RIEMANNIAN DENOISING DIFFUSION PROBABILISTIC MODELS*

ZICHEN LIU†, WEI ZHANG‡, CHRISTOF SCHÜTTE§, AND TIEJUN LI¶

Abstract. We propose Riemannian Denoising Diffusion Probabilistic Models (RDDPMs) for learning distributions on submanifolds of Euclidean space that are level sets of functions, including most of the manifolds relevant to applications. Existing methods for generative modeling on manifolds rely on substantial geometric information such as geodesic curves or eigenfunctions of the Laplace-Beltrami operator and, as a result, they are limited to manifolds where such information is available. In contrast, our method, built on a projection scheme, can be applied to more general manifolds, as it only requires being able to evaluate the value and the first order derivatives of the function that defines the submanifold. We provide a theoretical analysis of our method in the continuous-time limit, which elucidates the connection between our RDDPMs and score-based generative models on manifolds. The capability of our method is demonstrated on distributions over complex manifolds implicitly represented as level sets, with applications in statistical mechanics and molecular dynamics.

Keywords. generative modeling; diffusion probabilistic model; submanifold; projection scheme.

AMS subject classifications. 58J65; 60J05; 60J60.

1. Introduction

Generative models have achieved remarkable success in learning data distributions across various fields. Among them, diffusion models stand out for their superior ability to generate high-quality samples that resemble the data distributions. Two prominent frameworks are Denoising Diffusion Probabilistic Models (DDPMs; [10]), which minimize a variational bound in variational inference, and Score-based Generative Models (SGMs; [30, 31]), which learn the score function [14]. Both frameworks have demonstrated significant success in learning data distributions in Euclidean spaces.

In many scientific domains, data distributions are constrained to Riemannian manifolds rather than Euclidean spaces. For example, spheres are used in geographical sciences [23], while SE(3) and SO(3) are considered in studying protein structures [34] and robotic movements [27]. Other manifolds include SU(3) in lattice quantum chromodynamics [22], triangular meshes in 3D computer graphics [11], and the Poincaré disk in cell development research [17]. These applications highlight the need for developing generative models that can handle distributions on manifolds.

Several recent works have extended diffusion-based models to Riemannian manifolds. The Riemannian Score-based Generative Model (RSGM; [7]) extends SGM to Riemannian manifolds by incorporating the heat kernel into the denoising score-matching objective. Since heat kernels on manifolds are generally intractable, RSGM approximates them using eigenfunction expansion or Varadhan's approximation. Furthermore, RSGM leverages the exponential map to enable trajectory sampling on manifolds. The Riemannian Diffusion Model [13] adopts a variational diffusion model framework on Riemannian manifolds. It considers submanifolds embedded in an Euclidean space and utilizes a

*Received date, and accepted date. Communicated by ***

†**Equal contribution.** Center for Data Science, Peking University, Beijing 100871, P.R. China (zcliu@stu.pku.edu.cn).

‡**Equal contribution.** Zuse Institute Berlin, Takustrasse 7, Berlin 14195, Germany (wei.zhang@fu-berlin.de).

§Institute of Mathematics, Freie Universität Berlin and Zuse Institute Berlin, Takustrasse 7, Berlin 14195, Germany (schuette@zib.de).

¶**Corresponding author.** LMAM and School of Mathematical Sciences, Center for Machine Learning Research and Center for Data Science, Peking University, Beijing 100871, P.R. China (tieli@pku.edu.cn).

variational upper bound on the negative log-likelihood as loss function. Additionally, the Trivialized Diffusion Model [39] is able to solve generative tasks on Lie groups by utilizing group property of the underlying manifold. Beyond diffusion-based models, flow-based generative models [2, 3, 21, 23, 25] extend continuous normalizing flows to manifolds. In particular, the method proposed in [3] accommodates manifolds with relatively general geometries, provided that their geodesic curves can be efficiently computed.

Despite these advancements, existing methods heavily rely on geometric information of manifolds, such as geodesics, exponential maps, heat kernels, or metrics. This restricts their applicability to manifolds where such information is readily available. In view of these limitations, we aim to develop algorithms that bypass these geometric dependencies, making them applicable to more general manifolds.

In this work, we introduce Riemannian Denoising Diffusion Probabilistic Models (RDDPMs), an extension of DDPMs to Riemannian submanifolds. A key ingredient is the projection scheme used in Monte Carlo methods for sampling under constraints, which allows us to develop Markov chains on manifolds with explicit transition densities. To our best knowledge, no successful extension of DDPMs to manifolds has been proposed prior to this study. The main advantages of our method over existing methods are summarized below.

- Our method is developed for submanifolds that are level sets of smooth functions in Euclidean space. This general setting includes most of the often studied manifolds such as spheres and matrix groups. More importantly, it fits well with applications where constraints are involved, e.g. applications in statistical mechanics and molecular dynamics.
- Our method requires neither geodesic curves nor heat kernel, and it only relies on the computation of the value and the first-order derivatives of the function that defines the submanifold. This makes our approach applicable to more general manifolds.
- We present a theoretical analysis for the loss function of our method in the continuous-time limit, elucidating its connection to the existing methods [7]. This analysis also shows the equivalence between loss functions derived from variational bound in variational inference and from learning score function.

We successfully apply our method to distributions on manifolds studied in prior works, as well as new manifolds implicitly defined as level sets, such as the configuration space of alanine dipeptide with a fixed dihedral angle and the conserved Hamiltonian surface in phase space, both of which are challenging for existing methods due to their geometric complexity.

2. Background

2.1. Riemannian submanifolds. We consider the zero level set $\mathcal{M} = \{x \in \mathbb{R}^n | \xi(x) = 0\}$ of a smooth function $\xi\colon \mathbb{R}^n \to \mathbb{R}^{n-d}$, where $1 \le d < n$. We assume that $\mathcal{M}$ is non-empty and the matrix $\nabla\xi(x) \in \mathbb{R}^{n\times(n-d)}$, i.e. the Jacobian of ξ, has full rank at each $x \in \mathcal{M}$. Under this assumption, the regular value theorem [1, Corollary 5.9] implies that $\mathcal{M}$ is a d-dimensional submanifold of $\mathbb{R}^n$. We further assume that $\mathcal{M}$ is a smooth compact connected manifold without boundary. The Riemannian metric on $\mathcal{M}$ is endowed from the standard Euclidean distance on $\mathbb{R}^n$. For $x \in \mathcal{M}$, we denote by $T_x\mathcal{M}$ the tangent space of $\mathcal{M}$ at x. The orthogonal projection matrix $P(x) \in \mathbb{R}^{n\times n}$ mapping $T_x\mathbb{R}^n = \mathbb{R}^n$ to $T_x\mathcal{M}$ is given by $P(x) = I_n - \nabla\xi(x)\big(\nabla\xi(x)^\top\nabla\xi(x)\big)^{-1}\nabla\xi(x)^\top$. Let $U_x \in \mathbb{R}^{n\times d}$ be a matrix whose column vectors form an orthonormal basis of $T_x\mathcal{M}$ such that $U_x^\top U_x = I_d$. It is straightforward to verify that $P(x) = U_x U_x^\top$. The volume

element over $\mathcal{M}$ is denoted by $\sigma_{\mathcal{M}}$. All probability densities that appear in this paper refer to relative probability densities with respect to either $\sigma_{\mathcal{M}}$ or the product of $\sigma_{\mathcal{M}}$ over product spaces. For notational simplicity, we also use the shorthand $\int p(x^{(1:N)})dx^{(1:N)} := \int_{\mathcal{M}}\cdots\int_{\mathcal{M}} p(x^{(1)},\ldots,x^{(N)})d\sigma_{\mathcal{M}}(x^{(1)})\cdots d\sigma_{\mathcal{M}}(x^{(N)})$.

2.2. Denoising diffusion probabilistic models. Denoising diffusion probabilistic models DDPMs [10,28] employ a forward Markov chain to perturb data into noise and a reverse Markov chain to incrementally recover data from noise. The models are trained to minimize a variational bound on the negative log-likelihood. In the following, we formulate the general steps of DDPMs in the manifold setting.

Assume that the data distribution is $q_0(x)d\sigma_{\mathcal{M}}(x)$. DDPMs are a class of generative models built on Markov chains. Specifically, states $x^{(1)},\ldots,x^{(N)}\in\mathcal{M}$ are generated by evolving the data $x^{(0)}$ according to a Markov chain on $\mathcal{M}$, which is called the forward process. The joint probability density of $x^{(1)},\ldots,x^{(N)}$ given $x^{(0)}$ is

$$q(x^{(1:N)}\,|\,x^{(0)}) = \prod_{k=0}^{N-1} q(x^{(k+1)}\,|\,x^{(k)}), \tag{2.1}$$

where $q(x^{(k+1)}\,|\,x^{(k)})$ is the transition density of the forward process. The generative process, also called the reverse process, is a Markov chain on $\mathcal{M}$ that is learnt to reproduce the data by reversing the forward process. Its joint probability density is

$$p_\theta(x^{(0:N)}) = p(x^{(N)}) \prod_{k=0}^{N-1} p_\theta(x^{(k)}\,|\,x^{(k+1)}), \tag{2.2}$$

where $p(x^{(N)})$ is a (fixed) prior density, $p_\theta(x^{(k)}\,|\,x^{(k+1)})$ is the transition density of the reverse process, and θ is the parameter to be learnt. The probability density of $x^{(0)}$ generated by the reverse process is therefore $p_\theta(x^{(0)}) = \int p_\theta(x^{(0:N)})dx^{(1:N)}$. The learning objective is based on the standard variational bound on the negative log-likelihood. Specifically, using (2.1) and (2.2), and applying Jensen's inequality, we can derive

$$\begin{aligned}
\mathbb{E}_{q_0}\big(-\log p_\theta(x^{(0)})\big) &= \mathbb{E}_{q_0}\Big(-\log\int p_\theta(x^{(0:N)})dx^{(1:N)}\Big)\\
&= \mathbb{E}_{q_0}\Big(-\log\int \frac{p_\theta(x^{(0:N)})}{q(x^{(1:N)}\,|\,x^{(0)})}q(x^{(1:N)}\,|\,x^{(0)})dx^{(1:N)}\Big)\\
&\le \mathbb{E}_{\mathbb{Q}^{(N)}}\Big(-\log\frac{p_\theta(x^{(0:N)})}{q(x^{(1:N)}\,|\,x^{(0)})}\Big)\\
&= \mathbb{E}_{\mathbb{Q}^{(N)}}\Big(-\log p(x^{(N)}) - \sum_{k=0}^{N-1}\log\frac{p_\theta(x^{(k)}\,|\,x^{(k+1)})}{q(x^{(k+1)}\,|\,x^{(k)})}\Big),
\end{aligned} \tag{2.3}$$

where $\mathbb{E}_{q_0}, \mathbb{E}_{\mathbb{Q}^{(N)}}$ denote the expectation with respect to the data distribution on $\mathcal{M}$, and the expectation with respect to the joint density $q(x^{(0:N)})$ under the forward process, respectively. In order to derive an explicit training objective, we have to construct Markov chains on $\mathcal{M}$ with explicit transition densities. We discuss how this can be achieved in the next section.

We conclude this section by reformulating the variational bound (2.3) using relative entropy (see [29] for a similar formulation of score-based diffusion models). Recall that the relative entropy, or Kullback-Leibler (KL) divergence, from a probability density Q_2

to another probability density Q_1 in the same measure space, where Q_1 is absolutely continuous with respect to Q_2, is defined as $H(Q_1|Q_2) := \mathbb{E}_{Q_1}\left(\log \frac{Q_1}{Q_2}\right)$. For simplicity, we also use the same notation for two probability measures. Adding the term $\mathbb{E}_{q_0}(\log q_0)$ to both sides of the inequality (2.3), we see that it is equivalent to the so-called data processing inequality

$$H(q_0|p_\theta) \le H(\overleftarrow{\mathbb{Q}}^{(N)}|\mathbb{P}_\theta^{(N)}), \tag{2.4}$$

where the upper bound is the relative entropy from the path measure $\mathbb{P}_\theta^{(N)}$ of the reverse process to the path measure $\overleftarrow{\mathbb{Q}}^{(N)}$ of the forward process (the arrow in the notation indicates that paths of the forward process are viewed backwardly). Therefore, learning DDPMs using the variational bound (2.3) can be viewed as approximating probability measures in path space by the cross-entropy method [38].

3. Method

3.1. Projection scheme. We recall a projection scheme from Monte Carlo sampling methods on manifolds [4, 19, 20, 36], and we show that it allows us to construct Markov chains on $\mathcal{M}$ with tractable transition densities.

Given $x \in \mathcal{M}$ and a tangent vector $v \in T_x\mathcal{M}$ that is drawn from the standard Gaussian distribution on $T_x\mathcal{M}$, we compute the intermediate state $x' = x + \sigma^2 b(x) + \sigma v \in \mathbb{R}^n$, where $\sigma > 0$ is a positive constant and $b : \mathbb{R}^n \to \mathbb{R}^n$ is a smooth function. In general, x' does not belong to $\mathcal{M}$. We consider the projection $y \in \mathcal{M}$ of x' onto $\mathcal{M}$ along an orthogonal direction in the space spanned by column vectors of $\nabla\xi(x)$. Precisely, the projected state y is found by (numerically) solving the constraint equation for $c \in \mathbb{R}^{n-d}$

$$y = x + \sigma^2 b(x) + \sigma v + \nabla\xi(x)c, \quad \text{s.t.} \quad \xi(y) = 0 \in \mathbb{R}^{n-d}. \tag{3.1}$$

This projection scheme can be seen as a discretization of a certain stochastic process on the manifold $\mathcal{M}$, where σ^2 corresponds to the step size [4]. The choice of b will affect the final invariant distribution and the convergence rate to equilibrium of the resulting Markov chain (see Section 3.5 for further discussion). There are $n-d$ constraints in (3.1) with the same number of unknown variables. In particular, when ξ is scalar-valued, i.e., $n-d=1$, solving (3.1) amounts to finding a root of a (nonlinear) scalar function.

Algorithm 3 shows the details of solving the linear equations. Let k_{iter} denote the number of Newton iterations, and C_ξ be the computational cost of evaluating $\nabla\xi$. The complexity of solving the linear equations in line 4 of Algorithm 3 is $\mathcal{O}((n-d)^3)$. Thus, the total complexity of Newton's method is $\mathcal{O}(k_{\text{iter}}(C_\xi + (n-d)^3))$. When the step size σ is sufficiently small, Newton's method typically converges in a few iterations. Therefore, the computational cost of Newton's method is primarily determined by the cost of computing $\nabla\xi$ (i.e., C_ξ) and the codimension $n-d$ of the manifold.

While for some vectors v multiple solutions to (3.1) may exist in theory, the projected state, and hence the resulting Markov chain built on a numerical solver, is uniquely defined as long as the numerical solver finds one solution in a deterministic way. For example, this is the case when Newton's method is adopted to solve (3.1) with fixed initial condition $c = 0$. When no solution can be found for some v, we can either resample the tangent vector v or resample a new path (see Section 3.5 for further discussion). Let $\mathcal{F}_x^{(\sigma)}$ be the set of v for which a solution can be found and denote by $\epsilon_x^{(\sigma)} = \mathbb{P}(v \notin \mathcal{F}_x^{(\sigma)})$, i.e. the probability that no solution can be found. We denote $\mathcal{M}_x^{(\sigma)}$ the set of all states in $\mathcal{M}$ that can be reached from x by solving (3.1) with certain $v \in \mathcal{F}_x^{(\sigma)}$. Notice that

$\epsilon_x^{(\sigma)}=0$ when $\sigma=0$, because in this case $c=0$ is a solution to (3.1) for any v. Therefore, it is expected that $\epsilon_x^{(\sigma)}\to 0$ as σ decreases to zero. That is, when σ is small, a solution to (3.1) can be found with a probability that is close to one.

