

Input: $\theta, T, N, \delta, \lambda$
Output: $\mathcal{L}$
$t_0 \leftarrow 0, t_N \leftarrow T$
$(t_1 \le t_2 \le \cdots \le t_{N-1}) \leftarrow U([0, T])^{N-1}$
$X \sim \mathcal{N}\left(0, r^2(0)I_d\right)$
$\mathcal{L} \leftarrow 0$
for $i = 0, \cdots, N-1$ **do**
  $\mathcal{L} \leftarrow \mathcal{L} + \lambda * \text{LossFBSDE}(t_i, X, u^\theta, \delta, \mu, \sigma)$
  $X \leftarrow X + \left(\frac{\dot{r}(t_i)}{r(t_i)}X + \left(\dot{g}(t_i) - \frac{g(t_i)\dot{r}(t_i)}{r(t_i)}\right)\frac{r^2(t_i)}{\beta(t_i)g(t_i)}\nabla u^\theta\left(t_i, \frac{X}{\beta(t_i)}\right)\right)(t_{i+1} - t_i)$
end
$\mathcal{L} \leftarrow \mathcal{L} + \left\|\nabla u^\theta(T, X) - \nabla\varphi(X)\right\|^2$
return $\mathcal{L}$

given in (11). Procedure SampleHalfInterpolant describes the steps involved in sampling.

Procedure SampleHalfInterpolant

Input: $\theta^*, T, N', \varepsilon$
Output: S
$\Delta \leftarrow \frac{T}{N'}$
$S \sim \mathcal{N}\left(0, r^2(0) I_d\right)$
for $i = 0, \cdots, N'-1$ **do**
  $t = i * \Delta$
  $w \sim \mathcal{N}(0, I_d)$
  $b \leftarrow \frac{\dot{r}(t)}{r(t)} S + \frac{r^2(t)}{\beta(t)} \left(\frac{\dot{g}(t)}{g(t)} - \frac{\dot{r}(t)}{r(t)}\right) \nabla u^{\theta^*}\left(t, \frac{S}{\beta(t)}\right)$
  $s \leftarrow \frac{1}{\beta(t)} \nabla u^{\theta^*}\left(t, \frac{S}{\beta(t)}\right) - \frac{S}{r^2(t)}$
  $S \leftarrow S + \left(b + \frac{\varepsilon^2(t)}{2} s\right) \Delta + \sqrt{\Delta} \varepsilon(t) w$
end
return S

B Details of the Full Interpolant-based Sampler

In Section 3.2, we have seen that in a full interpolant-based sampler, we have to solve two HJB PDEs given by (22) and (20) under respective terminal conditions. Similar to the half interpolant-based sampler, we solve the PDEs by solving associated FBSDEs. An FBSDE corresponding to PDE (22) can be written as

$$\begin{aligned} dX_t &= \left(\mu(t, X_t) + \sigma(t) Z_t\right) dt + \sigma(t) dW_t, \quad X_0 = \xi \\ dY_t &= \frac{1}{2} \|Z_t\|^2 dt + Z_t^T dW_t, \quad Y_{T'} = \varphi'(X_{T'}), \end{aligned} \tag{23}$$

while for PDE (20), an FBSDE is

$$\begin{aligned} dX_t &= \left(\bar{\mu}(t, X_t) + \bar{\sigma}(t) Z_t\right) dt + \bar{\sigma}(t) dW_t, \quad X_{T'} = \xi' \\ dY_t &= \frac{1}{2} \|Z_t\|^2 dt + Z_t^T dW_t, \quad Y_T = \varphi(X_T). \end{aligned} \tag{24}$$

B.1 Solving the FBSDEs

As in the case of half interpolants, we take a machine learning-based approach to solve FBSDEs (24, 23). We approximate the solution to the PDEs (22, 20) u, v with neural networks u^θ and $v^{\theta'}$ paramterized by θ and θ' respectively. Subsequently, we obtain approximations to processes (Y_t, Z_t), which along with the FBSDEs (24,23) help us form a loss function. Specifically, for τ, τ' chosen independently and uniformly at randomly from $[0, T']$ and $[T', T]$ respectively, we form the δ-step discretization of the FBSDE (23) as

$$\begin{aligned} X &= X_\tau, \quad Z = \sigma(\tau) \nabla u^\theta(\tau, X), \\ \hat{Y}_\delta &= u^\theta(\tau, X) + \frac{1}{2} \|Z\|^2 \delta + \sqrt{\delta} Z^T w, \\ \hat{X}_\delta &= X + \left(\mu(\tau, X) + \sigma(\tau) Z\right) \delta + \sigma(\tau) \sqrt{\delta} w \\ Y_\delta &= u^\theta(\tau + \delta, \hat{X}_\delta), \end{aligned}$$

and that of FBSDE (24) as

$$\begin{aligned} X' &= X_{\tau'} \quad Z' = \bar{\sigma}(\tau') \nabla v^{\theta'}(\tau', X') \\ \hat{Y}'_\delta &= v^{\theta'}(\tau', X') + \frac{1}{2} \|Z'\|^2 \delta + \sqrt{\delta} Z'^T w' \\ \hat{X}'_\delta &= X' + \left(\bar{\mu}(\tau', X') + \bar{\sigma}(\tau') Z'\right) \delta + \bar{\sigma}(\tau') \sqrt{\delta} w' \\ Y'_\delta &= v^{\theta'}(\tau' + \delta, \hat{X}'_\delta) \end{aligned}$$

where $w, w' \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d)$ and X_t is the solution to the following ODE:

$$X_0 \sim \mathcal{N}\left(0, r^2(0) I_d\right)$$
$$\frac{dX_t}{dt} = \begin{cases} \frac{\dot{r}(t)}{r(t)} X_t + \left(\dot{g}(t) - \frac{g(t)\dot{r}(t)}{r(t)}\right) \frac{r^2(t)}{\beta(t)g(t)} \nabla u^\theta\left(t, \frac{X_t}{\beta(t)}\right), & 0 \le t \le T' \\ \frac{\dot{g}(t)}{g(t)} X_t + \frac{r(t)}{\alpha(t)} \left(\frac{\dot{g}(t)}{g(t)} r(t) - \dot{r}(t)\right) \nabla v^{\theta'}(t, \frac{X_t}{\alpha(t)}), & T' < t \le T. \end{cases}$$

We then form a loss function as follows:

$$\mathcal{L}(\theta, \theta') = \mathbb{E}\left[\left\|\nabla u^\theta(T', X_{T'}) - \nabla \varphi'(X_{T'})\right\|^2\right] + \lambda \mathbb{E}\left[\left(\hat{Y}_\delta - Y_\delta\right)^2\right] + \mathbb{E}\left[\left\|\nabla v^{\theta'}(T, X_T) - \nabla \varphi(X_T)\right\|^2\right] + \lambda \mathbb{E}\left[\left(\hat{Y}'_\delta - Y'_\delta\right)^2\right]. \quad (25)$$

We find a solution to FBSDEs (24,23) and in turn to the PDEs (22, 20) by minimizing $\mathcal{L}(\theta, \theta')$ over θ, θ'. We compile the steps involved in computing the loss in Procedure LossFullInterpolant.