To derive the transition density of jumping to y from x, we notice that, by applying the orthogonal projection matrix $P(x)$ to both sides of (3.1) and using the fact that $P(x)\nabla\xi(x)=0$, we have the relation $\sigma v=P(x)(y-x-\sigma^2 b(x))$. This indicates that, given a state $x\in\mathcal{M}$ and $y\in\mathcal{M}_x^{(\sigma)}$, there is a unique tangent vector $v\in\mathcal{F}_x^{(\sigma)}\subseteq T_x\mathcal{M}$ that leads to y by solving (3.1). In other words, the mapping from $v\in\mathcal{F}_x^{(\sigma)}$ to $y\in\mathcal{M}_x^{(\sigma)}$ is a bijection. Moreover, its inverse is explicitly given by $G_x^{(\sigma)}:\mathcal{M}_x^{(\sigma)}\to\mathcal{F}_x^{(\sigma)}\subseteq T_x\mathcal{M}$, where

$$G_x^{(\sigma)}(y;b)=\frac{1}{\sigma}P(x)(y-x-\sigma^2 b(x)). \tag{3.2}$$

To simplify the notation, we also write $G_x^{(\sigma)}(y)$ when omitting the dependence on b does not cause ambiguity. Recall that $U_x,U_y\in\mathbb{R}^{n\times d}$ denote the matrices whose columns form an orthonormal basis of $T_x\mathcal{M}$ and $T_y\mathcal{M}$, respectively. Using (3.2), we can derive

$$\det(DG_x^{(\sigma)}(y))=\sigma^{-d}\det(U_x^\top U_y), \tag{3.3}$$

where the left hand side denotes the determinant of the Jacobian $DG_x^{(\sigma)}(y):T_y\mathcal{M}\to T_v T_x\mathcal{M}\cong\mathbb{R}^d$ of the map $G_x^{(\sigma)}$ at y. See the texts above (4.5) in Section 4 of [20] for detailed discussions. Since v is a Gaussian variable confined in $\mathcal{F}_x^{(\sigma)}$ (with a normalizing constant rescaled by $(1-\epsilon_x^{(\sigma)})^{-1}$), applying the change of variables formula for probability densities, we obtain the probability density of y conditioned on x:

$$\begin{aligned} q(y|x)&=(2\pi)^{-\frac{d}{2}}(1-\epsilon_x^{(\sigma)})^{-1}\mathrm{e}^{-\frac{1}{2}|G_x^{(\sigma)}(y)|^2}|\det DG_x^{(\sigma)}(y)| \\ &=(2\pi\sigma^2)^{-\frac{d}{2}}(1-\epsilon_x^{(\sigma)})^{-1}\mathrm{e}^{-\frac{1}{2}|G_x^{(\sigma)}(y)|^2}|\det(U_x^\top U_y)|, \quad y\in\mathcal{M}_x^{(\sigma)}. \end{aligned} \tag{3.4}$$

For $y\in\mathcal{M}\setminus\mathcal{M}_x^{(\sigma)}$, the probability density is zero, i.e., $q(y|x)=0$.

3.2. Forward process. We construct the forward process in our model as a Markov chain on $\mathcal{M}$ whose transitions are defined by the projection scheme in (3.1). Specifically, given the current state $x^{(k)}\in\mathcal{M}$ at step k, where $k=0,1,\dots,N-1$, the next state $x^{(k+1)}\in\mathcal{M}$ is determined by solving the constraint equation (for $c\in\mathbb{R}^{n-d}$):

$$x^{(k+1)}=x^{(k)}+\sigma_k^2 b(x^{(k)})+\sigma_k v^{(k)}+\nabla\xi(x^{(k)})c,\quad \text{s.t.}\quad \xi(x^{(k+1)})=0\in\mathbb{R}^{n-d}, \tag{3.5}$$

where $\sigma_k>0$ and $v^{(k)}\in T_{x^{(k)}}\mathcal{M}$ is a standard Gaussian variable in $T_{x^{(k)}}\mathcal{M}$. According to (2.1) and (3.4), we obtain the transition probability density of the forward process as

$$q(x^{(k+1)}|x^{(k)})=(2\pi\sigma_k^2)^{-\frac{d}{2}}|\det(U_{x^{(k)}}^\top U_{x^{(k+1)}})|\big(1-\epsilon_{x^{(k)}}^{(\sigma_k)}\big)^{-1}\exp\left(-\frac{1}{2}\left|G_{x^{(k)}}^{(\sigma_k)}(x^{(k+1)};b)\right|^2\right), \tag{3.6}$$

where the function $G_{x^{(k)}}^{(\sigma_k)}(x^{(k+1)};b)$ is defined in (3.2).

3.3. Reverse process. The reverse process in our model is a Markov chain on $\mathcal{M}$ whose transitions (from $x^{(k+1)}$ to $x^{(k)}$) are defined by the constraint equation

$$\begin{aligned} &x^{(k)}=x^{(k+1)}-\beta_{k+1}^2 b(x^{(k+1)})+\beta_{k+1}^2 s^{(k+1),\theta}(x^{(k+1)})+\beta_{k+1}\bar{v}^{(k+1)}+\nabla\xi(x^{(k+1)})c,\\ &\text{s.t.}\quad \xi(x^{(k)})=0, \end{aligned} \tag{3.7}$$

for $k=N-1,N-2,\ldots,0$, where $\beta_{k+1}>0$, $\bar{v}^{(k+1)}$ is a standard Gaussian variable in $T_{x^{(k+1)}}\mathcal{M}$, and $s^{(k+1),\theta}(x^{(k+1)})\in\mathbb{R}^n$ depends on the learning parameter θ. Combining (2.2) and (3.4), we obtain the transition density of the reverse process as

$$\begin{aligned}&p_\theta(x^{(k)}\,|\,x^{(k+1)})\\ =&(2\pi\beta_{k+1}^2)^{-\frac{d}{2}}(1-\epsilon^{(\beta_{k+1})}_{x^{(k+1)},\theta})^{-1}|\det(U^\top_{x^{(k+1)}}U_{x^{(k)}})|\exp\left(-\frac{1}{2}\left|G^{(\beta_{k+1})}_{x^{(k+1)}}(x^{(k)};s^{(k+1),\theta}-b)\right|^2\right).\end{aligned}\tag{3.8}$$

In the above, $\epsilon^{(\beta_{k+1})}_{x^{(k+1)},\theta}$ denotes the probability of having $\bar{v}^{(k+1)}$ with which no solution to (3.7) can be found and $G^{(\beta_{k+1})}_{x^{(k+1)}}(x^{(k)};s^{(k+1),\theta}-b)$ is defined in (3.2).

3.4. Training objective. The training objective follows directly from the variational bound (2.3) on the negative log-likelihood, as well as the explicit expressions of transition densities. Concretely, substituting (3.6) and (3.8) into the last line of (2.3), we get

$$\mathbb{E}_{q_0}[-\log p_\theta(x^{(0)})]\le \mathrm{Loss}^{(N)}(\theta)+C^{(N)},\tag{3.9}$$

where

$$\mathrm{Loss}^{(N)}(\theta)=\frac{1}{2}\mathbb{E}_{\mathbb{Q}^{(N)}}\sum_{k=0}^{N-1}\left|G^{(\beta_{k+1})}_{x^{(k+1)}}(x^{(k)};s^{(k+1),\theta}-b)\right|^2\tag{3.10}$$

is our objective for training the parameter θ in the reverse process (recall that $\mathbb{E}_{\mathbb{Q}^{(N)}}$ denotes the expectation with respect to the forward process), the constant

$$\begin{aligned}C^{(N)}=&-\mathbb{E}_{\mathbb{Q}^{(N)}}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left|G^{(\sigma_k)}_{x^{(k)}}(x^{(k+1)};b)\right|^2+\log p(x^{(N)})\right]\\&+d\sum_{k=0}^{N-1}\log\frac{\beta_{k+1}}{\sigma_k}-\mathbb{E}_{\mathbb{Q}^{(N)}}\sum_{k=0}^{N-1}\log\left(1-\epsilon^{(\sigma_k)}_{x^{(k)}}\right)\end{aligned}\tag{3.11}$$

is independent of θ, and we have used the inequality $\log\left(1-\epsilon^{(\beta_{k+1})}_{x^{(k+1)},\theta}\right)\le 0$ in deriving (3.9).

3.5. Algorithmic details. The algorithms for sampling the trajectories of the forward and reverse processes are summarized in Algorithms 1 and 2, respectively, both of which involve solving constraint equations, as detailed in Algorithm 3. The training algorithm is summarized in Algorithm 4. In the following, we discuss several algorithmic details of our method.

3.5.1. Choice of the Markov chain. The total number of steps N should be large enough so that the forward Markov chain can approximately reach equilibrium starting from the data distribution. To simplify algorithm implementation, we may set $\beta_{k+1}=\sigma_k$, a choice that is also supported by our theoretical derivation in the continuous-time limit (see Theorem 4.1). While larger σ_k, β_{k+1} allow the Markov chains to make larger jumps, their sizes should be chosen properly (depending on the manifold) so that the solution to the constraint equations (3.5) and (3.7) can be found with high probability.

For relatively simple manifolds, we can simply choose $b=0$ in the forward Markov chain (3.5). As we will discuss in Section 4, in the continuous-time limit, the Markov chain with $b=0$ converges to the Brownian motion on $\mathcal{M}$ up to a rescaling of time ((4.1) with $b=0$), whose invariant distribution is the uniform distribution on $\mathcal{M}$. Motivated by this fact, we can choose the prior (see line 2 in Algorithm 2) as the uniform distribution on $\mathcal{M}$. When $\mathcal{M}$ is non-compact or when the convergence of Markov chain to equilibrium is slow with $b=0$, we can choose non-zero b such as $b=-\nabla V$, i.e. the (full space) gradient of some function $V:\mathbb{R}^n\to\mathbb{R}$ in the ambient space. In this case, we choose the prior as the invariant distribution of the forward process, and sampling the prior can be done by simulating a single long trajectory of the forward process.

3.5.2. Method for solving constraint equations. As in Monte Carlo sampling methods on submanifolds, we employ Newton's method to solve the constraint equations (3.5) and (3.7). This method has (local) quadratic convergence and its implementation is simple. In most cases, a solution with high precision can be found within a few iteration steps (usually less than 5 steps). When no solution is found, one can re-generate the state by sampling a new tangent vector or re-generate the entire trajectory. Our implementation of Newton's method is summarized in Algorithm 3.

3.5.3. Generation of trajectory data. The optimal parameter θ is sought by minimizing the objective (3.10), which requires the trajectory data of the forward process. Although sampling trajectories to evaluate the loss function may seem computationally expensive, the computational cost can be alleviated by using a pre-prepared trajectory dataset that is updated during training with a tunable frequency. In our implementation, trajectories are initially sampled in a preparatory step, and the model is trained with mini-batches from this trajectory dataset, which is periodically updated (see line 2 and lines 10–12 in Algorithm 4).

Algorithm 1 Sampling trajectory of forward process

1: **Input:** $x^{(0)}\in\mathcal{M}$, constants σ_k, function b, and integer N
2: **for** $k=0$ to $N-1$ **do**
3: sample $z^{(k)}\sim\mathcal{N}(0,I_n)$ and set $v^k=P(x^{(k)})z^{(k)}$
4: set $x^{(k+\frac{1}{2})}:=x^{(k)}+\sigma_k^2 b(x^{(k)})+\sigma_k v^{(k)}$
5: $c,\text{flag}=\text{newton\_solver}(x^{(k)},x^{(k+\frac{1}{2})};\xi)$. // solve (3.5) by Algorithm 3
6: **if** flag == true **then**
7: set $x^{(k+1)}:=x^{(k+\frac{1}{2})}+\nabla\xi(x^{(k)})c$
8: **else**
9: discard the trajectory and re-generate // alternatively, go back to line 3
10: **end if**
11: **end for**
12: **Return** $(x^{(0)},x^{(1)},\ldots,x^{(N)})$

4. Theoretical results In this section, we study the continuous-time limit of our proposed method. Let $T>0$ and $g:[0,T]\to\mathbb{R}^+$ be a continuous function. Define $h=\frac{T}{N}$ and consider the case where $\sigma_k=\sqrt{h}g(kh)$, for $k=0,1,\ldots,N-1$. It is shown in [4] that the forward process (3.5) converges strongly to the SDE on $\mathcal{M}$

$$dX_t=g^2(t)P(X_t)b(X_t)dt+g(t)dW_t^{\mathcal{M}},\qquad t\in[0,T], \tag{4.1}$$

Algorithm 2 Sampling trajectory of reverse process

1: **Input:** trained functions $(s^{(k+1),\theta}(x))_{0\le k\le N-1}$, constants β_k, function b, and integer N
2: draw sample $x^{(N)}$ from the prior distribution $p(x^{(N)})$
3: **for** $k=N-1$ to 0 **do**
4: sample $\bar{z}^{(k+1)}\sim\mathcal{N}(0,I_n)$ and set $\bar{v}^{(k+1)}=P(x^{(k+1)})\bar{z}^{(k+1)}$
5: set $x^{(k+\frac{1}{2})}=x^{(k+1)}+\beta_{k+1}^2P(x^{(k+1)})\left(s^{(k+1),\theta}-b\right)(x^{(k+1)})+\beta_{k+1}\bar{v}^{(k+1)}$
6: $c,\text{flag}=\text{newton\_solver}(x^{(k+1)},x^{(k+\frac{1}{2})};\xi)$ // solve (3.7) by Algorithm 3
7: **if** flag == true **then**
8: $x^{(k)}:=x^{(k+\frac{1}{2})}+\nabla\xi(x^{(k+1)})c$
9: **else**
10: discard the trajectory and re-generate // alternatively, go back to line 4
11: **end if**
12: **end for**
13: **Return** $(x^{(N)},x^{(N-1)},\dots,x^{(0)})$

Algorithm 3 newton_solver$(x,x';\xi)$ // solve $\xi(x'+\nabla\xi(x)c)=0$ by Newton's method

1: **Input:** $x\in\mathcal{M}$, $x'\in\mathbb{R}^n$, $\xi:\mathbb{R}^n\to\mathbb{R}^{n-d}$, maximal steps n_{step}, tolerance $\text{tol}>0$
2: **Initialization**: set $c=0\in\mathbb{R}^{n-d}$
3: **for** $k=0$ to $n_{\text{step}}-1$ **do**
4: Solve linear equation $\left[\nabla\xi\big(x'+\nabla\xi(x)c\big)^\top\nabla\xi(x)\right]u=-\xi\big(x'+\nabla\xi(x)c\big)$ for $u\in\mathbb{R}^{n-d}$
5: $c\leftarrow c+u$
6: **if** $|\xi(x'+\nabla\xi(x)c)|<\text{tol}$ **then**
7: **Return** c, true
8: **end if**
9: **end for**
10: **Return** c, false

where $W_t^{\mathcal{M}}$ is a Brownian motion over $\mathcal{M}$. Denote by $p(\cdot,t)$ the probability density of X_t with respect to $\sigma_{\mathcal{M}}$ at time $t\in[0,T]$. We have the following result, which characterizes the loss function in (3.10) as $N\to+\infty$.

THEOREM 4.1. *Let $T>0$ and $g:[0,T]\to\mathbb{R}^+$ be a continuous function. Define $h=\frac{T}{N}$ and $t_k=kh$, for $k=0,1,\dots,N-1$. Assume that $\sigma_k=\beta_{k+1}=\sqrt{h}g(t_k)$. Also assume that there is a C^1 function $s_\theta:\mathbb{R}^n\times[0,T]\to\mathbb{R}^n$ such that $s^{(k+1),\theta}(x)=s_\theta(x,t_{k+1})$ for all $k=0,1,\dots,N-1$ and $x\in\mathcal{M}$. For the loss function defined in (3.10), we have*

$$\begin{aligned}&\lim_{N\to+\infty}\left(\text{Loss}^{(N)}(\theta)-\frac{1}{2}\sum_{k=0}^{N-1}\left|v^{(k)}\right|^2\right)\\=&\mathbb{E}_{\mathbb{Q}}\Big[\frac{1}{2}\int_0^T\left|P(X_t)s_\theta(X_t,t)-\nabla_{\mathcal{M}}\log p(X_t,t)\right|^2g^2(t)\,dt\\&+\int_0^T\Big(P(X_t)b(X_t)-\frac{1}{2}\nabla_{\mathcal{M}}\log p(X_t,t)\Big)\cdot\nabla_{\mathcal{M}}\log p(X_t,t)\,g^2(t)\,dt\Big],\end{aligned}$$

where $\mathbb{E}_{\mathbb{Q}}$ on the right hand side denotes the expectation with respect to the paths of SDE (4.1) and $\nabla_{\mathcal{M}}$ denotes the gradient operator on $\mathcal{M}$.

Based on Theorem 4.1, the variational bound (2.3), and its relative entropy formula-

Algorithm 4 Training procedure

Input: training data $(y^i)_{1\le i\le M}$, functions $(s^{(k+1),\theta}(x))_{0\le k\le N-1}$, constants $\sigma_k,\beta_k>0$, function $b:\mathbb{R}^n\to\mathbb{R}^n$, integer N, batch size $B>0$, number of total training epochs N_{epoch}, integer $l_{\text{f}}>0$, learning rate $r>0$.
generate a path $(x^{(0),i},x^{(1),i},\dots,x^{(N),i})$ using Algorithm 1, for each $x^{(0),i}=y^i$
for $l=1$ to N_{epoch} **do**
  for $j=1$ to $\lfloor M/B\rfloor$ **do**
    sample a min-batch $\mathcal{I}=(i_1,i_2,\dots,i_B)$ from the set of indices $\{1,2,\dots,M\}$
    calculate loss: $\ell(\theta)=\frac{1}{2|\mathcal{I}|}\sum_{i\in\mathcal{I}}\sum_{k=0}^{N-1}\left|G^{(\beta_{k+1})}_{x^{(k+1),i}}(x^{(k),i};s^{(k+1),\theta}-b)\right|^2$
    $\theta=\text{optimizer\_update}(\theta,\ell(\theta),r)$
  end for
  // update trajectories every l_{f} epochs
  if $l\%l_{\text{f}}==0$ **then**
    re-generate paths $(x^{(0),i},x^{(1),i},\dots,x^{(N),i})$ using Algorithm 1, for each $x^{(0),i}=y^i$
  end if
end for
Return θ

tion in (2.4), we obtain the following corollary, which states that our RDDPMs learn the score function as $N\to+\infty$. It elucidates the connection between our RDDPMs and Riemannian score-based generative models [7] as $N\to+\infty$.

COROLLARY 4.1. *Under the same assumptions of Theorem 4.1, we have, for any parameter* θ,

$$\lim_{N\to+\infty} H\big(\overleftarrow{\mathbb{Q}}^{(N)}\,|\,\mathbb{P}_\theta^{(N)}\big)=\frac{1}{2}\mathbb{E}_{\mathbb{Q}}\int_0^T\left|P(X_t)s_\theta(X_t,t)-\nabla_{\mathcal{M}}\log p(X_t,t)\right|^2 g^2(t)dt=H(\overleftarrow{\mathbb{Q}}\,|\,\mathbb{P}_\theta),\tag{4.2}$$

where $\overleftarrow{\mathbb{Q}}$ *denotes the path measure of the time-reversal of SDE (4.1), and* $\mathbb{P}_\theta$ *denotes the path measure of the SDE*

$$dY_t=g^2(T-t)P(Y_t)\big(-b(Y_t)+s_\theta(Y_t,T-t)\big)dt+g(T-t)dW_t^{\mathcal{M}},\quad t\in[0,T],\tag{4.3}$$

starting from $Y_0=X_T$.

The proofs of Theorem 4.1 and Corollary 4.1 are presented in Appendix A.

5. Experiments

We evaluate our method on distributions defined on mesh manifolds, the high-dimensional special orthogonal group (both with and without a nonlinear transformation), conserved Hamiltonian surfaces in phase space, and molecular conformations under constraints. The last three novel datasets have not been studied by existing methods. Further experimental details and results can be found in Appendix B. In particular, parameters for each example are summarized in Table B.1 in Appendix B.

5.1. Mesh data on learned manifolds. Our method can effectively handle manifolds with general geometries. For demonstration, we examine the Stanford Bunny [32] and Spot the Cow [6], two manifolds defined by triangle meshes. To create the target distribution, we follow the approach in [15] and [3], which utilizes the k-th clamped eigenfunction of the Laplace-Beltrami operator on meshes.

Similar to [25], we first learn a function $\xi : \mathbb{R}^3 \to \mathbb{R}$, whose zero level set matches the manifold. We adopt the approach in [9], where ξ is represented by a neural network and is trained such that on mesh data ξ is close to zero and $|\nabla \xi|$ is close to one. Using this approach, we obtain a function ξ whose order of magnitude is 10^{-2} on mesh data. Then, we perform a further refinement to the dataset such that all points belong to the learned manifold $\mathcal{M} = \{x \in \mathbb{R}^3 | \xi(x) = 0\}$ up to a small error 10^{-5}. The maximal distance between the original data and the refined data is smaller than 0.017.

The function b in the forward Markov chain (3.5) is set to be the zero function, and the prior distribution $p(x^{(N)})$ is a uniform distribution on the meshes (see discussions in Section 3.5). We perform the training with the learned function ξ.

In Table 5.1, we present the negative log-likelihood (NLL) on the test set, estimated via the second line of (2.3). Our method outperforms existing manifold-based methods, including RFM [3] and LogBM [15]. One possible explanation for this improvement is that existing methods require computing the premetric on meshes through infinite series (see Equation (16) in [3]), which introduces bias due to truncation in practice. In principle, our method is unbiased, as it does not require computing distances on the manifold. Figure 5.1 visualizes the generated samples, demonstrating good agreement with the target data distribution.

	Stanford Bunny		**Spot the Cow**	
	$k = 50$	$k = 100$	$k = 50$	$k = 100$
RFM w/ Diff.	1.48±0.01	1.53±0.01	0.95±0.05	1.08±0.05
RFM w/ Bihar.	1.55±0.01	1.49±0.01	1.08±0.05	1.29±0.05
LogBM w/ Diff.	1.42±0.01	1.41±0.00	0.99±0.03	0.97±0.03
LogBM w/ Bihar.	1.55±0.02	1.45±0.01	1.09±0.06	0.97±0.02
Ours				
RDDPM	**1.36**±0.00	**1.31**±0.01	**0.84**±0.00	**0.77**±0.00

Table 5.1: Test negative log-likelihood (NLL) on mesh datasets. Lower values indicate better performance. The table shows the mean and the standard deviation of the NLL over five independent runs. For RFM and LogBM, *with Diff* and *with Bihar* refer to different weighting functions used in computing the spectral distances (diffusion distance and biharmonic distance, respectively).

5.2. High-dimensional special orthogonal group. We apply our method to the special orthogonal group SO(10), viewed as a 45-dimensional submanifold embedded in $\mathbb{R}^{100}$. This manifold can be characterized as (one of the two connected components of) the zero level set of the map $\xi : \mathbb{R}^{100} \to \mathbb{R}^{55}$, whose components are defined by the upper triangular entries of the matrix $S^\top S - I_{10}$, where S is a 10×10 matrix and I_{10} denotes the identity matrix of size 10.

The dataset is sampled from a multimodal distribution on SO(10) with 5 modes. As in the previous example, we choose b in the forward Markov chain to be zero and the prior distribution to be the uniform distribution on the manifold.

To assess the quality of generated data, we consider the statistics $\mathrm{tr}(S)$, $\mathrm{tr}(S^2)$, $\mathrm{tr}(S^4)$, and $\mathrm{tr}(S^5)$, where tr denotes the trace operator of matrices. Figure 5.2 indicates that our learned model can generate the data distribution accurately. What is more, the distributions of the forward process at intermediate steps are also faithfully reproduced.

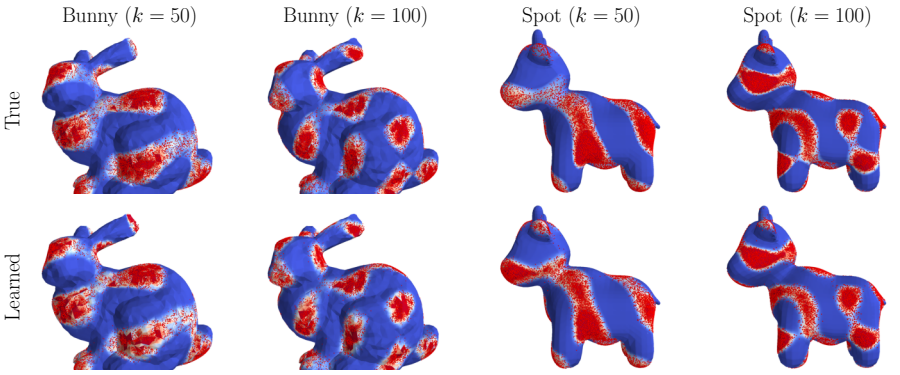


Fig. 5.1: First row: datasets and true distributions. Second row: generated samples and distributions from the trained models.

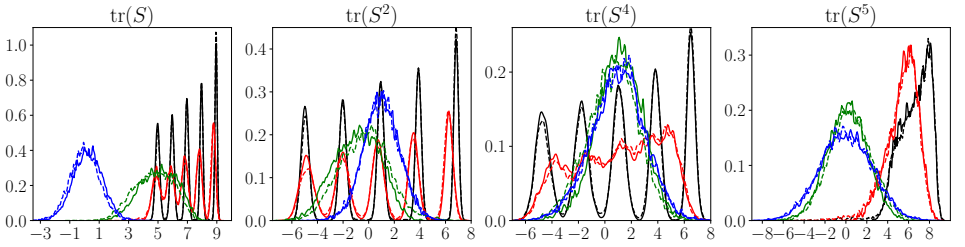


Fig. 5.2: Results for SO(10). Empirical densities of the statistics $\mathrm{tr}(S), \mathrm{tr}(S^2), \mathrm{tr}(S^4)$, and $\mathrm{tr}(S^5)$ for the forward process (solid line) and the learned reverse process (dashed line) at different steps $k=0,50,200,500$, colored in black, red, green, and blue, respectively.

Besides, to better demonstrate that our method can be applied to more general manifolds, we also consider the manifold $\varphi^{-1}(\mathrm{SO}(10))$, which is a nonlinear transformation of SO(10). Specifically, the manifold is defined as the space of matrices $S=(x_{i,j})_{1\le i,j\le 10}$ that satisfy the constraints $\sum_{k=1}^{10}\varphi(x_{i,k})\varphi(x_{j,k})-\delta_{i,j}=0$, $1\le i\le j\le 10$, where $\varphi(x)=\tan(\frac{\pi}{4}x)$. Consequently, it is also a 45-dimensional submanifold of $\mathbb{R}^{100}$. The target distribution is again chosen as a multimodal distribution with five modes. The function b in the forward Markov chain is defined as $b(x)=-\nabla V(x)$, where $V(x)=\frac{5}{2}\|x-I_{10}\|_F^2$ ($\|\cdot\|_F$ denotes the Frobenius norm). As shown in Figure 5.3, our method successfully generates data that matches the true data.

5.3. Conserved Hamiltonian surface in phase space. We consider conserved Hamiltonian surface in phase space, where the manifold is defined as the space of states $x=(q,p)\in\mathbb{R}^{n_0}\times\mathbb{R}^{n_0}$ satisfying the constraint $\xi(q,p)=H(q,p)-E=0$, where $E>0$ is a constant, and the Hamiltonian is defined as $H(q,p)=\frac{|p|^2}{2m}+U(q)$ with $U(q)=\frac{\kappa}{2}|q|^2+\lambda\sum_{i=1}^{n_0}q_i^4$ representing the potential energy of a nonlinear oscillator.

In our experiment, we choose $n_0=10$, $E=10$, $m=0.5$, and $\kappa=\lambda=2$. Since the uniform distribution on the manifold is difficult to obtain, we define the function b as $b(x)=-\nabla V(x)$, where $V(x)=\frac{5}{2}\sum_{i=1}^{n_0}(q_i^2+(p_i-1)^2)$. We assume that each component q_i follows a mixture of two Gaussian distributions. The resulting target distribution has

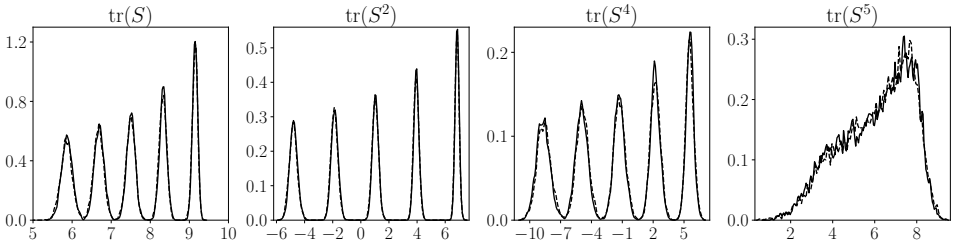


Fig. 5.3: Results for Transformed-SO(10). Empirical densities of the statistics $\mathrm{tr}(S), \mathrm{tr}(S^2), \mathrm{tr}(S^4)$, and $\mathrm{tr}(S^5)$ for the samples generated by our method (dashed line) and samples from the target distribution (solid line).

in total 2^{n_0} modes. Figure 5.4 shows the empirical densities of components q_1, q_2, p_1, and p_2. The solid and dashed lines compare the distributions of the forward and the learned reverse processes at different steps k. The overlap of the black lines at $k=0$ shows that the generated samples match well with the target distribution.

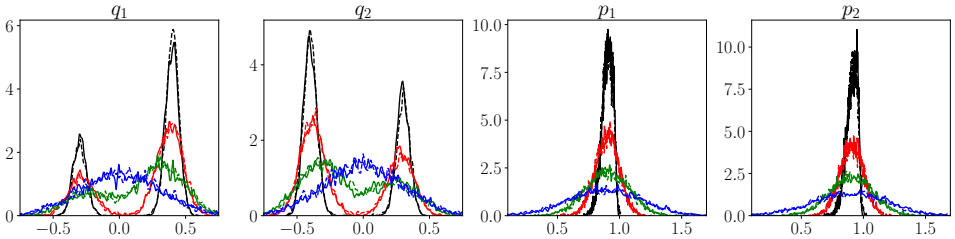


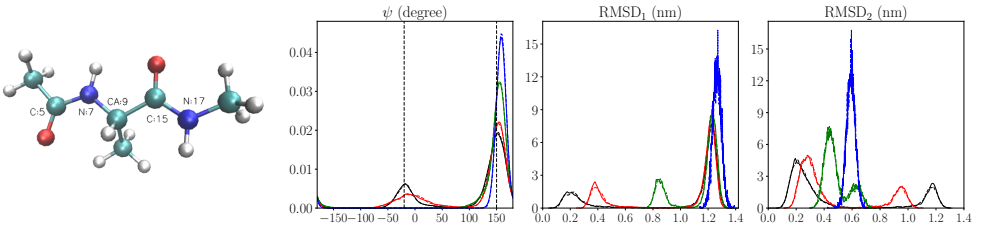
Fig. 5.4: Results for the conserved Hamiltonian surface. Empirical densities of q_1, q_2, p_1, and p_2 for the forward process (solid line) and the learned reverse process (dashed line) at different steps $k=0,20,40,150$, colored in black, red, green, and blue, respectively.