Procedure LossFullInterpolant

Input: $\theta, T, T', N, \delta, \lambda$
Output: $\mathcal{L}$
$X \sim \mathcal{N}\left(0, r^2(0) I_d\right)$
$\mathcal{L} \leftarrow 0$
$N' = \left\lceil \frac{NT'}{T} \right\rceil$
$t_0 \leftarrow 0, t_{N'} \leftarrow T'$
$(t_1 \le t_2 \le \cdots \le t_{N'-1}) \leftarrow U([0, T'])^{N'-1}$
for $i = 0, \cdots, N' - 1$ **do**
  $\mathcal{L} \leftarrow \mathcal{L} + \lambda * \text{LossFBSDE}(t_i, X, u^\theta, \delta, \mu, \sigma)$
  $X \leftarrow X + \left(\frac{\dot{r}(t_i)}{r(t_i)} X + \left(\dot{g}(t_i) - \frac{g(t_i)\dot{r}(t_i)}{r(t_i)}\right) \frac{r^2(t_i)}{\beta(t_i)g(t_i)} \nabla u^\theta\left(t_i, \frac{X}{\beta(t_i)}\right)\right)(t_{i+1} - t_i)$
end
$\mathcal{L} \leftarrow \mathcal{L} + \left\|\nabla u^\theta(T', X) - \nabla \varphi'(X)\right\|^2$
$t_0 \leftarrow T', t_{N-N'} \leftarrow T$
$(t_1 \le t_2 \le \cdots \le t_{N-N'-1}) \leftarrow U([T', T])^{N-N'-1}$
for $i = 0, \cdots, N - N' - 1$ **do**
  $\mathcal{L} \leftarrow \mathcal{L} + \lambda * \text{LossFBSDE}(t_i, X, v^{\theta'}, \delta, \bar{\mu}, \bar{\sigma})$
  $X \leftarrow X + \left(\frac{\dot{g}(t_i)}{g(t_i)} X + \frac{r(t_i)}{\alpha(t_i)} \left(\frac{\dot{g}(t_i)}{g(t_i)} r(t_i) - \dot{r}(t_i)\right) \nabla v^{\theta'}(t_i, \frac{X}{\alpha(t_i)})\right)(t_{i+1} - t_i)$
end
$\mathcal{L} \leftarrow \mathcal{L} + \left\|\nabla v^{\theta'}(T, X) - \nabla \varphi(X)\right\|^2$
return $\mathcal{L}$

B.1.1 Sampling

Once we have an estimate for $\nabla u, \nabla v$, we can estimate the functions b and s by using equations (12). We have

$$s(t, x) = \begin{cases} \frac{1}{\beta(t)} \nabla u(t, \frac{x}{\beta(t)}) - \frac{x}{r^2(t)}, & 0 \le t < T' \\ \frac{1}{\alpha(t)} \nabla v(t, \frac{x}{\alpha(t)}), & T' \le t \le T', \end{cases}$$

$$b(t, x) = \begin{cases} \frac{\dot{r}(t)}{r(t)} x + \frac{r^2(t)}{\beta(t)g(t)} \left(\dot{g}(t) - \frac{g(t)\dot{r}(t)}{r(t)}\right) \nabla u(t, \frac{x}{\beta(t)}), & 0 \le t < T' \\ \frac{\dot{g}(t)}{g(t)} x + \frac{r(t)}{\alpha(t)} \left(r(t)\frac{\dot{g}(t)}{g(t)} - \dot{r}(t)\right) \nabla v(t, \frac{x}{\alpha(t)}), & T' \le t \le T'. \end{cases}$$

Once we have estimates of b and s, we can sample from π by simulating ODE (10) or SDE (11). In the Procedure SampleFullInterpolant, we present the steps involved in the sampling phase.

Procedure SampleFullInterpolant

Input: $\theta, \theta', T, T', N', \varepsilon$
Output: S
$\Delta \leftarrow \frac{T}{N}$
$S \sim \mathcal{N}\left(0, r^2(0) I_d\right)$
for $i = 0, \cdots, N-1$ **do**
  $t = i * \Delta$
  $w \sim \mathcal{N}(0, I_d)$
  if $t \leq T'$ **then**
    $b \leftarrow \frac{\dot{r}(t)}{r(t)} S + \frac{r^2(t)}{\beta(t)} \left(\frac{\dot{g}(t)}{g(t)} - \frac{\dot{r}(t)}{r(t)} \right) \nabla u^{\theta} \left(t, \frac{S}{\beta(t)} \right)$
    $s \leftarrow \frac{1}{\beta(t)} \nabla u^{\theta} \left(t, \frac{S}{\beta(t)} \right) - \frac{S}{r^2(t)}$
  else
    $b \leftarrow \frac{\dot{g}(t)}{g(t)} S + \frac{r(t)}{\alpha(t)} \left(\frac{\dot{g}(t)}{g(t)} r(t) - \dot{r}(t) \right) \nabla v^{\theta'}(t, \frac{S}{\alpha(t)})$
    $s \leftarrow \frac{1}{\alpha(t)} \nabla v^{\theta'}(t, \frac{S}{\alpha(t)})$
  end
  $S \leftarrow S + \left(b + \frac{\varepsilon^2(t)}{2} s \right) \Delta + \sqrt{\Delta} \varepsilon(t) w$
end

C Optimal Control-based Approach

We remark that ∇u could a priori be obtained as a solution to an optimization problem (see Lemma 6). In particular, let u be the solution to PDE (14) and for an $m \in C^1([0,T] \times \mathbb{R}^d, \mathbb{R}^d)$ let the process X_t be generated by the SDE $dX_t = \left(\sigma^2 m(t, X_t) + \mu(t, X_t) \right) dt + \sigma(t) dW_t$. Then, we have

$$\nabla u = \arg\min_m \mathbb{E}\left[\frac{1}{2} \int_0^T \sigma^2(t) \|m(s, X_s)\|^2 ds - \varphi(X_T) \,\middle|\, X_0 \right]. \tag{26}$$

Typically, m is parameterized by a neural network, and its parameters are learned by solving the minimization problem in (26). However, observe that the domain over which the function ∇u is learned depends on the distribution of the diffusion process X_t for the optimal m. In our case, this presents issues since the diffusion process employed for sampling (11) is different from X_t. Hence, if solved using the optimal control approach, ∇u would be learned over a domain that differs from what is required.

Lemma 6. *Let u be the solution to PDE (14) and let the process X_t be generated by the SDE*

$$dX_t = \left(\sigma^2 m(t, X_t) + \mu(t, X_t) \right) dt + \sigma(t) dW_t. \tag{27}$$

Then, we have

$$u(0, X_0) = -\min_m \mathbb{E}\left[\frac{1}{2} \int_0^T \sigma^2(t) \|m(s, X_s)\|^2 ds - \varphi(X_T) \,\middle|\, X_0 \right],$$
$$\nabla u = \arg\min_m \mathbb{E}\left[\frac{1}{2} \int_0^T \sigma^2(t) \|m(s, X_s)\|^2 ds - \varphi(X_T) \,\middle|\, X_0 \right].$$

Proof. This result is a special example of the general theory of stochastic optimal control, but we present a short

proof for the convenience of the reader. Applying Ito's lemma to $u(t, X_t)$ gives

$$\begin{aligned}
du(t,X_t) &= \partial_t u(t,X_t)dt + \nabla u(t,X_t)^T dX_t + \frac{\sigma^2}{2}\Delta u(t,X_t)dt, \\
&= \left(\partial_t u(t,X_t) + \sigma^2 m^T \nabla u(t,X_t) + \mu^T \nabla u(t,X_t) + \frac{\sigma^2}{2}\Delta u(t,X_t)\right)dt + \sigma \nabla u^T dW_t, \\
&= \left(\sigma^2 m^T \nabla u(t,X_t) - \frac{\sigma^2}{2}\|\nabla u(t,X_t)\|^2\right)dt + \sigma \nabla u^T dW_t, \\
&= -\frac{\sigma^2}{2}\left(\|\nabla u(t,X_t) - m\|^2 - \|m\|^2\right)dt + \sigma \nabla u^T dW_t.
\end{aligned}$$