5.4. Alanine dipeptide. We apply our method to alanine dipeptide, a commonly studied model system in bio-physics. The configuration of the system can be characterized by its two dihedral angles ϕ and ψ (see Figure 5.5a). In this study, we are interested in the configurations of the 10 non-hydrogen atoms of the system (in $\mathbb{R}^{30}$) with the fixed angle $\phi=-70°$.

In the forward process, b is chosen as $-\nabla V$, where V is proportional to the root mean squared deviation (RMSD) from a pre-selected reference configuration x^{ref}. Consequently, the prior distribution $p(x^{(N)})$ is a single-well distribution centered at x^{ref}. Furthermore, we model $s^{(k+1),\theta}(x)$ in the reverse process using a network that preserves rotational equivariance and translational invariance. This design, together with our choice of b, guarantees that the distribution $p_\theta(x^{(0)})$ generated by our model is invariant under SE(3) [35]. We refer to Appendix B.4 for further experimental details and to Appendix C for theoretical support.

We employ three metrics to assess the quality of the generated configurations: the angle ψ, and two RMSDs (denoted by RMSD_1 and RMSD_2) with respect to two pre-defined reference configurations that are selected from two different wells. Figure 5.5b illustrates the empirical densities of these three metrics for the configurations generated

by our model and the configurations in the dataset. The solid and dashed lines show the agreement between the distributions of the learned reverse process and the distributions of the forward process at different time steps k. In particular, the overlap between the lines in black, which correspond to step $k=0$, demonstrates that the distribution of the generated samples (dashed) closely matches the data distribution (solid).



(a) Alanine dipeptide (b) Results for alanine dipeptide

Fig. 5.5: (a) Illustration of the system. Names and 1-based indices are shown for atoms that define the dihedral angles. The dihedral angles ϕ and ψ are defined by atoms whose 1-based indices are 5,7,9,15 and 7,9,15,17, respectively. (b) Empirical densities of the angle ψ, RMSD_1, and RMSD_2 for the forward process (solid line) and the learned reverse process (dashed line) at steps $k=0,10,40,200$, colored in black, red, green, blue, respectively. The ψ values of the two reference points that are used to define RMSD_1 and RMSD_2 are $-20°$ and $150°$, respectively (as shown by the two vertical dashed lines in the left panel).

6. Conclusion We have proposed Riemannian Denoising Diffusion Probabilistic Models for generative modeling on submanifolds. Our method does not rely on sophisticated geometric objects on manifold and it is applicable to high-dimensional manifolds with nontrivial geometry. We have provided a theoretical analysis of our method in the continuous-time limit, which elucidates its connection to Riemannian score-based generative models. We have demonstrated the strong capability of our method on manifolds from previous studies, as well as on those with complex geometries that can not be easily explored by existing methods.

Acknowledgment. Zichen Liu acknowledges support from the China Scholarship Council under Grant No. 202306010047. Wei Zhang is funded by the Eigene Stelle of Deutsche Forschungsgemeinschaft (DFG), project No. 524086759. Christof Schütte acknowledges funding by DFG under Germany's Excellence Strategy MATH+: The Berlin Mathematics Research Centre (EXC-2046/1, project No. 390685689) and CRC 1114 "Scaling Cascades in Complex Systems" (project No. 235221301). Tiejun Li acknowledges the support from National Key R&D Program of China under grant 2021YFA1003301, and National Science Foundation of China under grant 12288101.

Appendix A. Proofs of the continuous-time limit.

In this section, we prove Theorem 4.1 and Corollary 4.1 in Section 4. For simplicity of notation, we denote by ∂_i the derivative with respect to x_i in the ambient space, and by I the identity matrix of order n. We use subscripts to denote components of a vector and entries of a matrix. Also recall that the orthogonal projection matrix $P(x)\in\mathbb{R}^{n\times n}$

is well defined for $x\in\mathbb{R}^n$ and has the expression

$$P_{ij}(x)=\delta_{ij}-\sum_{\alpha,\alpha'=1}^{n-d}\partial_i\xi_\alpha(x)(\nabla\xi^\top\nabla\xi)^{-1}_{\alpha\alpha'}(x)\partial_j\xi_{\alpha'}(x),\quad 1\le i,j\le n, \tag{A.1}$$

where δ_{ij} is the Dirac delta function. First, we present the proof of Theorem 4.1.

Proof. **(Proof of Theorem 4.1.)** Let us write the forward process in (3.5) as

$$x^{(k+1)}=x^{(k+\frac{1}{2})}+\nabla\xi(x^{(k)})c(x^{(k+\frac{1}{2})}),$$

where $x^{(k+\frac{1}{2})}=x^{(k)}+\sigma_k^2 b(x^{(k)})+\sigma_k v^{(k)}$ and the dependence of c on $x^{(k+\frac{1}{2})}$ is made explicit. Applying Lemma A.1 at the end of this section, we obtain the expansion, for $1\le i\le n$,

$$\begin{aligned}
&x_i^{(k+1)} \qquad\qquad (A.2)\\
=&x_i^{(k)}+\sum_{j=1}^n P_{ij}(x^{(k)})\Big(\sigma_k^2 b_j(x^{(k)})+\sigma_k v_j^{(k)}\Big)\\
&+\frac{1}{2}\sum_{j,l,r,r'=1}^n\Big((I-P)_{ir}P_{r'l}\,\partial_{r'}P_{rj}\Big)(x^{(k)})\Big(\sigma_k^2 b_j(x^{(k)})+\sigma_k v_j^{(k)}\Big)\Big(\sigma_k^2 b_l(x^{(k)})+\sigma_k v_l^{(k)}\Big)\\
&+\frac{1}{6}\sum_{j,l,r=1}^n\sum_{\eta=1}^{n-d}\big(\partial_i\xi_\eta\,\partial^3_{jlr}c_\eta\big)(x^{(k)})\Big(x_j^{(k+\frac{1}{2})}-x_j^{(k)}\Big)\Big(x_l^{(k+\frac{1}{2})}-x_l^{(k)}\Big)\Big(x_r^{(k+\frac{1}{2})}-x_r^{(k)}\Big)\\
&+o\big(|x^{(k+\frac{1}{2})}-x^{(k)}|^3\big)\\
=&x_i^{(k)}+\sigma_k v_i^{(k)}+\sigma_k^2\sum_{j=1}^n P_{ij}(x^{(k)})b_j(x^{(k)})+\frac{\sigma_k^2}{2}\sum_{j,r,r'=1}^n\big((I-P)_{ir}\,\partial_{r'}P_{rj}\big)(x^{(k)})v_j^{(k)}v_{r'}^{(k)}\\
&+\sigma_k^3 R_i^{(k)}+o(\sigma_k^3),
\end{aligned}\tag{A.3}$$

where we have used the identity $\sum_{j=1}^n P_{ij}(x^{(k)})v_j^{(k)}=v_i^{(k)}$ (since $v^{(k)}$ is a tangent vector), and $R_i^{(k)}$ is a term that satisfies $\sum_{i'=1}^n P_{ii'}(x^{(k)})R_{i'}^{(k)}=0$, for $1\le i\le n$.

With the expansion above, we compute the loss function in (3.10). Using (A.2), the relation $\beta_{k+1}=\sigma_k=\sqrt{h}g(kh)$, and the assumption that $s^{(k+1),\theta}(x^{(k+1)})=s_\theta(x^{(k+1)},(k+1)h)\in\mathbb{R}^n$, we can derive

$$\begin{aligned}
&\beta_{k+1}^2\Big|P(x^{(k+1)})\Big(s^{(k+1),\theta}(x^{(k+1)})-b(x^{(k+1)})+\frac{x^{(k+1)}-x^{(k)}}{\beta_{k+1}^2}\Big)\Big|^2\\
=&\sigma_k^2\sum_{i=1}^n\bigg|\sum_{i'=1}^n P_{ii'}(x^{(k+1)})\Big[s_{\theta,i'}(x^{(k+1)},(k+1)h)-b_{i'}(x^{(k+1)})+\sum_{j=1}^n P_{i'j}(x^{(k)})b_j(x^{(k)})\\
&\quad+\frac{1}{2}\sum_{j,r,r'=1}^n\big((I-P)_{i'r}\,\partial_{r'}P_{rj}\big)(x^{(k)})v_j^{(k)}v_{r'}^{(k)}+\frac{v_{i'}^{(k)}}{\sigma_k}+\sigma_k R_{i'}^{(k)}+o(\sigma_k)\Big]\bigg|^2\\
=&\mathcal{I}_1+\mathcal{I}_2+\mathcal{I}_3+o(\sigma_k^2),
\end{aligned}\tag{A.4}$$

where the three terms on the last line are defined as

$$\mathcal{I}_1:=\sigma_k^2\sum_{i=1}^n\bigg|\sum_{i'=1}^n P_{ii'}(x^{(k+1)})\Big[s_{\theta,i'}(x^{(k+1)},(k+1)h)-b_{i'}(x^{(k+1)})+\sum_{j=1}^n P_{i'j}(x^{(k)})b_j(x^{(k)})$$

$$+\frac{1}{2}\sum_{j,r,r'=1}^{n}\big((I-P)_{i'r}\partial_{r'}P_{rj}\big)(x^{(k)})v_j^{(k)}v_{r'}^{(k)}+\sigma_k R_{i'}^{(k)}\Big]\Big|^2,$$

$$\mathcal{I}_2:=\sum_{i=1}^{n}\Big(\sum_{i'=1}^{n}P_{ii'}(x^{(k+1)})v_{i'}^{(k)}\Big)^2,$$

$$\mathcal{I}_3:=2\sigma_k\sum_{i,i'=1}^{n}P_{ii'}(x^{(k+1)})\Big[s_{\theta,i'}(x^{(k+1)},(k+1)h)-b_{i'}(x^{(k+1)})+\sum_{j=1}^{n}P_{i'j}(x^{(k)})b_j(x^{(k)})$$
$$+\frac{1}{2}\sum_{j,r,r'=1}^{n}\big((I-P)_{i'r}\partial_{r'}P_{rj}\big)(x^{(k)})v_j^{(k)}v_{r'}^{(k)}+\sigma_k R_{i'}^{(k)}\Big]v_i^{(k)},$$

respectively. In the following, we derive the expansions of the three terms above. For $\mathcal{I}_1$, expanding the functions P, s_θ, b using (A.2), we can derive

$$\begin{aligned}\mathcal{I}_1=&\sigma_k^2\sum_{i=1}^{n}\Big|\sum_{i'=1}^{n}P_{ii'}(x^{(k)})\Big[s_{\theta,i'}(x^{(k)},kh)-b_{i'}(x^{(k)})+\sum_{j=1}^{n}P_{i'j}(x^{(k)})b_j(x^{(k)})\\&+\frac{1}{2}\sum_{j,r,r'=1}^{n}\big((I-P)_{i'r}\partial_{r'}P_{rj}\big)(x^{(k)})v_j^{(k)}v_{r'}^{(k)}\Big]+o(1)\Big|^2\\=&\sigma_k^2\sum_{i=1}^{n}\Big|\sum_{i'=1}^{n}P_{ii'}(x^{(k)})s_{\theta,i'}(x^{(k)},kh)\Big|^2+o(\sigma_k^2),\end{aligned}\tag{A.5}$$

where we have used the relations $P^2=P$ and $P(I-P)=0$ satisfied by the orthogonal projection matrix P to derive the second equality. For $\mathcal{I}_2$, using the relation $P^2=P$ and (A.2), we can compute

$$\begin{aligned}\mathcal{I}_2=&\sum_{i,i'=1}^{n}P_{ii'}(x^{(k+1)})v_i^{(k)}v_{i'}^{(k)}\\=&\sum_{i,i'=1}^{n}P_{ii'}(x^{(k)})v_i^{(k)}v_{i'}^{(k)}+\sum_{i,i',r=1}^{n}\partial_r P_{ii'}(x^{(k)})\big(x_r^{(k+1)}-x_r^{(k)}\big)v_i^{(k)}v_{i'}^{(k)}\\&+\frac{1}{2}\sum_{i,i'=1}^{n}\sum_{r,r'=1}^{n}\partial_{rr'}^2P_{ii'}(x^{(k)})v_i^{(k)}v_{i'}^{(k)}(x_r^{(k+1)}-x_r^{(k)})(x_{r'}^{(k+1)}-x_{r'}^{(k)})\\&+o(|x^{(k+1)}-x^{(k)}|^2)\\=&|v^{(k)}|^2+\sum_{i,i',r=1}^{n}\partial_r P_{ii'}(x^{(k)})\big(x_r^{(k+1)}-x_r^{(k)}\big)v_i^{(k)}v_{i'}^{(k)}\\&+\frac{\sigma_k^2}{2}\sum_{i,i'=1}^{n}\sum_{r,r'=1}^{n}\partial_{rr'}^2P_{ii'}(x^{(k)})v_i^{(k)}v_{i'}^{(k)}v_r^{(k)}v_{r'}^{(k)}+o(\sigma_k^2).\end{aligned}\tag{A.6}$$

Let's compute the three terms in (A.6). Using the expression of $P_{ii'}$ in (A.1), the fact that $\sum_{i=1}^{n}\partial_i\xi_\alpha(x^{(k)})v_i^{(k)}=0$, and the product rule, it is straightforward to verify that, for $1\le r\le n$,

$$\sum_{i,i'=1}^{n}\partial_r P_{ii'}(x^{(k)})v_i^{(k)}v_{i'}^{(k)}=-\sum_{i,i'=1}^{n}\partial_r\Big(\sum_{\alpha,\alpha'=1}^{n-d}\partial_i\xi_\alpha(\nabla\xi^\top\nabla\xi)^{-1}_{\alpha\alpha'}\partial_{i'}\xi_{\alpha'}\Big)(x^{(k)})v_i^{(k)}v_{i'}^{(k)}=0.\tag{A.7}$$

Similarly, we can verify that

$$\begin{aligned}&\sum_{i,i',r,r'=1}^{n}\partial^2_{rr'}P_{ii'}(x^{(k)})v_i^{(k)}v_{i'}^{(k)}v_r^{(k)}v_{r'}^{(k)}\\ =&-2\sum_{i,i',r,r'=1}^{n}\sum_{\alpha,\alpha'=1}^{n-d}\Big(\partial^2_{ir}\xi_\alpha(\nabla\xi^\top\nabla\xi)^{-1}_{\alpha\alpha'}\partial^2_{i'r'}\xi_{\alpha'}\Big)(x^{(k)})v_i^{(k)}v_{i'}^{(k)}v_r^{(k)}v_{r'}^{(k)}.\end{aligned}\tag{A.8}$$