Integrating from t to T and taking expectation conditioned on X_t gives

$$\begin{aligned}
u(t,X_t) &= \mathbb{E}\left[\frac{\sigma^2}{2}\int_t^T \|\nabla u(s,X_s) - m(s,X_s)\|^2 ds \;\middle|\; X_t\right] \\
&\qquad + \mathbb{E}\left[u(T,X_T) - \frac{\sigma^2}{2}\int_t^T \|m(s,X_s)\|^2 ds \;\middle|\; X_t\right], \\
&\geq \mathbb{E}\left[u(T,X_T) - \frac{\sigma^2}{2}\int_t^T \|m(s,X_s)\|^2 ds \;\middle|\; X_t\right].
\end{aligned}$$

and the equality is attained when $m(t, X_t) = \nabla u(t, X_t)$ a.s. Thus, we get the following variational formulation:

$$\begin{aligned}
u(t,X_t) &= \max_m \mathbb{E}\left[\varphi(X_T) - \frac{\sigma^2}{2}\int_t^T \|m(s,X_s)\|^2 ds \;\middle|\; X_t\right], \\
&= -\min_m \mathbb{E}\left[\frac{\sigma^2}{2}\int_t^T \|m(s,X_s)\|^2 ds - \varphi(X_T) \;\middle|\; X_t\right], \\
\nabla u(t,X_t) &= \arg\min_m \mathbb{E}\left[\frac{\sigma^2}{2}\int_t^T \|m(s,X_s)\|^2 ds - \varphi(X_T) \;\middle|\; X_t\right].
\end{aligned}$$

□

D Interpolants

Figure 3 shows some examples of linear interpolants that we used in our experiments. The first row has half interpolants and the second row contains full interpolants. We observed that keeping $\frac{\dot{r}(t)}{r(t)}$ close to 0 for small values of t is better for training. This is because, at the beginning of training, $\frac{\dot{r}(t)}{r(t)}X_t$ is the drift in (17). Having this negative will decrease the initial exploration of the space, while having it positive might cause stability issues.

E Proofs of Lemmas

Lemma 7. *The probability density function of x_t defined in (7) satisfies a PDE given by*

$$\partial_t \rho + \nabla \cdot (b\rho) = 0, \quad \rho(0,\cdot) \equiv \mathcal{N}\left(0, r^2(0) I_d\right), \quad \rho(T,\cdot) = \pi(\cdot), \tag{28}$$

where $b(t,x) = \dot{g}(t)\mathbb{E}[\, x^* \mid x_t = x\,] - r(t)\dot{r}(t)s(t,x)$,. Equivalently, ρ satisfies the following Fokker-Planck equation:

$$\partial_t \rho - \frac{\varepsilon^2(t)}{2}\Delta\rho + \nabla \cdot \left(\left(b + \frac{\varepsilon^2(t)}{2}s(t,x)\right)\rho\right) = 0, \quad \rho(0,\cdot) \equiv \mathcal{N}\left(0, r^2(0) I_d\right). \tag{29}$$

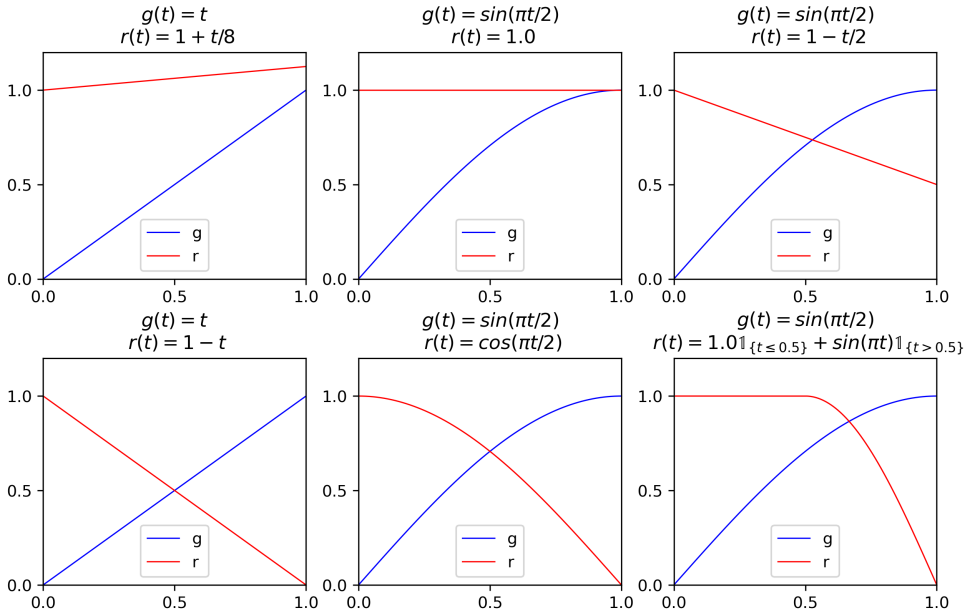

Figure 3: Examples of linear interpolants. Top row: half interpolants. Bottom row: full interpolants.

Proof. Let $\hat{\rho}(t,\cdot)$ be the characteristic funciton of $\rho(t,\cdot)$. That is,

$$\begin{aligned}\hat{\rho}(t,k) &= \int_{\mathbb{R}^d} \rho(t,x) e^{i\langle k,x\rangle} dx, \\ &= \mathbb{E}\Big[e^{i\langle k,x_t\rangle}\Big].\end{aligned}$$

Taking derivative of $\hat{\rho}$ w. r. t. t, we get

$$\begin{aligned}\partial_t \hat{\rho} &= \Big\langle ik, \mathbb{E}\Big[\dot{x}_t e^{i\langle k,x_t\rangle}\Big]\Big\rangle, \\ &= \Big\langle ik, \mathbb{E}\Big[\mathbb{E}[\,\dot{x}_t \mid x_t\,] e^{i\langle k,x_t\rangle}\Big]\Big\rangle.\end{aligned}$$

Taking the Fourier transform of above equation, we get

$$\partial_t \rho = \nabla \cdot (\mathbb{E}[\,\dot{x}_t \mid x_t = x\,]\rho).$$

It remains to show that $b(t,x) = \mathbb{E}[\,\dot{x}_t \mid x_t\,]$. We have

$$\begin{aligned}\mathbb{E}[\,\dot{x}_t \mid x_t\,] &= \dot{g}(t)\mathbb{E}[\,x^* \mid x_t = x\,] + \dot{r}(t)\mathbb{E}[\,z \mid x_t = x\,], \\ &= \dot{g}(t)\mathbb{E}[\,x^* \mid x_t = x\,] - r(t)\dot{r}(t)s(t,x),\end{aligned}$$

where we used the result that $\mathbb{E}[\,z \mid x_t = x\,] = -r(t)s(t,x)$. Fokker-Planck equation (29) can be readily obtained by noting that $\nabla \cdot (s\rho) = \Delta\rho$. □