Hence, substituting (A.7) and (A.8) into (A.6), we obtain

$$\mathcal{I}_2=|v^{(k)}|^2-\sigma_k^2\sum_{i,i',r,r'=1}^{n}\sum_{\alpha,\alpha'=1}^{n-d}\Big(\partial^2_{ir}\xi_\alpha(\nabla\xi^\top\nabla\xi)^{-1}_{\alpha\alpha'}\partial^2_{i'r'}\xi_{\alpha'}\Big)(x^{(k)})v_i^{(k)}v_{i'}^{(k)}v_r^{(k)}v_{r'}^{(k)}+o(\sigma_k^2).\tag{A.9}$$

For $\mathcal{I}_3$, we have

$$\begin{aligned}\mathcal{I}_3=&2\sigma_k\sum_{i,i'=1}^{n}P_{ii'}(x^{(k)})\Big[s_{\theta,i'}(x^{(k)},kh)-b_{i'}(x^{(k)})+\sum_{j=1}^{n}P_{i'j}(x^{(k)})b_j(x^{(k)})\\&+\frac{1}{2}\sum_{j,r,r'=1}^{n}\big((I-P)_{i'r}\partial_{r'}P_{rj}\big)(x^{(k)})v_j^{(k)}v_{r'}^{(k)}+\sigma_kR_{i'}^{(k)}\Big]v_i^{(k)}\\&+2\sigma_k^2\sum_{i,i',r=1}^{n}\partial_r(P_{ii'}(s_{\theta,i'}-b_{i'}))(x^{(k)},kh)v_r^{(k)}v_i^{(k)}\\&+2\sigma_k^2\sum_{i,i',r,j=1}^{n}\partial_rP_{ii'}(x^{(k)})P_{i'j}(x^{(k)})b_j(x^{(k)})v_r^{(k)}v_i^{(k)}\\&+\sigma_k^2\sum_{i,i',r,j,r',j'=1}^{n}\partial_rP_{ii'}(x^{(k)})\big((I-P)_{i'j'}\partial_{r'}P_{j'j}\big)(x^{(k)})v_j^{(k)}v_{r'}^{(k)}v_r^{(k)}v_i^{(k)}+o(\sigma_k^2)\\=&2\sigma_k\sum_{i,i'=1}^{n}P_{ii'}(x^{(k)})s_{\theta,i'}(x^{(k)},kh)v_i^{(k)}+2\sigma_k^2\sum_{i,i',r=1}^{n}\partial_r(P_{ii'}(s_{\theta,i'}-b_{i'}))(x^{(k)},kh)v_r^{(k)}v_i^{(k)}\\&+2\sigma_k^2\sum_{i,i',r,j=1}^{n}\partial_rP_{ii'}(x^{(k)})P_{i'j}(x^{(k)})b_j(x^{(k)})v_r^{(k)}v_i^{(k)}\\&+\sigma_k^2\sum_{i,i',r,j,r',j'=1}^{n}\partial_rP_{ii'}(x^{(k)})\big((I-P)_{i'j'}\partial_{r'}P_{j'j}\big)(x^{(k)})v_j^{(k)}v_{r'}^{(k)}v_r^{(k)}v_i^{(k)}+o(\sigma_k^2),\end{aligned}\tag{A.10}$$

where we have used Taylor expansion with (A.2) and the fact that $|s_\theta(x,(k+1)h)-s_\theta(x,kh)|=O(h)=O(\sigma_k^2)$ to derive the first equality, and we have used the relations $P^2=P$, $P(I-P)=0$, and $\sum_{i'=1}^{n}P_{ii'}(x^{(k)})R_{i'}^{(k)}=0$ to derive the second equality. We further simplify the last two terms in the expression above. Notice that, similar to (A.7), we can verify that

$$\sum_{i,i',r=1}^{n}\partial_rP_{ii'}(x^{(k)})P_{i'j}(x^{(k)})v_r^{(k)}v_i^{(k)}=0,\quad 1\le j\le n.\tag{A.11}$$

For the last term in (A.10), we can derive

$$\begin{aligned}
&\sum_{i,i',r,j,r',j'=1}^{n} \partial_r P_{ii'}(x^{(k)})\big((I-P)_{i'j'}\,\partial_{r'}P_{j'j}\big)(x^{(k)})v_j^{(k)}v_{r'}^{(k)}\,v_r^{(k)}v_i^{(k)}\\
=&\sum_{i',r,j,r',j'=1}^{n} \big(\partial_r P_{i'j'}\,\partial_{r'}P_{j'j}\big)(x^{(k)})v_j^{(k)}v_{r'}^{(k)}\,v_r^{(k)}v_{i'}^{(k)}\\
=&\sum_{i',r,j,r',j'=1}^{n}\Bigg[\sum_{\alpha,\alpha',\eta,\eta'=1}^{n-d}\Big(\partial^2_{i'r}\xi_\alpha(\nabla\xi^\top\nabla\xi)^{-1}_{\alpha\alpha'}\partial_{j'}\xi_{\alpha'}\Big)\Big(\partial^2_{jr'}\xi_\eta(\nabla\xi^\top\nabla\xi)^{-1}_{\eta\eta'}\partial_{j'}\xi_{\eta'}\Big)\Bigg](x^{(k)})\\
&\qquad\cdot v_j^{(k)}v_{r'}^{(k)}v_r^{(k)}v_{i'}^{(k)}\\
=&\sum_{i',r,j,r'=1}^{n}\Bigg[\sum_{\alpha,\eta=1}^{n-d}\Big(\partial^2_{i'r}\xi_\alpha(\nabla\xi^\top\nabla\xi)^{-1}_{\alpha\eta}\partial^2_{jr'}\xi_\eta\Big)\Bigg](x^{(k)})v_j^{(k)}v_{r'}^{(k)}\,v_r^{(k)}v_{i'}^{(k)},
\end{aligned}\tag{A.12}$$

where the first equation follows by applying the product rule to the identity $P(I-P)=0$ and using the relation $\sum_{i=1}^n P_{ii'}v_i^{(k)}=v_{i'}^{(k)}$, the second equation follows from the expression (A.1) and the fact that several terms vanish due to the orthogonality relation $\sum_{i=1}^n\partial_i\xi_\alpha(x^{(k)})v_i^{(k)}=0$, and the last equation follows from the fact that $\sum_{j'=1}^n\partial_{j'}\xi_{\alpha'}\partial_{j'}\xi_{\eta'}=(\nabla\xi^\top\nabla\xi)_{\alpha'\eta'}$. Combining (A.10), (A.11), and (A.12), we obtain

$$\begin{aligned}
\mathcal{I}_3=&2\sigma_k\sum_{i,i'=1}^{n}P_{ii'}(x^{(k)})s_{\theta,i'}(x^{(k)},kh)v_i^{(k)}+2\sigma_k^2\sum_{i,i',r=1}^{n}\partial_r(P_{ii'}(s_{\theta,i'}-b_{i'}))(x^{(k)},kh)v_r^{(k)}v_i^{(k)}\\
&+\sigma_k^2\sum_{i',r,j,r'=1}^{n}\Bigg[\sum_{\alpha,\eta=1}^{n-d}\Big(\partial^2_{i'r}\xi_\alpha(\nabla\xi^\top\nabla\xi)^{-1}_{\alpha\eta}\partial^2_{jr'}\xi_\eta\Big)\Bigg](x^{(k)})v_j^{(k)}v_{r'}^{(k)}\,v_r^{(k)}v_{i'}^{(k)}+o(\sigma_k^2).
\end{aligned}\tag{A.13}$$

Substituting (A.5), (A.9), and (A.13) into (A.4), we obtain (after cancellation of terms in $\mathcal{I}_2$ and $\mathcal{I}_3$)

$$\begin{aligned}
&\beta_{k+1}^2\left|P(x^{(k+1)})\Big(s^{(k+1),\theta}(x^{(k+1)})-b(x^{(k+1)})+\frac{x^{(k+1)}-x^{(k)}}{\beta_{k+1}^2}\Big)\right|^2\\
=&\mathcal{I}_1+\mathcal{I}_2+\mathcal{I}_3+o(\sigma_k^2)\\
=&\sigma_k^2\big|P(x^{(k)})s_\theta(x^{(k)},kh)\big|^2+\big|v^{(k)}\big|^2+2\sigma_k^2\sum_{i,i',r=1}^{n}\partial_r(P_{ii'}(s_{\theta,i'}-b_{i'}))(x^{(k)},kh)v_r^{(k)}v_i^{(k)}\\
&+2\sigma_k\sum_{i,i'=1}^{n}P_{ii'}(x^{(k)})s_{\theta,i'}(x^{(k)},kh)v_i^{(k)}+o(\sigma_k^2).
\end{aligned}\tag{A.14}$$

Now, we consider the terms on the right hand side of the above expression in the limit $N\to+\infty$. Using the fact that the forward process $x^{(k)}$ converges to the SDE (4.1) [4] , we have

$$\lim_{N\to+\infty}\mathbb{E}_{\mathbb{Q}^{(N)}}\Big(\sum_{k=0}^{N-1}\sigma_k^2\big|P(x^{(k)})s_\theta(x^{(k)},kh)\big|^2\Big)=\mathbb{E}_{\mathbb{Q}}\int_0^T\Big(\big|P(X_t)s_\theta(X_t,t)\big|^2\Big)g^2(t)dt.$$

Since $v^{(k)}$ is the standard Gaussian random variable in $T_{x^{(k)}}\mathcal{M}$ confined in $\mathcal{F}_{x^{(k)}}^{(\sigma_k)}$ (the set of tangent vectors with which (3.5) has a solution), the set $\mathcal{F}_{x^{(k)}}^{(\sigma_k)}$ increases to the

entire tangent space $T_{x^{(k)}}\mathcal{M}$ as $N\to+\infty$, we can show that (by bounded convergence theorem)

$$\lim_{N\to+\infty}\mathbb{E}_{\mathbb{Q}^{(N)}}\Big(\sigma_k\sum_{i,i'=1}^{n}P_{ii'}(x^{(k)})s_{\theta,i'}(x^{(k)},kh)v_i^{(k)}\Big)=0,$$

and

$$\begin{aligned}&\lim_{N\to+\infty}\mathbb{E}_{\mathbb{Q}^{(N)}}\Big(\sigma_k^2\sum_{i,i',r=1}^{n}\partial_r(P_{ii'}(s_{\theta,i'}-b_{i'}))(x^{(k)},kh)v_r^{(k)}v_i^{(k)}\Big)\\ =&\mathbb{E}_{\mathbb{Q}}\bigg[\int_0^T\Big(\sum_{i,i',r=1}^{n}\partial_r(P_{ii'}(s_{\theta,i'}-b_{i'}))(X_t,t)P_{ri}(X_t)\Big)g^2(t)dt\bigg]\\ =&\mathbb{E}_{\mathbb{Q}}\bigg[\int_0^T\Big(\mathrm{div}_{\mathcal{M}}\big(P(s_\theta-b)\big)(X_t,t)\Big)g^2(t)dt\bigg],\end{aligned}$$

where the equality above can be verified using the fact that $v^{(k)}=P(x^{(k)})z^{(k)}$ and $z^{(k)}$ converges to a standard Gaussian random variable in $\mathbb{R}^n$ as $N\to+\infty$.

Substituting (A.14) into (3.10) and using the above equations, we can derive

$$\begin{aligned}&\lim_{N\to+\infty}\Big(\mathrm{Loss}^{(N)}(\theta)-\frac{1}{2}\sum_{k=0}^{N-1}\big|v^{(k)}\big|^2\Big)\\ =&\mathbb{E}_{\mathbb{Q}}\int_0^T\bigg[\frac{1}{2}\big|P(X_t)s_\theta(X_t,t)\big|^2+\mathrm{div}_{\mathcal{M}}\big(P(s_\theta-b)\big)(X_t,t)\bigg]g^2(t)dt\\ =&\int_0^T\bigg[\int_{\mathcal{M}}\Big(\frac{1}{2}\big|P(x)s_\theta(x,t)\big|^2+\mathrm{div}_{\mathcal{M}}\big(P(s_\theta-b)\big)(x,t)\Big)p(x,t)\,d\sigma_{\mathcal{M}}(x)\bigg]g^2(t)\,dt\\ =&\int_0^T\bigg[\int_{\mathcal{M}}\Big(\frac{1}{2}\big|P(x)s_\theta(x,t)\big|^2-(P(s_\theta-b))(x,t)\cdot\nabla_{\mathcal{M}}\log p(x,t)\Big)p(x,t)\,d\sigma_{\mathcal{M}}(x)\bigg]g^2(t)\,dt\\ =&\frac{1}{2}\int_0^T\bigg[\int_{\mathcal{M}}\big|P(x)s_\theta(x,t)-\nabla_{\mathcal{M}}\log p(x,t)\big|^2p(x,t)\,d\sigma_{\mathcal{M}}(x)\bigg]g^2(t)\,dt\\ &+\int_0^T\bigg[\int_{\mathcal{M}}\Big(\big(P(x)b(x)-\frac{1}{2}\nabla_{\mathcal{M}}\log p(x,t)\big)\cdot\nabla_{\mathcal{M}}\log p(x,t)\Big)p(x,t)\,d\sigma_{\mathcal{M}}(x)\bigg]g^2(t)\,dt\\ =&\mathbb{E}_{\mathbb{Q}}\bigg[\frac{1}{2}\int_0^T\big|P(X_t)s_\theta(X_t,t)-\nabla_{\mathcal{M}}\log p(X_t,t)\big|^2g^2(t)\,dt\\ &+\int_0^T\Big(P(X_t)b(X_t)-\frac{1}{2}\nabla_{\mathcal{M}}\log p(X_t,t)\Big)\cdot\nabla_{\mathcal{M}}\log p(X_t,t)g^2(t)\,dt\bigg],\end{aligned}$$

where we have used integration by parts on $\mathcal{M}$, and the expression $\mathrm{div}_{\mathcal{M}}f=\sum_{i,r=1}^{n}P_{ir}\partial_r f_i$ for $f:\mathcal{M}\to\mathbb{R}^n$ (which can be verified using Lemma A.1 in [37]). ∎

Next, we present the proof of Corollary 4.1.

Proof. **(Proof of Corollary 4.1.)** Using the assumption $\beta_{k+1}=\sigma_k$, the projection scheme in (3.5) and the relation $P(x^{(k)})\nabla\xi(x^{(k)})=0$, we can simplify the constant $C^{(N)}$ in (3.11) as

$$C^{(N)}=-\mathbb{E}_{\mathbb{Q}^{(N)}}\Big(\log p(x^{(N)})+\frac{1}{2}\sum_{k=0}^{N-1}|v^{(k)}|^2+\sum_{k=0}^{N-1}\log\big(1-\epsilon_{x^{(k)}}^{(\sigma_k)}\big)\Big).\tag{A.15}$$

Therefore, using the definition of relative entropy (see (2.4)), the loss function in (3.10), the constant $C^{(N)}$ in (3.11), and applying Theorem 4.1, we have

$$
\begin{aligned}
&\lim_{N\to+\infty} H\big(\overleftarrow{\mathbb{Q}}^{(N)}\,|\,\mathbb{P}_\theta^{(N)}\big)\\
=&\lim_{N\to+\infty}\left[\mathrm{Loss}^{(N)}(\theta)+C^{(N)}+\mathbb{E}_{\mathbb{Q}^{(N)}}\Big(\sum_{k=0}^{N-1}\log\big(1-\epsilon^{(\sigma_k)}_{x^{(k+1)},\theta}\big)\Big)\right]+\mathbb{E}_{q_0}(\log q_0)\\
=&\lim_{N\to+\infty}\left[\mathrm{Loss}^{(N)}(\theta)-\mathbb{E}_{\mathbb{Q}^{(N)}}\left(\log p(x^{(N)})+\frac{1}{2}\sum_{k=0}^{N-1}|v^{(k)}|^2+\sum_{k=0}^{N-1}\log\frac{1-\epsilon^{(\sigma_k)}_{x^{(k)}}}{1-\epsilon^{(\sigma_k)}_{x^{(k+1)},\theta}}\right)\right]\\
&+\mathbb{E}_{q_0}(\log q_0)\\
=&\lim_{N\to+\infty}\Big(\mathrm{Loss}^{(N)}(\theta)-\mathbb{E}_{\mathbb{Q}^{(N)}}\log p(x^{(N)})-\frac{1}{2}\sum_{k=0}^{N-1}|v^{(k)}|^2\Big)+\mathbb{E}_{q_0}(\log q_0)\\
=&\mathbb{E}_{\mathbb{Q}}\Big[\log p(X_0,0)-\log p(X_T,T)+\frac{1}{2}\int_0^T\big|P(X_t)s_\theta(X_t,t)-\nabla_{\mathcal{M}}\log p(X_t,t)\big|^2 g^2(t)\,dt\\
&+\int_0^T\Big(P(X_t)b(X_t)-\frac{1}{2}\nabla_{\mathcal{M}}\log p(X_t,t)\Big)\cdot\nabla_{\mathcal{M}}\log p(X_t,t)\,g^2(t)\,dt\Big],
\end{aligned}
\tag{A.16}
$$

where the third equality follows because the terms containing $\epsilon^{(\sigma_k)}_{x^{(k)}}$ and $\epsilon^{(\sigma_k)}_{x^{(k+1)},\theta}$ converge to zero sufficiently fast as $N\to+\infty$. Note that the density $p(x,t)$ of SDE (4.1) solves the Fokker-Planck equation

$$
\frac{\partial p}{\partial t}(x,t)=-g^2(t)\mathrm{div}_{\mathcal{M}}\big(P(x)b(x)p(x,t)\big)+\frac{g^2(t)}{2}\Delta_{\mathcal{M}}p(x,t),\quad x\in\mathcal{M},\quad t\in[0,T].
\tag{A.17}
$$

Therefore, we have

$$
\begin{aligned}
&\mathbb{E}_{\mathbb{Q}}\Big[\log p(X_0,0)-\log p(X_T,T)\Big]\\
=&\int_{\mathcal{M}}\log p(x,0)p(x,0)\,d\sigma_{\mathcal{M}}(x)-\int_{\mathcal{M}}\log p(x,T)p(x,T)\,d\sigma_{\mathcal{M}}(x)\\
=&-\int_0^T\frac{d}{dt}\Big(\int_{\mathcal{M}}\log p(x,t)\,p(x,t)\,d\sigma_{\mathcal{M}}(x)\Big)\,dt\\
=&-\int_0^T\left[\int_{\mathcal{M}}\big(\log p(x,t)+1\big)\frac{\partial p}{\partial t}(x,t)\,d\sigma_{\mathcal{M}}(x)\right]dt\\
=&-\int_0^T\left[\int_{\mathcal{M}}\big(\log p(x,t)+1\big)\Big(-\mathrm{div}_{\mathcal{M}}\big(P(x)b(x)p(x,t)\big)+\frac{1}{2}\Delta_{\mathcal{M}}p(x,t)\Big)g^2(t)d\sigma_{\mathcal{M}}(x)\right]dt\\
=&-\int_0^T\left[\int_{\mathcal{M}}\Big(\big(P(x)b(x)-\frac{1}{2}\nabla_{\mathcal{M}}\log p(x,t)\big)\cdot\nabla_{\mathcal{M}}\log p(x,t)\Big)p(x,t)\,d\sigma_{\mathcal{M}}(x)\right]g^2(t)\,dt\\
=&-\mathbb{E}_{\mathbb{Q}}\left[\int_0^T\Big(P(X_t)b(X_t)-\frac{1}{2}\nabla_{\mathcal{M}}\log p(X_t,t)\Big)\cdot\nabla_{\mathcal{M}}\log p(X_t,t)\,g^2(t)\,dt\right],
\end{aligned}
\tag{A.18}
$$

where we have used (A.17) to derive the fourth equality and integration by parts on $\mathcal{M}$ to derive the fifth equality. Combining (A.16) and (A.18), we obtain

$$
\lim_{N\to+\infty} H\big(\overleftarrow{\mathbb{Q}}^{(N)}\,|\,\mathbb{P}_\theta^{(N)}\big)=\mathbb{E}_{\mathbb{Q}}\left[\frac{1}{2}\int_0^T\big|P(X_t)s_\theta(X_t,t)-\nabla_{\mathcal{M}}\log p(X_t,t)\big|^2 g^2(t)\,dt\right].
\tag{A.19}
$$

Finally, note that $\overleftarrow{\mathbb{Q}}$ is the path measure of the time-reversal $Y_t = X_{T-t}$ of SDE (4.1), which satisfies [7, Theorem 3.1]

$$dY_t = g^2(T-t)\Big(-P(Y_t)b(X_t) + \nabla_{\mathcal{M}} \log p(Y_t, T-t)\Big)\,dt + g(T-t)dW_t^{\mathcal{M}}, \quad t \in [0,T], \tag{A.20}$$

and $\mathbb{P}_\theta$ is the path measure of SDE (4.3). Applying Girsanov's theorem [12, Theorem 8.1.2], we obtain

$$\begin{aligned}\frac{d\mathbb{P}_\theta}{d\overleftarrow{\mathbb{Q}}} = \exp\Big(&\int_0^T g^2(T-t)\big(P(Y_t)s_\theta(Y_t,T-t) - \nabla_{\mathcal{M}}\log p(Y_t,T-t)\big)\cdot dW_t^{\mathcal{M}} \\ &- \frac{1}{2}\int_0^T \big|P(Y_t)s_\theta(Y_t,T-t) - \nabla_{\mathcal{M}}\log p(Y_t,T-t)\big|^2 g^2(T-t)\,dt\Big),\end{aligned} \tag{A.21}$$

where $W_t^{\mathcal{M}}$ is a Brownian motion on $\mathcal{M}$ under $\overleftarrow{\mathbb{Q}}$. Therefore, we have

$$\begin{aligned}H(\overleftarrow{\mathbb{Q}}\,|\,\mathbb{P}_\theta) &= \mathbb{E}_{\overleftarrow{\mathbb{Q}}}\Big(\log\frac{d\overleftarrow{\mathbb{Q}}}{d\mathbb{P}_\theta}\Big) \\ &= \mathbb{E}_{\overleftarrow{\mathbb{Q}}}\Big(\frac{1}{2}\int_0^T \big|P(Y_t)s_\theta(Y_t,T-t) - \nabla_{\mathcal{M}}\log p(Y_t,T-t)\big|^2 g^2(T-t)\,dt\Big) \\ &= \mathbb{E}_{\mathbb{Q}}\Big(\frac{1}{2}\int_0^T \big|P(X_t)s_\theta(X_t,t) - \nabla_{\mathcal{M}}\log p(X_t,t)\big|^2 g^2(t)\,dt\Big),\end{aligned} \tag{A.22}$$

where the second equality follows from the fact that the stochastic integration in (A.21) vanishes after taking logarithm and expectation, and the third equality follows by a change of variable $t \leftarrow T-t$ and the fact that $Y_t = X_{T-t}$. The conclusion is obtained after combining (A.19) and (A.22). ∎

Finally, we present the technical lemma on the projection scheme in (3.1), which was used in the proof of Theorem 4.1.

LEMMA A.1. *Given $x \in \mathcal{M}$ and $x' \in \mathbb{R}^n$, the solution to the problem*

$$y = x' + \nabla\xi(x)c(x'), \quad c(x') \in \mathbb{R}^{n-d}, \quad s.t. \quad \xi(y) = 0 \tag{A.23}$$

has the following two expansions as x' approaches x

$$\partial_j c_\eta(x) = -\sum_{\alpha=1}^{n-d} (\nabla\xi^\top\nabla\xi)^{-1}_{\eta\alpha}(x)\partial_j\xi_\alpha(x), \quad 1 \le j \le n, \tag{A.24}$$

$$\partial^2_{jl} c_\eta(x) = \sum_{\alpha=1}^{n-d} (\nabla\xi^\top\nabla\xi)^{-1}_{\eta\alpha}(x) \sum_{r,r'=1}^{n} \Big(\partial_r\xi_\alpha \partial_{r'} P_{rj} P_{r'l}\Big)(x), \quad 1 \le j,l \le n, \tag{A.25}$$

for $1 \le \eta \le n-d$. Moreover, as x' approaches x, the following expansion of y in (A.23) holds

$$\begin{aligned}y_i = x_i &+ \sum_{j=1}^n P_{ij}(x)(x'_j - x_j) + \frac{1}{2}\sum_{j,l=1}^n \Big[\sum_{r,r'=1}^n \big((I-P)_{ir}P_{r'l}\partial_{r'}P_{rj}\big)(x)\Big](x'_j - x_j)(x'_l - x_l) \\ &+ \frac{1}{6}\sum_{j,l,r=1}^n \Big(\sum_{\eta=1}^{n-d} \partial_i\xi_\eta(x)\,\partial^3_{jlr} c_\eta(x)\Big)(x'_j - x_j)(x'_l - x_l)(x'_r - x_r) + o(|x'-x|^3),\end{aligned} \tag{A.26}$$

where $1\le i\le n$.

Proof. Differentiating (with respect to x') the constraint equation

$$\xi_\alpha(x'+\nabla\xi(x)c(x'))=0,\quad \alpha=1,\dots,n-d,$$

we get

$$\sum_{r=1}^{n}\partial_r\xi_\alpha\big(x'+\nabla\xi(x)c(x')\big)\Big(\delta_{rj}+\sum_{\eta=1}^{n-d}\partial_r\xi_\eta(x)\partial_j c_\eta(x')\Big)=0,\quad 1\le j\le n. \tag{A.27}$$

Setting $x'=x$ in (A.27) (notice that $c(x')=0$ when $x'=x$) and multiplying both sides by $(\nabla\xi^\top\nabla\xi)^{-1}(x)$, we obtain (A.24). In particular, using (A.1), we have

$$\delta_{rj}+\sum_{\eta=1}^{n-d}\partial_r\xi_\eta(x)\partial_j c_\eta(x)=\delta_{rj}-\sum_{\eta,\alpha=1}^{n-d}\Big(\partial_r\xi_\eta(\nabla\xi^\top\nabla\xi)^{-1}_{\eta\alpha}\partial_j\xi_\alpha\Big)(x)=P_{rj}(x),\quad 1\le r,j\le n. \tag{A.28}$$

Next, we show (A.25). Differentiating (A.27) again, setting $x'=x$ and using (A.28), we get, for $1\le\alpha\le n-d$ and $1\le j,l\le n$,

$$\begin{aligned}0=&\sum_{r,r'=1}^{n}\partial^2_{rr'}\xi_\alpha(x)\Big(\delta_{rj}+\sum_{\eta=1}^{n-d}\partial_r\xi_\eta(x)\partial_j c_\eta(x)\Big)\Big(\delta_{r'l}+\sum_{\eta=1}^{n-d}\partial_{r'}\xi_\eta(x)\partial_l c_\eta(x)\Big)\\&+\sum_{r=1}^{n}\partial_r\xi_\alpha(x)\Big(\sum_{\eta=1}^{n-d}\partial_r\xi_\eta(x)\partial^2_{jl}c_\eta(x)\Big)\\=&\sum_{r,r'=1}^{n}\Big(\partial^2_{rr'}\xi_\alpha P_{rj}P_{r'l}\Big)(x)+\sum_{\eta=1}^{n-d}\Big((\nabla\xi^\top\nabla\xi)_{\alpha\eta}\partial^2_{jl}c_\eta\Big)(x),\end{aligned}$$

from which we can solve, for $1\le\eta\le n-d$ and $1\le j,l\le n$,

$$\begin{aligned}\partial^2_{jl}c_\eta(x)=&-\sum_{\alpha=1}^{n-d}\sum_{r,r'=1}^{n}\Big((\nabla\xi^\top\nabla\xi)^{-1}_{\eta\alpha}\partial^2_{rr'}\xi_\alpha P_{rj}P_{r'l}\Big)(x)\\=&\sum_{\alpha=1}^{n-d}\sum_{r,r'=1}^{n}\Big((\nabla\xi^\top\nabla\xi)^{-1}_{\eta\alpha}\partial_r\xi_\alpha\partial_{r'}P_{rj}P_{r'l}\Big)(x),\end{aligned}$$

where the second equality follows from the product rule and the identity $\sum_{r=1}^{n}P_{rj}\partial_r\xi_\alpha=0$. This shows (A.25).

Lastly, we prove the expansion in (A.26). Note that (A.25) and (A.1) implies

$$\sum_{\eta=1}^{n-d}(\partial_i\xi_\eta\partial^2_{jl}c_\eta)(x)=\sum_{r,r'=1}^{n}\big((I-P)_{ir}P_{r'l}\partial_{r'}P_{rj}\big)(x),\quad 1\le i,j,l\le n. \tag{A.29}$$

By expanding $c(x')$ at $x'=x$ to the third order, noticing that $c(x)=0$, and using (A.28) and (A.29) for the first and second order derivatives respectively, we can derive

$$y_i=x'_i+\sum_{\eta=1}^{n-d}\partial_i\xi_\eta(x)c_\eta(x')$$

$$
\begin{aligned}
=&x_i+(x_i'-x_i)+\sum_{\eta=1}^{n-d}\partial_i\xi_\eta(x)\Bigg[\sum_{j=1}^{n}\partial_j c_\eta(x)(x_j'-x_j)+\frac{1}{2}\sum_{j,l=1}^{n}\partial_{jl}^2 c_\eta(x)(x_j'-x_j)(x_l'-x_l)\\
&+\frac{1}{6}\sum_{j,l,r=1}^{n}\partial_{jlr}^3 c_\eta(x)(x_j'-x_j)(x_l'-x_l)(x_r'-x_r)\Bigg]+o(|x'-x|^3)\\
=&x_i+\sum_{j=1}^{n}P_{ij}(x)(x_j'-x_j)+\frac{1}{2}\sum_{j,l=1}^{n}\Bigg[\sum_{r,r'=1}^{n}\Big((I-P)_{ir}\partial_{r'}P_{rj}P_{r'l}\Big)(x)\Bigg](x_j'-x_j)(x_l'-x_l)\\
&+\frac{1}{6}\sum_{j,l,r=1}^{n}\Bigg(\sum_{\eta=1}^{n-d}\partial_i\xi_\eta(x)\partial_{jlr}^3 c_\eta(x)\Bigg)(x_j'-x_j)(x_l'-x_l)(x_r'-x_r)+o(|x'-x|^3),
\end{aligned}
$$

which proves (A.26). ∎

Appendix B. Details of algorithms and experiments.

B.1. Neural networks and training setup. As described in Theorem 4.1, the functions $(s^{(k+1),\theta}(x))_{0\le k\le N-1}$ are represented by a single function $s_\theta(x,t)$ with parameter θ, which is in turn modeled by a multilayer perceptron (MLP). We employ SiLU as the activation function. We do not require that the output of the neural network belongs to the tangent space, thanks to the presence of the projection in both the forward and the reverse processes. Alternative strategies for designing neural networks with outputs in tangent space are proposed in [7].