Lemma 8. *Let x_t be a linear stochastic interpolant. Let $s(t,x) = \nabla\rho(t,x)$ and $b(t,x) = \dot{g}(t)\mathbb{E}[\,x^* \mid x_t = x\,] - r(t)\dot{r}(t)s(t,x)$. Then, we have*

$$\begin{aligned}b(t,x) &= \begin{cases} \frac{\dot{r}(t)}{r(t)}x + \left(\dot{g}(t) - \frac{g(t)\dot{r}(t)}{r(t)}\right)\mathbb{E}[\,x^* \mid x_t = x\,], \\ \frac{\dot{g}(t)}{g(t)}x + \left(r^2(t)\frac{\dot{g}(t)}{g(t)} - \dot{r}(t)r(t)\right)s(t,x), \end{cases} \\ s(t,x) &= \frac{g(t)\mathbb{E}[\,x^* \mid x_t = x\,] - x}{r^2(t)}.\end{aligned} \tag{30}$$

Proof. We have $x_t = g(t)x^* + r(t)z$. Taking expectation conditioned on $x_t = x$, we get

$$x = g(t)\mathbb{E}[\, x^* \mid x_t = x\,] + r(t)\mathbb{E}[\, z \mid x_t = x\,].$$

A direct computation (also known as the Tweedie's formula) yields that $\mathbb{E}[\, z \mid x_t = x\,] = -r(t)s(t,x)$. Hence we can derive $s(t,x)$ from $\mathbb{E}[\, x^* \mid x_t = x\,]$ and vice-versa. This proves the claims in the Lemma.

□

Lemma 9. *Let $u : [0,T] \times \mathbb{R}^d :\to \mathbb{R}$ be the function given in (13). Then u satisfies the following Hamilton-Jacobi-Bellman equation*

$$\partial_t u + \frac{\sigma^2}{2}\Delta u + \frac{\sigma^2}{2}\|\nabla u\|^2 + \mu^T \nabla u = 0, \tag{31}$$

where $\sigma^2(t) = 2\frac{r^2(t)}{\beta^2(t)}\left(\frac{\dot g(t)}{g(t)} - \frac{\dot r(t)}{r(t)}\right)$ and $\mu(t) = -\partial_t \log\left(\frac{\beta(t)g(t)}{r^2(t)}\right)x$.

Proof. From the definition of u in (13), we can directly compute $\partial_t u, \nabla u$ and Δu, which gives

$$\partial_t u = \frac{\int_{\mathbb{R}^d}\nu(dx^*)\left(\partial_t \frac{\beta(t)g(t)}{r^2(t)} < x, x^* > -\partial_t \frac{g^2(t)}{2r^2(t)}\|x^*\|^2\right) e^{\frac{\beta(t)g(t)}{r^2(t)} < x, x^* > - \frac{g^2(t)}{2r^2(t)}\|x^*\|^2}}{\int_{\mathbb{R}^d}\nu(dx^*) e^{\frac{\beta(t)g(t)}{r^2(t)} < x, x^* > - \frac{g^2(t)}{2r^2(t)}\|x^*\|^2}}$$

$$\nabla u = \frac{\int_{\mathbb{R}^d}\nu(dx^*)\left(\frac{\beta(t)g(t)}{r^2(t)}\right) x^* e^{\frac{\beta(t)g(t)}{r^2(t)} < x, x^* > - \frac{g^2(t)}{2r^2(t)}\|x^*\|^2}}{\int_{\mathbb{R}^d}\nu(dx^*) e^{\frac{\beta(t)g(t)}{r^2(t)} < x, x^* > - \frac{g^2(t)}{2r^2(t)}\|x^*\|^2}}$$

$$\nabla^2 u = \frac{\int_{\mathbb{R}^d}\nu(dx^*)\left(\frac{\beta(t)g(t)}{r^2(t)}\right)^2 x^* x^{*T} e^{\frac{\beta(t)g(t)}{r^2(t)} < x, x^* > - \frac{g^2(t)}{2r^2(t)}\|x^*\|^2}}{\int_{\mathbb{R}^d}\nu(dx^*) e^{\frac{\beta(t)g(t)}{r^2(t)} < x, x^* > - \frac{g^2(t)}{2r^2(t)}\|x^*\|^2}} - \nabla u \nabla u^T$$

$$\Delta u = \frac{\int_{\mathbb{R}^d}\nu(dx^*)\left(\frac{\beta(t)g(t)}{r^2(t)}\right)^2 \|x^*\|^2 e^{\frac{\beta(t)g(t)}{r^2(t)} < x, x^* > - \frac{g^2(t)}{2r^2(t)}\|x^*\|^2}}{\int_{\mathbb{R}^d}\nu(dx^*) e^{\frac{\beta(t)g(t)}{r^2(t)} < x, x^* > - \frac{g^2(t)}{2r^2(t)}\|x^*\|^2}} - \|\nabla u\|^2.$$

This shows that u satisfies the PDE (31). □

Lemma 10. *Let $u : [0,T] \times \mathbb{R}^d :\to \mathbb{R}$ be the solution to PDE (14). Then the processes Y_t and Z_t in (16) is given by $Y_t = u(t, X_t)$ and $Z_t = \sigma(t)\nabla u(t, X_t)$.*

Proof. We can rewrite the PDE (14) as

$$\partial_t u + \frac{\sigma^2}{2}\Delta u - \frac{\sigma^2}{2}\|\nabla u\|^2 + (\mu + \sigma\nabla u)^T \nabla u = 0.$$

Thus, appealing to the connection between PDEs and FBSDEs established in the Preliminaries, we identify that $Y_t = u(t, X_t)$ and $Z_t = \sigma(t)\nabla u(t, X_t)$. □

Lemma 11. *Let $v(t,x) = \log \rho(t, \alpha(t)x) + d \log g(t)$. Then v satisfies the following Hamilton-Jacobi-Bellman equation*

$$\partial_t v + \frac{\bar\sigma^2}{2}\Delta v + \frac{\bar\sigma^2}{2}\|\nabla v\|^2 + \partial_t \log\frac{g(t)}{\alpha(t)} x^T \nabla v = 0, \tag{32}$$

where $\bar\sigma^2 = 2\frac{r^2(t)}{\alpha^2(t)}\left(\frac{\dot g(t)}{g(t)} - \frac{\dot r(t)}{r(t)}\right)$.

Proof.

$$\begin{aligned}\partial_t v(t,x) &= (\partial_t \log\rho)(t,\alpha(t)x) + \dot\alpha x^T(\nabla\log\rho)(t,\alpha(t)x) + d\frac{\dot g(t)}{g(t)},\\ &= -(b^T\nabla\log\rho)(t,\alpha x) - (\nabla\cdot b)(t,\alpha x) + \dot\alpha x^T(\nabla\log\rho)(t,\alpha(t)x) + d\frac{\dot g(t)}{g(t)},\\ &= -\left(\frac{\dot g}{g}x + \frac{\bar\sigma^2}{2}\nabla v\right)^T\nabla v - \frac{\bar\sigma^2}{2}\Delta v + \frac{\dot\alpha}{\alpha}x^T\nabla v.\end{aligned}$$

where $\bar\sigma^2 = 2\frac{r^2(t)}{\alpha^2(t)}\left(\frac{\dot g(t)}{g(t)} - \frac{\dot r(t)}{r(t)}\right)$. Simplifying, we get

$$\partial_t v + \frac{\bar\sigma^2}{2}\Delta v + \left(\left(\frac{\dot g}{g} - \frac{\dot\alpha}{\alpha}\right)x + \frac{\bar\sigma^2}{2}\nabla v\right)^T\nabla v = 0. \tag{33}$$

□

F Implementation Details

We provide the details of our implementation here. Our implementation is based on the JAX[1] framework. We rely on the automatic differentiation feature in JAX to compute the gradients of functions required in our algorithms. Table 1 gives the hyperparameters of the sampler that we used in our simulations. All our simulations are performed with $T = 1$. We note that choosing a small T would require the neural network to represent functions with a large Lipshitz constant, thus increasing the complexity of the neural network. We next present the details of the neural networks used to represent the solution to the PDEs (14),(20) and (22).