Moreover, we define $g(t)$ as $g(t)=\gamma_{\min}+\frac{t}{T}(\gamma_{\max}-\gamma_{\min})$, where $\gamma_{\max}\ge\gamma_{\min}>0$. The parameters in the Markov chain are chosen as $\sigma_k=\beta_{k+1}=\sqrt{h}g(kh)$, with $h=\frac{T}{N}$ and $k=0,1,\dots,N-1$.

We train our models using PyTorch, where we employ the Adam optimizer with fixed learning rate $r=5\times10^{-4}$ and we clip the gradients of the parameters when the 2-norm exceeds 10. We also implement an exponential moving average for the model weights [24] with a decay rate of 0.999. All experiments are run on a single NVIDIA A40 GPU with 48G memory. In each run, the dataset is divided into training, validation, and test sets with ratio 80:10:10. Values of all the parameters in our experiments are summarized in Table B.1.

Datasets	$\gamma_{\min}$	$\gamma_{\max}$	N	T	l_{f}	N_{epoch}	B	N_{n}	N_{l}
Bunny, $k=50$	0.07	0.07	800	8.0	100	2000	2048	256	5
Bunny, $k=100$	0.07	0.07	500	5.0	100	2000	2048	256	5
Spot, $k=50$	0.1	0.1	500	5.0	100	2000	2048	256	5
Spot, $k=100$	0.1	0.1	300	3.0	100	2000	2048	256	5
SO(10)	0.2	2.0	500	1.0	100	2000	512	512	3
Transformed-SO(10)	0.2	1.3	250	2.5	100	1000	512	256	3
Hamiltonian	0.1	1.5	150	1.5	1	4000	512	256	3
Alanine dipeptide	1.0	1.0	200	0.1	100	5000	512	512	5

Table B.1: Parameters in our experiments. $\gamma_{\min},\gamma_{\max},N,T$ are the parameters in our model; $l_{\mathrm{f}},N_{\mathrm{epoch}},B$ are the parameters in Algorithm 4; $N_{\mathrm{n}},N_{\mathrm{l}}$ are the numbers of the hidden nodes per layer and the hidden layers of the neural networks, respectively.

B.2. Mesh data on learned manifolds. The function $\xi:\mathbb{R}^3\to\mathbb{R}$ is modeled by a MLP with 3 hidden layers, each of which has 128 nodes. Different from the activation function in our model, here we use Softplus activation function, where the parameter β is set to 10. The loss function for learning ξ is

$$\ell(\xi)=\frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}}|\xi(x)|+\frac{\lambda}{|\mathcal{D}'|}\sum_{y\in\mathcal{D}'}(|\nabla\xi(y)|-1)^2, \tag{B.1}$$

where $\lambda=0.1$, $\mathcal{D}$ denotes the set of vertices of a high-resolution mesh, and the set $\mathcal{D}'$ contains samples near the manifolds that are obtained by perturbing samples $x\in\mathcal{D}$ according to $y=x+c\epsilon$, with $\epsilon\sim\mathcal{N}(0,\ I_3)$ and $c=0.05$. The first term in (B.1) imposes that ξ is close to zero on vertices, whereas the second term serves as a regularization term and ensures that ξ has non-vanishing gradient near the manifold. The neural network is trained for 200000 steps using Adam optimizer, with batch size 512 and learning rate 10^{-4}.

With the learned function ξ, we consider the manifold defined by $\mathcal{M}=\{x\in\mathbb{R}^3|\xi(x)=0\}$. The values of ξ on the dataset are at the order 10^{-2}. To ensure that the data is on $\mathcal{M}$ with high precision, we refine the dataset by solving the following ordinary differential equation (ODE):

$$\frac{dx_t}{dt}=-\xi(x_t)\nabla\xi(x_t),\quad t\geq 0, \tag{B.2}$$

starting from each point in the dataset until the condition $|\xi(x_t)|<10^{-5}$ is reached (notice that (B.2) is a gradient flow and $\lim_{t\to\infty}|\xi(x_t)|=0$). This ensures that the refined points conform to the manifold accurately.

B.3. High-dimensional special orthogonal group. The dataset is constructed as a mixture of 5 wrapped normal distributions, each of which is the image (under the exponential map) of a normal distribution in the tangent space of a center $S_i\in\mathrm{SO}(10)$, $1\leq i\leq 5$. To ensure multimodality, we define the centers S_i as follows. We initially define a 2×2 matrix $A_0:=\begin{bmatrix}\cos\frac{\pi}{3} & \sin\frac{\pi}{3}\\ -\sin\frac{\pi}{3} & \cos\frac{\pi}{3}\end{bmatrix}$, which represents a rotation by $\frac{\pi}{3}$ radians. We then construct block diagonal matrices of order 10 by incorporating A_0 and the identity matrix I_2 in various combinations:

$$\begin{gathered}X_1=\mathrm{diag}\{A_0,I_2,I_2,I_2,I_2\},\quad X_2=\mathrm{diag}\{A_0,A_0,I_2,I_2,I_2\},\quad X_3=\mathrm{diag}\{A_0,A_0,A_0,I_2,I_2\},\\ X_4=\mathrm{diag}\{A_0,A_0,A_0,A_0,I_2\},\quad X_5=\mathrm{diag}\{A_0,A_0,A_0,A_0,A_0\}.\end{gathered} \tag{B.3}$$

The centers S_i of the wrapped normal distributions are chosen as $S_i=Q_i^\top X_iQ_i$, where $Q_i\in\mathrm{SO}(10)$ are randomly drawn from the uniform distribution. According to (B.3), the statistics $\eta(S)=(\mathrm{tr}(S),\mathrm{tr}(S^2),\mathrm{tr}(S^4),\mathrm{tr}(S^5))$ of the centers can be explicitly computed (using the trace identities $\mathrm{tr}(AB)=\mathrm{tr}(BA)$ and $\mathrm{tr}(Q_i^\top X_iQ_i)=\mathrm{tr}(X_i)$) as

$$\begin{gathered}\eta(S_1)=(9,7,7,9),\quad \eta(S_2)=(8,4,4,8),\quad \eta(S_3)=(7,1,1,7),\\ \eta(S_4)=(6,-2,-2,6),\quad \eta(S_5)=(5,-5,-5,5).\end{gathered} \tag{B.4}$$

To generate data in the dataset, we select a center S_i with equal probability, sample tangent vectors Y from the normal distribution (in the tangent space at S_i) with zero mean and standard deviation 0.05, and then compute their images S under the exponential map, that is, $S=S_i\mathrm{e}^{S_i^\top Y}$.

B.4. Alanine dipeptide. To generate the dataset, we initially perform a constrained molecular simulation of alanine dipeptide in water for 1ns using the molecular dynamics package GROMACS [33] with step size 1fs. We apply the harmonic biasing method in COLVARS module [8], where the collective variable is chosen as the dihedral angle ϕ and the harmonic potential is centered at $\phi = -70°$ with the force constant 5.0. Further simulation details are omitted since they are similar to those in [18]. In total, 10^4 configurations are obtained by recording every 100 simulation steps. We exclude the hydrogen atoms and work with the coordinates of the 10 non-hydrogen atoms in the system (see Figure 5.5a). In a final preparatory step, we apply the refinement technique in Appendix B.2 (see Eq. (B.2)) to the recorded coordinates, so that the data in the dataset lives in the manifold $\mathcal{M} = \{x \in \mathbb{R}^{30} | \phi(x) = -70°\}$ up to a small numerical error of order 10^{-5}.

Since $\mathcal{M}$ is unbounded, we adopt a nonzero function b in our model to make sure that the Markov chain processes stay in bounded region. To this end, we choose a reference configuration x^{ref} from the dataset and define the potential function

$$V(x) = \frac{\kappa}{2} |R_x^*(x - w_x^*) - x^{\text{ref}}|^2, \quad x \in \mathbb{R}^{30}, \tag{B.5}$$

with $\kappa = 50$, where R_x^*, w_x^* are the optimal rotation and the optimal translation that minimize the RMSD (see Eq. (C.2)). The function b is defined as (the negative gradient of V in full space)

$$b = -\nabla V(x) = -\kappa \left(R_x^*(x - w_x^*) - x^{\text{ref}} \right), \tag{B.6}$$

where the second equality follows by differentiating V in (B.5) and using the first order optimality equations satisfied by R_x^* and w_x^* (also see [5]).

We also build our model to make sure that the generated distribution is SE(3)-invariant (i.e. invariant under rotations and translations). For this, we rely on the theoretical results in [35] and in Appendix C. One can check that $V(x)$ is SE(3)-invariant and b satisfies property (C.1) in Appendix C, that is, b is equivariant under rotations and invariant under translations. This guarantees that the prior distribution $p(x^{(N)})$, which we choose as the invariant distribution of the forward process, is SE(3)-invariant as well. We still need to make sure that the transition densities of the reverse Markov chain are SE(3)-invariant. For this purpose, in the reverse process we set $s^{(k+1),\theta}(x) = (R_x^*)^\top f_\theta(R_x^*(x - w_x^*), \frac{(k+1)T}{N})$, where $f_\theta : \mathbb{R}^{30} \times \mathbb{R} \to \mathbb{R}^{30}$ is modeled by a single MLP with parameter θ, and both R_x^* and b_x^* are computed by the Kabsch algorithm [16]. With this choice, $s^{(k+1),\theta}(x)$ satisfies property (C.1) by Proposition C.1 in Appendix C, and the transition density of the reverse process is SE(3)-invariant by Proposition C.2 in Appendix C. Since the prior $p(x^{(N)})$ is also SE(3)-invariant, we conclude that the learned distribution $p_\theta(x^{(0)})$ is SE(3)-invariant [35]. Compared to the commonly used equivariant networks [26], our network fits our experiment better thanks to its lower computational cost and reduced memory usage.

B.5. Additional experimental results. This section presents additional experimental results, focusing on computation time and the non-convergence rate of trajectories. We first report the simulation time T_{sim} and training time T_{train}, with the percentages of the total runtime T_{total} indicated in parentheses in Table B.2. The trajectory update interval l_{f}, the number of steps N, and the cost of solving Newton's equation jointly determine the simulation time T_{sim}. Recall that the total complexity of Newton's method is $\mathcal{O}(k_{\text{iter}}(C_\xi + (n-d)^3))$, where C_ξ denotes the computational cost of evaluating $\nabla \xi$. In our experiments, $C_\xi = \mathcal{O}(1)$ for most datasets, except for meshes

where neural network evaluation is required. The codimension is typically $n-d=1$, except for SO(10), where $n-d=55$. When Newton's method converges, it requires at most $k_{\text{iter}}=3$ iterations.

Moreover, as discussed in Section 3.1, (3.1) may not yield solutions for certain vectors v, leading to the discarding of corresponding trajectories. Table B.3 provides the failure rate of trajectory generation in our experiments, as well as the maximum number of iterations required for Newton's method in cases where it converges.

Datasets	l_{f}	N_{epoch}	T_{sim}	T_{train}	T_{total}	T_{epoch}
Bunny, $k=50$	100	2000	252(1.8%)	14120	14372	7.06
Bunny, $k=100$	100	2000	142(1.6%)	8718	8860	4.36
Spot, $k=50$	100	2000	113(1.3%)	8751	8864	4.38
Spot, $k=100$	100	2000	70(1.3%)	5189	5259	2.59
SO(10)	100	2000	2426(11.2%)	19289	21715	9.64
Transformed-SO(10)	100	2000	491(2.2%)	22276	22767	22.28
Hamiltonian	1	4000	1938(79.6%)	496	2434	0.12
Alanine dipeptide	100	5000	159(1.1%)	14299	14458	2.86

Table B.2: Detailed runtime metrics in our experiments. We report T_{sim}, T_{train}, and T_{total} as the time for path generation, time for training, and total runtime, respectively, with the percentages of the total runtime T_{total} shown in parentheses. The parameter l_{f} determines the frequency of trajectory updates. The final column T_{epoch} shows the training time per epoch, calculated as $T_{\text{train}}/N_{\text{epoch}}$. All time metrics are reported in seconds.

Datasets	$\sigma_{\max}$	$R_{\text{fail\_fwd}}$	$R_{\text{fail\_bwd}}$	$\text{iter}_{\max}$	tol
Bunny, $k=50$	0.007	1.00%	0.82%	3	1e-4
Bunny, $k=100$	0.007	0.65%	0.55%	3	1e-4
Spot, $k=50$	0.010	0.15%	0.25%	3	1e-4
Spot, $k=100$	0.010	0.11%	0.10%	3	1e-4
SO(10)	0.089	0.00%	0.00%	3	1e-6
Transformed-SO(10)	0.13	0.00%	0.00%	3	5e-6
Hamiltonian	0.15	0.00%	0.00%	4	1e-5
Alanine dipeptide	0.022	0.01%	0.00%	2	1e-5

Table B.3: Failure rate of trajectory generation. $R_{\text{fail\_fwd}}$ and $R_{\text{fail\_bwd}}$ represent the percentages of discarded trajectories when sampling the forward and reverse process, respectively. $\sigma_{\max}$ denotes the maximum value of $(\sigma_k)_{0\le k\le N-1}$. $\text{iter}_{\max}$ denotes the maximum number of iterations for Newton's method to converge (i.e. until the error is less than tol).

Appendix C. Theoretical results on neural networks for molecular systems.

In this section, we present theoretical results for the neural network architecture we employed in studying alanine dipeptide.

Assume that the system consists of M atoms, where the coordinates of thee i-th atom are denoted by $\boldsymbol{x}_i \in \mathbb{R}^3$, for $i = 1, 2, \ldots, M$. Let $x \in \mathbb{R}^{3M}$ be the vector consisting of all the coordinates $\boldsymbol{x}_1 \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M \in \mathbb{R}^3$. For simplicity, given a rotation matrix $R \in \mathrm{SO}(3)$ and a translation vector $w \in \mathbb{R}^3$, we use the conventional notation $Rx + w$ to denote the vector in $\mathbb{R}^{3M}$ that consists of the transformed coordinates $R\boldsymbol{x}_1 + w, R\boldsymbol{x}_2 + w, \ldots, R\boldsymbol{x}_M + w \in \mathbb{R}^3$. We say that a function f defined in $\mathbb{R}^{3M}$ is SE(3)-invariant, if $f(Rx + w) = f(x)$, for all $R \in \mathrm{SO}(3)$, $w \in \mathbb{R}^3$, and for all $x \in \mathbb{R}^{3M}$. We say that function $f : \mathbb{R}^{3M} \to \mathbb{R}^{3M}$ possesses property (C.1), if it is both equivariant under rotations and invariant under translations, i.e.

$$f(Rx + w) = Rf(x), \text{ for all } R \in \mathrm{SO}(3), w \in \mathbb{R}^3, \text{ and all } x \in \mathbb{R}^{3M}. \tag{C.1}$$

Assume that a configuration x^{ref} is chosen as reference. Given x, the optimal rotation matrix and the optimal translation vector, which minimize the RMSD

$$\mathrm{RMSD}(x; x^{\mathrm{ref}}) = \left(\frac{1}{M} |R(x - w) - x^{\mathrm{ref}}|^2 \right)^{\frac{1}{2}} \tag{C.2}$$

from the reference x^{ref}, are denoted by R_x^* and w_x^*, respectively.

Proposition C.1 characterizes functions that are both equivariant under rotations and invariant under translations. Proposition C.2 guarantees the SE(3)-invariance of the transition densities of our diffusion model.