F.1 Neural networks

In score-based generative modeling, the score function is parameterized directly by a neural network. Our formulation of solving PDEs using FBSDE requires us to have a representation for log-likelihood. Consequently, we utilize energy-based models and obtain the score function by taking the gradient of log-likelihood. However, since we consider unnormalized densities, we use only the score function to obtain the loss. Similar to the approach taken in [Zhang and Chen, 2022, Vargas et al., 2022, Berner et al., 2023], we integrate the terminal condition of PDEs into the architecture of the neural network. In particular, we use a neural network of the following form:

$$\Phi^\theta(t,x) = \Phi_1^{\theta_1}(t,x) + \Phi_2^{\theta_2}(t)\phi(x),$$

where $\theta = \{\theta_1,\theta_2\}$ and ϕ is the terminal condition of the PDE under consideration. We initialize the networks $\Phi_1^{\theta_1}$ with 0 and $\Phi_2^{\theta_2}$ with 1 so that $u^\theta(T,\cdot) = \phi(\cdot)$ in the beginning. Both networks use Fourier Embedding [Tancik et al., 2020] to encode time. The architecture of the neural networks $\Phi_1^{\theta_1}$ and $\Phi_2^{\theta_2}$ can be described as follows:

$$\Phi_1^{\theta_1}(t,x) = L_1\circ\varrho\circ L\circ\varrho\circ L\circ\varrho\Big(L(x) + L\circ\varrho\circ L\circ\text{Fourier}(t)\Big),$$

$$\Phi_2^{\theta_2}(t) = L_1\circ\varrho\circ L\circ\varrho\circ L\circ\varrho\circ L\circ\text{Fourier}(t),$$

where Fourier$(\cdot)$ is the Fourier embedding function with 128 outputs, L is a linear dense layer with 64 outputs, L_1 is a linear dense layer with single output and ϱ is the GELU [Hendrycks and Gimpel, 2023] activation function. We mention that this neural network architecture is similar to the one used in [Berner et al., 2023]. However, the above neural network implements a scalar function that approximates the solution to a PDE, whereas the neural network in [Berner et al., 2023] approximates the gradient of the solution to the PDE.

For a full interpolant-based sampler, as explained earlier, we need to solve two PDEs. This requires two neural networks–one for $t\in[0,T']$ and another for $t\in[T',T]$. The networks we use are as follows:

$$u^\theta(t,x) = \Phi_1^{\theta_1}(t,x) + \Phi_2^{\theta_2}(t)(-\psi(T',x)),$$

$$v^{\theta'}(t,x) = \Phi_1^{\theta'_1}(t,x) + \Phi_2^{\theta'_2}(t)\varphi(x),$$

where ψ is the function that appear in (13) and φ is given in (21.

[1] https://jax.readthedocs.io/en/latest/index.html

Table 1: Hyperparameters

Sampler	
T	1.0
T'	0.5
$\beta(t)$	$\frac{r(t)}{g(t)}$
$\alpha(t)$	1.0
δ	$5 * 10^{-6}$
λ	$2000/\delta$
Training	
Number of steps	10000
Batch Size	128
Initial Learning rate	$5 * 10^{-3}$
Max Grad Norm	1.0
SDE Steps	60
Sampling	
SDE steps	1000
Samples	10000
ε	1.0

F.2 Training

We use the Adam optimizer [Kingma and Ba, 2017] to optimize the weights of the neural networks. We use a piecewise linear scheduling for the learning rate with its value becoming $1/5$th every 2000 training steps. We also do exponential moving averaging of the neural network weights. We detached the process 17 from the computational graph while computing the gradients. Table 1 shows the values of the training-related hyperparameters used in our experiments.

Figure 4 shows the training loss as a function of training steps. The plot on the left is for GMM with $d = 2$, and the plot on the right is for MoG ($d = 100$).

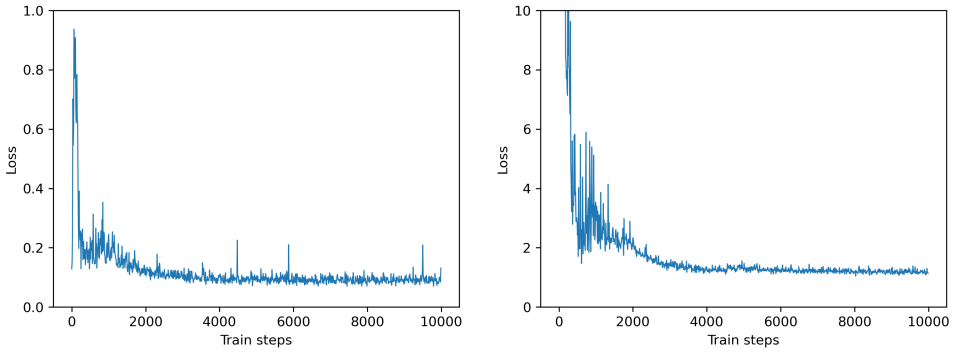

Figure 4: Training phase plot at $d = 2$ and $d = 100$

F.3 Sampling

We sample using procedures given in SampleHalfInterpolant and SampleFullInterpolant. The number of discretization steps we use is 1000. We use a constant diffusion coefficient while sampling. See Table 1.

F.4 Computational complexity

The overall computational complexity of our algorithm can be split into the complexity associated with the training phase and the sampling phase. Evaluating the complexity during the training phase depends on various factors such as the architecture of the neural network and the optimization algorithms employed. Moreover, since training is a one-time procedure, its computational complexity can be amortized given we draw a large number of samples.

In our implementation, the number of parameters in the neural network representing the function scales linearly in the dimension d. During the sampling phase, the dominant computational cost arises from the evaluation of the learned denoiser ∇u^θ (which involves $O(d^2)$ computations.). This leads to a complexity of $O(Nd^2)$, where N is the number of discretization steps in the sampling SDE.

G Estimating the normalization constant

One of the remarkable features of diffusion-based sampling approaches is that they allow for estimating the normalization constant of the given unnormalized density. We describe here the method for estimating the normalization constant for the half interpolant-based sampler. The log normalization constant can be estimated similarly for a full interpolant-based sampler. Let u be the function given in (13). We know u is the solution to PDE (14) under the terminal condition 15. Let X_t be the solution to an SDE given by

$$dX_t = \left(\sigma^2(t)\nabla u(t, X_t) + \mu(t, X_t)\right) dt + \sigma(t) dW_t, \quad X_0 = 0.$$

Applying Itô's lemma to $u(t, X_t)$, we have

$$\begin{aligned} du(t, X_t) &= \partial_t u(t, X_t)dt + \nabla u(t, X_t)^T dX_t + \frac{1}{2}\sigma^2(t)\Delta u(t, X_t)dt \\ &= \left(\partial_t u + \frac{1}{2}\sigma^2(t)\Delta u + \left(\sigma^2(t)\nabla u + \mu(t, X_t)\right)^T \nabla u\right) dt + \sigma(t)\nabla u^T dW_t \\ &= \frac{1}{2}\sigma^2(t)\|\nabla u(t, X_t)\|^2 dt + \sigma(t)\nabla u^T dW_t \end{aligned}$$

Integrating from 0 to T, we get

$$u(0,0) = u(T, X_T) - \frac{1}{2}\int_0^T \sigma^2(t)\|\nabla u(t, X_t)\|^2 dt - \int_0^T \sigma(t)\nabla u^T dW_t \tag{34}$$

Equation (34) gives a viable strategy to estimate the normalization constant. Suppose we want to sample from $\pi = \frac{\hat{\pi}}{Z}$, given only $\hat{\pi}$. From the definition of u, we have $u(T, X_T) = \log \frac{\pi(\beta(T)X_T)}{\psi(T,\beta(T)X_T)} = \log \frac{\hat{\pi}(\beta(T)X_T)}{\psi(T,\beta(T)X_T)} - \log Z$. Moreover, we have $u(0,0) = 0$. Hence, we get

$$0 = -\frac{1}{2}\int_0^T \sigma^2(t)\|\nabla u(t, X_t)\|^2 dt - \int_0^T \sigma(t)\nabla u^T dW_t + \log \frac{\hat{\pi}(\beta(T)X_T)}{\psi(T, \beta(T)X_T)} - \log Z,$$

which after taking expectation gives

$$\log Z = \mathbb{E}\Bigg[-\frac{1}{2}\int_0^T \sigma^2(t)\|\nabla u(t, X_t)\|^2 dt - \int_0^T \sigma(t)\nabla u^T dW_t + \log \hat{\pi}(\beta(T)X_T) - \log(\psi(T, \beta(T)X_T))\Bigg] \tag{35}$$

H Additional Experiments and Results

Figure 5 shows the kernel density estimation (KDE) plots of samples obtained by our methods for 2-dimensional Gaussian mixture distribution and Rings distribution. These plots illustrate that our method can sample from

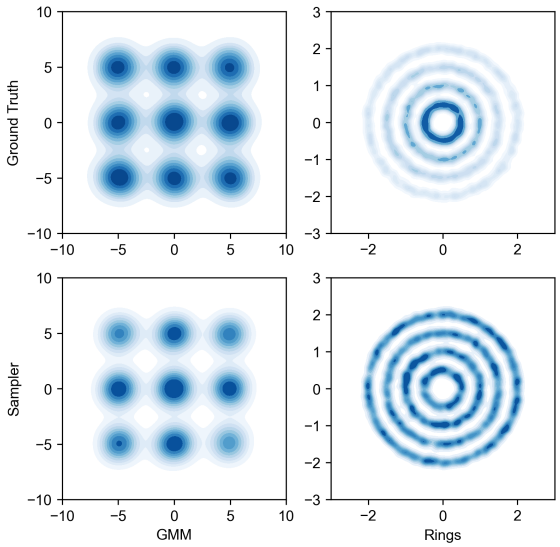

Figure 5: KDE plots of Gaussian mixture, and Rings distributions.

Table 2: Estimates (True values in the paranthesis)

Quantity	Distributions		
	GMM	DW($d = 10, w = 3, \delta = 2$)	DW($d = 20, w = 5, \delta = 3$)
$\mathbb{E}\|X\|_1$	6.04±0.18 (7.19)	9.75±0.04 (9.54)	22.59±0.87 (20.42)
$\mathbb{E}\|X\|^2$	28.36±1.05 (35.26)	13.32±0.08 (12.51)	36.11±2.88 (29.54)

distributions with multiple modes. Figure 6 shows the estimates of $\mathbb{E}|X|_1$ (Mean Absolute) and $\mathbb{E}\|X\|^2$ (Mean Squared) computed from the samples obtained using full interpolant based sampler with different interpolants. Refer to Table 2 for the estimates at the end of the training for the interpolant $g(t) = \sin(\pi t/2), r(t) = \cos(\pi t/2)$. These estimates are without correction using important weights and hence indicate the true quality of the samples.

H.1 Sampling from a statistical physics inspired high-dimensional distribution

Now, we consider a statistical physics *spin glass* model [Arous et al., 2001, Barra et al., 2014]. The precise definition is slightly different in these two works and we follow the incarnation of [Barra et al., 2014]. This model was introduced as a "soft spin" version of the celebrated *spherical* Sherrington-Kirkpatrick model [Kosterlitz et al., 1976]. The free energy can be computed exactly and here serves as a benchmark to assess the experiments.

The probability density of this soft spin spherical spin glass is given by

$$\pi(x) = \frac{1}{Z} \exp\left\{ \frac{\beta}{\sqrt{2d}} \sum_{i,j=1}^{d} A_{ij} x_i x_j - \frac{\beta^2}{4d} \left(\sum_{i=1}^{d} x_i^2 \right)^2 - \frac{1}{2} \sum_{i=1}^{d} x_i^2 \right\},$$

where $A_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d)$. The asymptotic rigorous prediction for the free energy [Barra et al., 2014] is given by

$$\lim_{d\to\infty} \frac{1}{d} \mathbb{E} \log Z = \begin{cases} 0 & \text{for } \beta < 1, \\ -\frac{1}{2}\log\beta + \frac{\beta\bar{q}}{2} + \frac{\beta^2\bar{q}^2}{d}, \quad \bar{q} = \frac{\beta-1}{\beta^2} & \text{for } \beta \geq 1 \end{cases}$$

where the limit is attained almost surely, in other words it is equal to the limit of the expectation with respect to $A_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d)$). The theory predicts that there is a phase transition at $\beta = 1$.

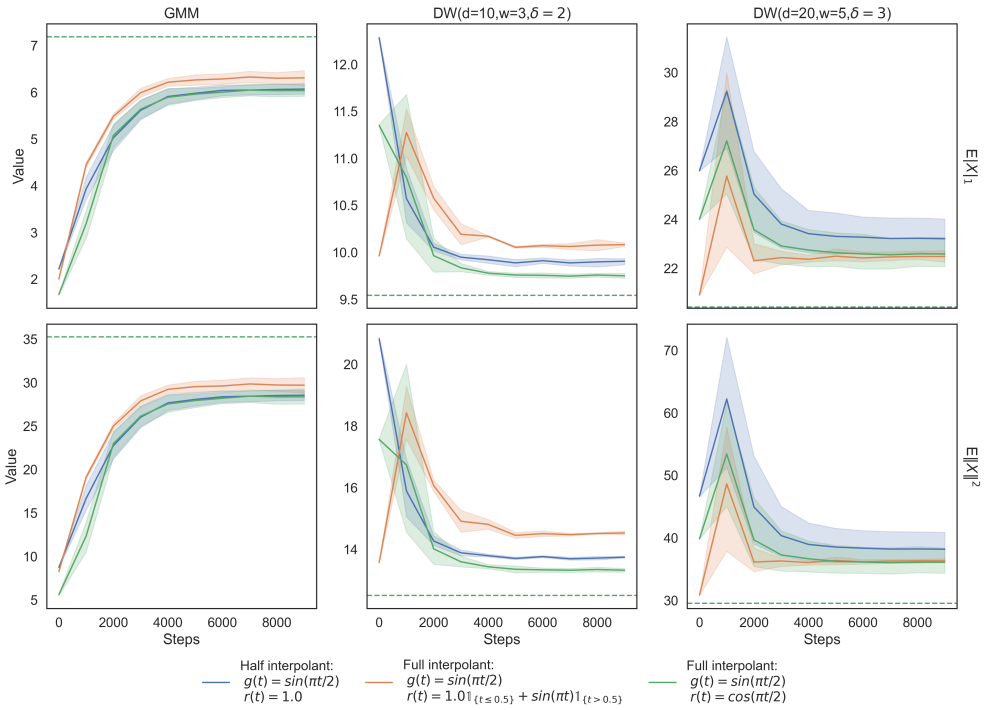

Figure 6: Estimates of $\mathbb{E}|X|_1$ and $\mathbb{E}\|X\|^2$ as a function of training steps.