PROPOSITION C.1. *The following two claims are equivalent.*

- *Function $s : \mathbb{R}^{3M} \to \mathbb{R}^{3M}$ possesses property (C.1).*
- *There is a function $f : \mathbb{R}^{3M} \to \mathbb{R}^{3M}$, such that $s(x) = (R_x^*)^\top f(R_x^*(x - w_x^*))$, for all $x \in \mathbb{R}^{3M}$.*

Proof. It is straightforward to verify that the first claim implies the second claim. In fact, setting $R = R_x^*$, $w = -R_x^* w_x^*$, and using the identity $R^\top R = I_3$, we obtain from the first claim that $s(x) = (R_x^*)^\top s(R_x^*(x - w_x^*))$. Hence, the second claim holds with $f = s$. To show that the second claim also implies the first one, we use the fact that the optimal rotation R_{Rx+w}^* and the optimal translation w_{Rx+w}^*, which minimize the RMSD of the state $Rx + w$ from the reference x^{ref}, are given by $R_{Rx+w}^* = R_x^* R^\top$ and $w_{Rx+w}^* = Rw_x^* + w$, respectively. This fact can be directly checked using (C.2). In particular, we have

$$R_{Rx+w}^*(Rx + w - w_{Rx+w}^*) = R_x^*(x - w_x^*).$$

Therefore, for the function s defined in the second claim, we can compute, for any $R \in \mathrm{SO}(3)$, $w \in \mathbb{R}^3$, and any $x \in \mathbb{R}^{3M}$,

$$\begin{aligned} s(Rx + w) &= (R_{Rx+w}^*)^\top f\big(R_{Rx+w}^*(Rx + w - w_{Rx+w}^*)\big) \\ &= R(R_x^*)^\top f(R_x^*(x - w_x^*)) \\ &= Rs(x), \end{aligned}$$

which shows the first claim. ∎

PROPOSITION C.2. *Assume that ξ is* SE(3)*-invariant and b possesses property (C.1). Then, the transition density (3.6) of the forward process is* SE(3)*-invariant. Further assume that the function $s^{(k+1),\theta}$ possesses property (C.1) for $0 \le k \le N - 1$. Then, the transition density (3.8) of the reverse process is also* SE(3)*-invariant.*

N	σ	$R_{\text{fail\_fwd}}$	T_{sim}	T_{train}	JS	NLL
50	2.45e-2	2.86%	19.2(3.2%)	589.3	4.382e-01$\pm$3.81e-03	0.77$\pm$0.01
100	1.73e-2	1.23%	36.9(2.7%)	1313.1	4.314e-01$\pm$2.18e-03	0.77$\pm$0.01
200	1.22e-2	0.34%	54.0(2.0%)	2673.0	4.312e-01$\pm$2.66e-03	0.77$\pm$0.01
300	1.00e-2	0.11%	66.4(1.6%)	4186.2	4.298e-01$\pm$2.89e-03	0.77$\pm$0.01
400	0.87e-2	0.05%	84.8(1.2%)	7297.1	4.300e-01$\pm$1.68e-03	0.78$\pm$0.01

Table C.1: Ablation study of the number of steps N on the Spot the Cow dataset with $k=100$. Here, σ denotes the step size and $R_{\text{fail\_fwd}}$ represents the proportion of failed forward trajectories. T_{sim} and T_{train} are the simulation and training times (see Table B.2). JS denotes the Jensen-Shannon distance on mesh face histograms between the ground truth and generated distributions.

N	σ	$R_{\text{fail\_fwd}}$	T_{sim}	T_{train}	$\mathcal{W}_1$
50	0.26	0%	823.2(81.1%)	191.5	3.152e-01$\pm$1.96e-03
100	0.18	0%	1429.0(85.0%)	253.0	3.117e-01$\pm$8.09e-04
150	0.15	0%	2035.8(84.3%)	378.4	3.098e-01$\pm$3.48e-03
200	0.13	0%	2717.9(84.9%)	482.9	3.093e-01$\pm$2.66e-03

Table C.2: Ablation study of the number of steps N on conserved Hamiltonian surfaces. $\mathcal{W}_1$ denotes the 1-Wasserstein distance in coordinate space $(q_1, q_2, \ldots, q_n)$ between the ground truth and generated distributions.

Proof. We consider the transition density (3.6). Recall that $U_x \in \mathbb{R}^{n\times d}$ is a matrix whose columns form an orthonormal basis of $T_x\mathcal{M}$. Since ξ is SE(3)-invariant, we have $\xi(Rx+w)=\xi(x)$, for all rotations R and translation vectors w, which implies that $Rx+w\in\mathcal{M}$, if and only if $x\in\mathcal{M}$. Differentiating the identity $\xi(Rx+w)=\xi(x)$ with respect to x, we obtain the relation $\nabla\xi(Rx+w)=R\nabla\xi(x)$, from which we see that U_{Rx+w} can be chosen such that $U_{Rx+w}=RU_x$. For the orthogonal projection matrix P in (A.1), using the identity $R^\top R=I_3$, we can compute

$$\begin{aligned}
P(Rx+w) &= I_n - \nabla\xi(Rx+w)\big(\nabla\xi(Rx+w)^\top\nabla\xi(Rx+w)\big)^{-1}\nabla\xi(Rx+w)^\top \\
&= I_n - R\nabla\xi(x)\big(\nabla\xi(x)^\top\nabla\xi(x)\big)^{-1}\nabla\xi(x)^\top R^\top \\
&= RP(x)R^\top .
\end{aligned}$$

Moreover, since both b and $\nabla\xi$ satisfy the property (C.1), we also have $\epsilon^{(\sigma_k)}_{Rx^{(k)}+w}=\epsilon^{(\sigma_k)}_{x^{(k)}}$ (i.e. the probabilities of having no solution are the same). Therefore, for the transition density (3.6), we can derive, for any $R\in\mathrm{SO}(3)$ and $w\in\mathbb{R}^3$,

$$\begin{aligned}
&q(Rx^{(k+1)}+w\,|\,Rx^{(k)}+w) \\
=&(2\pi\sigma_k^2)^{-\frac{d}{2}}\big(1-\epsilon^{(\sigma_k)}_{Rx^{(k)}+w}\big)^{-1}\big|\det(U^\top_{Rx^{(k)}+w}U_{Rx^{(k+1)}+w})\big| \\
&\quad\cdot\exp\Big(-\frac{\big|P(Rx^{(k)}+b)\big(Rx^{(k+1)}-Rx^{(k)}-\sigma_k^2 b(Rx^{(k)}+w)\big)\big|^2}{2\sigma_k^2}\Big) \\
=&(2\pi\sigma_k^2)^{-\frac{d}{2}}\big(1-\epsilon^{(\sigma_k)}_{x^{(k)}}\big)^{-1}\big|\det(U^\top_{x^{(k)}}R^\top RU_{x^{(k+1)}})\big|
\end{aligned}$$

$$\cdot \exp\Big(-\frac{\big|RP(x^{(k)})R^\top\big(Rx^{(k+1)}-Rx^{(k)}-\sigma_k^2 Rb(x^{(k)})\big)\big|^2}{2\sigma_k^2}\Big)$$

$$=(2\pi\sigma_k^2)^{-\frac{d}{2}}\big(1-\epsilon^{(\sigma_k)}_{x^{(k)}}\big)^{-1}\big|\det(U^\top_{x^{(k)}}U_{x^{(k+1)}})\big|\exp\Big(-\frac{\big|P(x^{(k)})\big(x^{(k+1)}-x^{(k)}-\sigma_k^2 b(x^{(k)})\big)\big|^2}{2\sigma_k^2}\Big)$$

$$=q(x^{(k+1)}\,|\,x^{(k)}),$$

which shows the SE(3)-invariance of the transition density of the forward process. The invariance of the transition density of the reverse process in (3.8) can be proved using the same argument, assuming that $s^{(k+1),\theta}$ satisfies the relation $s^{(k+1),\theta}(Rx+w)=Rs^{(k+1),\theta}(x)$. ∎

Appendix D. Ablation study.

Table C.1 presents the computational cost and performance analysis for the Spot the Cow dataset with $k=100$ under varying N. We observe that, as N increases, the Newton method's failure rate decreases, and both trajectory simulation time and training time increase accordingly. To better assess the generation quality, we compare the Jensen-Shannon distance between the ground truth and generated distributions, as the NLL metric fails to reveal meaningful differences. The results indicate that $N=50$ yields relatively poor performance, while other values of N show comparable results with marginal differences.

Table C.2 presents the ablation study on conserved Hamiltonian surfaces. The results demonstrate that as N increases, the computational cost grows while the accuracy improves accordingly.

The performance of our method is not highly sensitive to the number of steps N, as long as N is not too small; the computational cost scales positively correlated with N. Thus, a viable strategy is to choose N by balancing the Newton method's failure rate against computational overhead, while maintaining satisfactory generation quality.

REFERENCES

[1] A. Banyaga, D. Hurtubise, and D. Ajayi, *Lectures on Morse homology*, volume 29, Springer, 2004. 2.1

[2] H. Ben-Hamu, S. Cohen, J. Bose, B. Amos, M. Nickel, A. Grover, R. T. Q. Chen, and Y. Lipman, *Matching normalizing flows and probability paths on manifolds*, ICML, 162:1749–1763, 2022. 1

[3] R. T. Q. Chen and Y. Lipman, *Flow matching on general geometries*, ICLR, 27070–27093, 2024. 1, 5.1

[4] G. Ciccotti, T. Lelièvre, and E. Vanden-Eijnden, *Projection of diffusions on submanifolds: Application to mean force computation*, Comm. Pure Appl. Math., 61(3):371–408, 2008. 3.1, 4, A

[5] E. A. Coutsias, C. Seok, and K. A. Dill, *Using quaternions to calculate RMSD*, J. Comput. Chem., 25(15):1849–1857, 2004. B.4

[6] K. Crane, U. Pinkall, and P. Schröder, *Robust fairing via conformal curvature flow*, ACM Trans. Graph., 32(4):1–10, 2013. 5.1

[7] V. De Bortoli, E. Mathieu, M. Hutchinson, J. Thornton, Y. W. Teh, and A. Doucet, *Riemannian score-based generative modelling*, NeurIPS, 35:2406–2422, 2022. 1, 4, A, B.1

[8] G. Fiorin, M. L. Klein, and J. Hénin, *Using collective variables to drive molecular dynamics simulations*, Mol. Phys., 111(22-23):3345–3362, 2013. B.4

[9] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, *Implicit geometric regularization for learning shapes*, ICML, 119:3569–3579, 2020. 5.1

[10] J. Ho, A. Jain, and P. Abbeel, *Denoising diffusion probabilistic models*, NeurIPS, 33:6840–6851, 2020. 1, 2.2

[11] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Surface reconstruction from unorganized points*, Proc. ACM Comput. Graph. Interact. Tech., 71–78, 1992. 1

[12] E. P. Hsu, *Stochastic analysis on manifolds*, Number 38 in Grad. Stud. Math., American Mathematical Society, 2002. A

[13] C.-W. Huang, M. Aghajohari, J. Bose, P. Panangaden, and A. Courville, *Riemannian diffusion models*, NeurIPS, 35:2750–2761, 2022. 1

[14] A. Hyvärinen, *Estimation of non-normalized statistical models by score matching*, J. Mach. Learn. Res., 6(24):695–709, 2005. 1

[15] J. Jo and S. J. Hwang, *Generative modeling on manifolds through mixture of Riemannian diffusion processes*, ICML, 235:22348–22370, 2024. 5.1

[16] W. Kabsch, *A solution for the best rotation to relate two sets of vectors*, Acta Crystallogr., Sect. A, 32(5):922–923, 1976. B.4

[17] A. Klimovskaia, D. Lopez-Paz, L. Bottou, and M. Nickel, *Poincaré maps for analyzing complex hierarchies in single-cell data*, Nat. Commun., 11(1):2966, 2020. 1

[18] T. Lelièvre, T. Pigeon, G. Stoltz, and W. Zhang, *Analyzing multimodal probability measures with autoencoders*, J. Phys. Chem. B, 128(11):2607–2631, 2024. B.4

[19] T. Lelièvre, M. Rousset, and G. Stoltz, *Hybrid Monte Carlo methods for sampling probability measures on submanifolds*, Numer. Math., 143(2):379–421, 2019. 3.1

[20] T. Lelièvre, G. Stoltz, and W. Zhang, *Multiple projection MCMC algorithms on submanifolds*, IMA J. Numer. Anal., 43(2):737–788, 2022. 3.1

[21] A. Lou, D. Lim, I. Katsman, L. Huang, Q. Jiang, S. N. Lim, and C. M. De Sa, *Neural manifold ordinary differential equations*, NeurIPS, 33:17548–17558, 2020. 1

[22] A. Lou, M. Xu, A. Farris, and S. Ermon, *Scaling Riemannian diffusion models*, NeurIPS, 36:80291–80305, 2023. 1

[23] E. Mathieu and M. Nickel, *Riemannian continuous normalizing flows*, NeurIPS, 33:2503–2515, 2020. 1

[24] B. T. Polyak and A. B. Juditsky, *Acceleration of stochastic approximation by averaging*, SIAM J. Control Optim., 30(4):838–855, 1992. B.1

[25] N. Rozen, A. Grover, M. Nickel, and Y. Lipman, *Moser flow: Divergence-based generative modeling on manifolds*, NeurIPS, 34:17669–17680, 2021. 1, 5.1

[26] V. G. Satorras, E. Hoogeboom, and M. Welling, *E(n) equivariant graph neural networks*, ICML, 139:9323–9332, 2021. B.4

[27] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann, *Neural descriptor fields: SE (3)-equivariant object representations for manipulation*, Proc. IEEE Int. Conf. Robot. Autom., 6394–6400, 2022. 1

[28] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, *Deep unsupervised learning using nonequilibrium thermodynamics*, ICML, 37:2256–2265, 2015. 2.2

[29] Y. Song, C. Durkan, I. Murray, and S. Ermon, *Maximum likelihood training of score-based diffusion models*, NeurIPS, 34:1415–1428, 2021. 2.2

[30] Y. Song and S. Ermon, *Generative modeling by estimating gradients of the data distribution*, NeurIPS, 32:11895–11907, 2019. 1

[31] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, *Score-based generative modeling through stochastic differential equations*, ICLR, 240–275, 2021. 1

[32] G. Turk and M. Levoy, *Zippered polygon meshes from range images*, Proc. ACM Comput. Graph. Interact. Tech., 311–318, 1994. 5.1

[33] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. Berendsen, *GROMACS: Fast, flexible, and free*, J. Comput. Chem., 26(16):1701–1718, 2005. B.4

[34] J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, et al., *De novo design of protein structure and function with RFdiffusion*, Nature, 620(7976):1089–1100, 2023. 1

[35] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang, *GeoDiff: A geometric diffusion model for molecular conformation generation*, ICLR, 14163–14181, 2022. 5.4, B.4

[36] E. Zappa, M. Holmes-Cerfon, and J. Goodman, *Monte Carlo on manifolds: Sampling densities and integrating functions*, Comm. Pure Appl. Math., 71(12):2609–2647, 2018. 3.1

[37] W. Zhang, *Ergodic SDEs on submanifolds and related numerical sampling schemes*, ESAIM Math. Model. Numer. Anal., 54(2):391–430, 2020. A

[38] W. Zhang, H. Wang, C. Hartmann, M. Weber, and C. Schütte, *Applications of the cross-entropy method to importance sampling and optimal control of diffusions*, SIAM J. Sci. Comput., 36(6):A2654–A2672, 2014. 2.2

[39] Y. Zhu, T. Chen, L. Kong, E. Theodorou, and M. Tao, *Trivialized momentum facilitates diffusion generative modeling on Lie Groups*, ICLR, 30049–30078, 2025. 1