Table 3: Training and sampling times

Distribution	Training time per step (mS)	Sampling time per discretization step (mS)
GMM	97.7	9.3
Funnel	99.7	10.3
Double well ($d = 10$)	102.7	9.8
Double well ($d = 20$)	105.0	10.8
Spin Glass ($d = 100$)	98.1	10.4

We use our full interpolant-based sampler to estimate $\log Z$ for different values of β. In Figure 7 we compare the obtained estimate of $\frac{1}{d}\log Z$ with the theoretical prediction. The estimates obtained through our sampler are consistent with the phase transition. The experimental points correspond to one instance of teh coupling matrix A_{ij} indicating that already for $d = 100$ the free energy is well concentrated around its average.

H.2 Training and Sampling Times

In this section, we discuss the typical training and sampling time for our sampler. We ran our experiments on an NVIDIA GeForce GTX 1070 (8GB) GPU. In comparison to the conventional MCMC algorithms, a disadvantage of our methods is that we have learnable parameters. However, the training time required can be amortized in time if we are interested in drawing a large number of samples from a specific target distribution. Table 3 shows the training time per gradient descent step required for different distributions. We take a total of 10,000 gradient descent steps during training. Hence, the total time required for training is around 17 minutes for each of the distributions. Table 3 also shows the time required during the sampling phase per discretization step of the SDE. The number of Euler-Maruyama discretization steps we take is 1000, which amounts to a total of 10 seconds for drawing 10,000 samples.

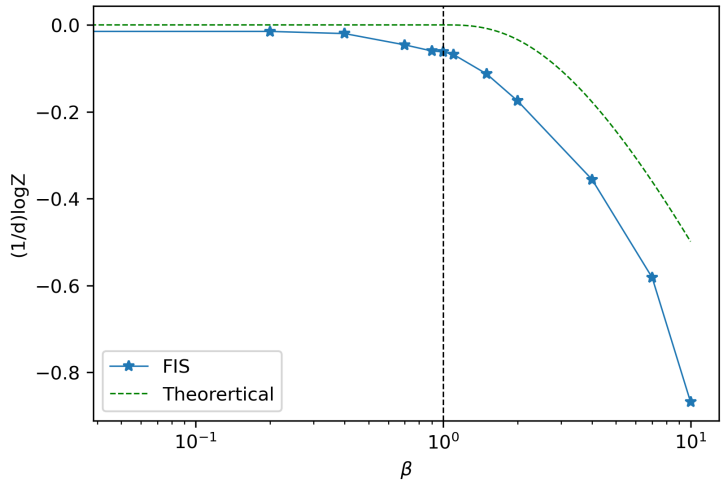

Figure 7: Estimate of $\frac{1}{d}\log Z$ for Gaussian Spin Glass model with $d = 100$.

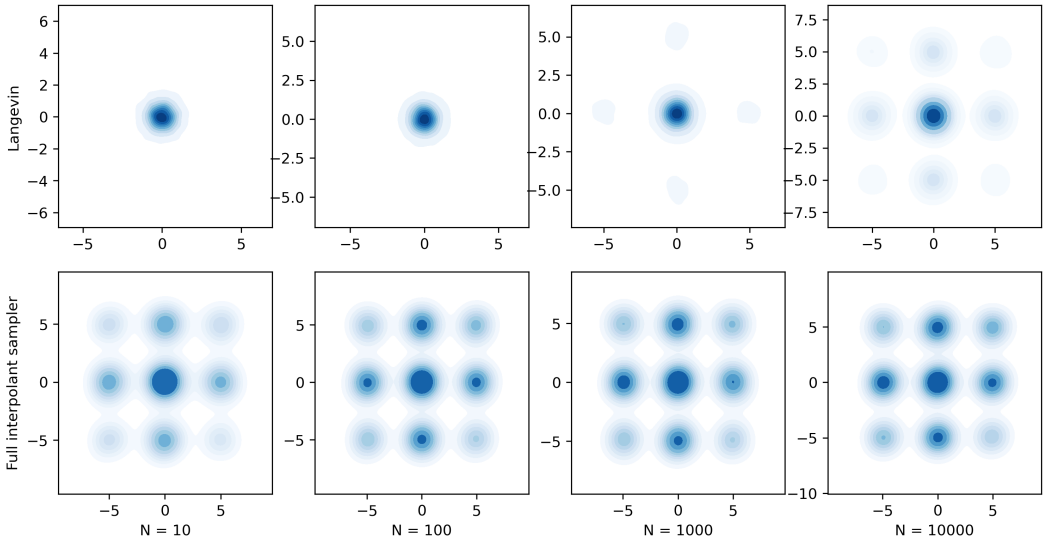

Figure 8: KDE plots of GMM samples generated by Langevin diffusion-based algorithm and full interpolant-based algorithm for different number of discretization steps.

Table 4: Comparison of estimates using Langevin and Full interpolant sampler.

Distribution	Steps	$\mathbb{E}\lvert X\rvert_1$		$\mathbb{E}\lVert X\rVert^2$	
		LMC	FIS	LMC	FIS
GMM	10	1.1	5.5	1.5	25.2
	100	1.1	5.9	1.7	28.0
	1000	1.6	6.0	4.4	28.0
	10000	4.7	6.0	21.3	28.4
DW($d = 10$)	10	8.1	9.7	9.9	13.3
	100	9.0	9.72	11.42	13.2
	1000	9.7	9.5	12.5	13.3
	10000	9.7	9.5	12.5	13.3
DW($d = 20$)	10	16.4	22.0	20.6	22.15
	100	19.5	22.2	27.3	34.6
	1000	20.5	22.2	29.6	34.6
	10000	20.4	22.1	29.4	34.5

H.3 Comparison to Langevin Diffusion for Sampling

To illustrate the benefits of our method, we compare it to the vanilla Langevin diffusion-based sampling algorithm. A Langevin diffusion is given by the solution to the SDE:

$$dS_t = \eta \nabla \log \pi(S_t)dt + \sqrt{2\eta}dW_t.$$

It is well known that the distribution of S_t as t goes to infinity is π. A sampling scheme based on Langevin diffusion is given by its discretization:

$$S_{n+1} = S_n + \nabla \log \pi(S_t)\delta + \sqrt{2\delta}w_{n+1},$$

where $w_i \overset{i.i.d.}{\sim} \mathcal{N}(0, I_d)$. In Figure 8 we compare the KDE plots of samples generated by a Langevin diffusion-based algorithm ($\delta = 0.1$) and a full interpolant-based algorithm as a function of the number of discretization steps for a Gaussian mixture model. In this experiment, we made sure that the time required per discretization step is the same for both algorithms. We observe that the Langevin diffusion requires a large number of steps to explore all the modes of the distribution, while our algorithm explores all the modes even with merely 10 discretization steps. A comparison of estimates of $\mathbb{E}|X|_i$ and $\mathbb{E}\|X\|^2$ of Gaussian mixture and double well distributions obtained through Langevin Monte Carlo (LMC) and Full interpolant sampler (FIS) is given in Table 4.

H.4 Comparison with Generalized Bridge Sampler (GBS)

In Figure 9 we compare the performance of FIS (Full Interpolant Sampler with trigonometric interpolant) and GBS (General Bridge Sampler) [Richter et al., 2023, Blessing et al., 2024] as a function of training steps for two different target distributions and for dimensions $d = 10, 100, 200$. Here, MoG refers to a Mixture of isotropic Gaussian distribution with 10 components with their mean selected uniformly at random from $[-10, 10]^d$ and MoS is a Mixture of Student t-distribution (degree of freedom = 2) with 10 components with their mean selected uniformly at random from $[-10, 10]^d$. We chose to compare our algorithm with GBS because of their similar computational complexity. The implementation of GBS was taken from [Blessing et al., 2024].

Table 5 compares the performance of FIS and GBS at the end of training (10000 iterations). The experimental setting is the same as that of Figure 9. The metrics we use for comparison are and 2-Wasserstein distance (Entropy regularized optimal transport cost) [Blessing et al., 2024]. Based on the observations in Figure 9 and Table 5, it is not possible to definitively conclude that one method is better than the other, as their performance may depend on the specific distribution being considered.

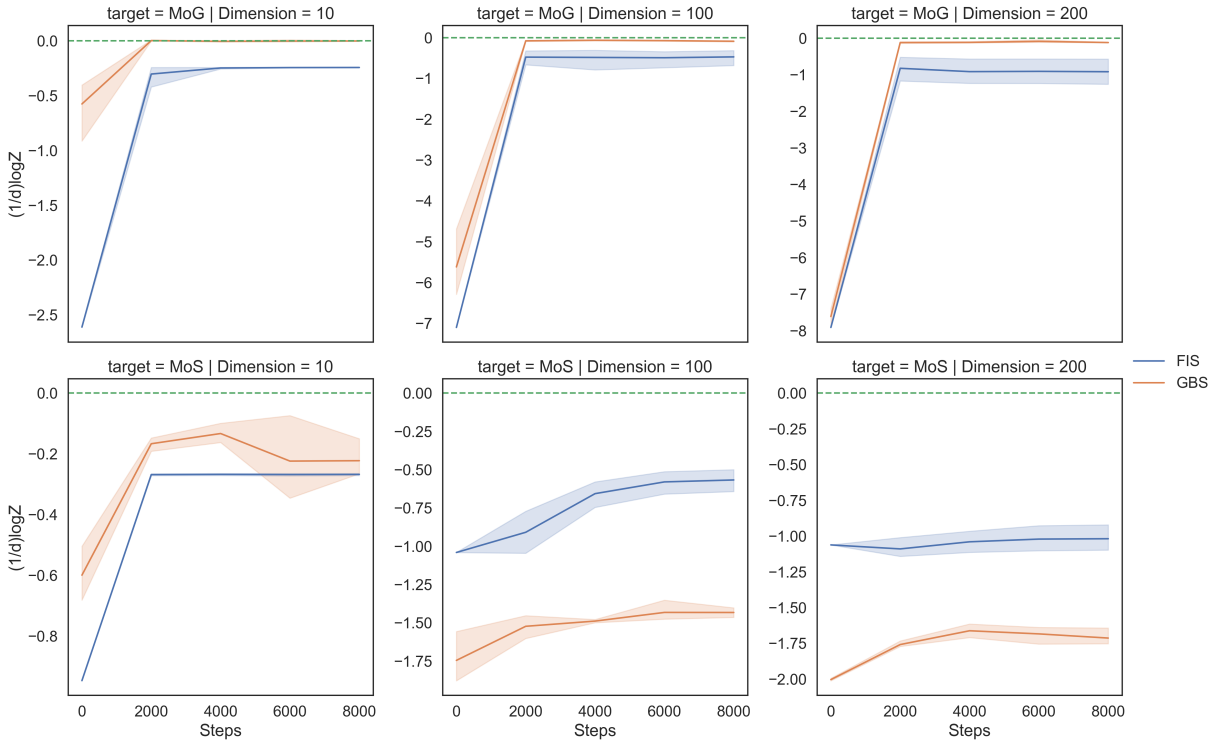

Figure 9: Estimates of $\frac{1}{d}\log Z$ as a function of training steps given by FIS and GBS for different targets and dimensions.

Table 5: Estimates of $\log Z$ and 2-Wasserstein distance obtained using FIS and GBS for different targets and dimensions.

Distribution	Dimension	$\log Z$ (True value $= 0$)		2-Wasserstein	
		FIS	GBS	FIS	GBS
MoG	$d = 10$	-2.43 ± 0.02	-0.02 ± 0.03	243.5 ± 5	6.21 ± 0.04
	$d = 100$	-47.24 ± 26.1	-8.96 ± 1.25	5088 ± 688	147.8 ± 0.83
	$d = 200$	-183.56 ± 90.24	-23.91 ± 1.26	10035 ± 1136	323.4 ± 2.3
MoS	$d = 10$	-2.68 ± 0.04	-2.22 ± 0.63	285.9 ± 110	317.1 ± 8.8
	$d = 100$	-56.75 ± 9.5	-143.3 ± 3.11	3156 ± 1253	14932 ± 76
	$d = 200$	-203.77 ± 23.65	-342.7 ± 12.24	9208 ± 1653	34573 ± 3650

H.5 Comparison with other Diffusion-based Samplers

Existing diffusion-based samplers that could be considered closest to our work are PIS [Zhang and Chen, 2022], DIS [Berner et al., 2023], and DDS [Vargas et al., 2022]. These methods facilitates the use of important sampling techniques to correct the samples while estimating funcitonals. The estimate of $\log Z$ of GMM without importance sampling is mentioned in [Zhang and Chen, 2022], and it has a bias of -0.44. In this setting, our method has a lower bias given by -0.26. The $\log Z$ estimate of the Funnel distribution reported in [Zhang and Chen, 2022] is much better than ours. However, as noted in [Berner et al., 2023] and [Vargas et al., 2022], the Funnel distribution used in [Zhang and Chen, 2022] has a favourable variance compared to the standard Funnel distribution that we used.

Training of neural networks in [Zhang and Chen, 2022], DIS [Berner et al., 2023], and DDS [Vargas et al., 2022] involves computing gradient of a Neural SDE, which can turn quite expensive computationally. However, our methods do not take gradient with respect to the process (17).

H.6 Effect of different parameters

Table 6 shows the influence of different hyperparameters on the estimates. The target distribution used is GMM ($d = 2$).

Table 6: Influence of different hyperparameters on the estimates obtained. Here, $c_1 = 5 * 10^{-6}$, $c_2 = 4 * 10^8$.

Hyperparameter	Value	Estimates		
		$\log Z$	$\mathbb{E}\lvert X\rvert_1$	$\mathbb{E}\lVert X\rVert^2$
δ	$0.1c_1$	-0.66 ± 0.10	4.69 ± 0.39	20.44 ± 2.26
	$1.0c_1$	-0.25 ± 0.02	6.05 ± 0.24	28.42 ± 1.37
	$10c_1$	-0.67 ± 0.19	4.27 ± 0.71	18.41 ± 3.93
λ	$0.1c_2$	-0.30 ± 0.02	5.85 ± 0.17	27.25 ± 0.97
	$1c_2$	-0.26 ± 0.01	6.02 ± 0.21	28.28 ± 1.18
	$10c_2$	-0.74 ± 0.16	4.1 ± 0.55	17.4 ± 3.05
SDE steps	30	-0.31 ± 0.04	5.86 ± 0.31	27.45 ± 1.77
	60	-0.27 ± 0.02	6.05 ± 0.22	28.47 ± 1.22
	100	-0.27 ± 0.03	6.07 ± 0.20	28.55 ± 1.17