The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Constrained Langevin Algorithms for Score-Based Diffusion

by

Tadgh Kelly

Dissertation Presented for the Degree of
MSc in Computational Applied Mathematics

August 2023

Supervised by
Prof. Benedict Leimkuhler

Abstract

Score-based models have been seen to perform remarkably well at generating data in a range of domains. Score-based generative modelling (SBGM) consists of a noising stochastic differential equation (SDE) which is used to transform the datapoints to a known distribution, and a denoising SDE which transforms back from the known distribution to the data distribution. Constrained generative modelling has applications in many fields where the data is supported on a manifold rather than on Euclidean space. Most work on constrained score-based diffusion (SBD) has focused on Riemannian manifolds using the Riemannian metric, and has used overdamped Langevin dynamics to noise and denoise the data. Here we examine the use of g-BAOAB, an integrator for constrained underdamped Langevin dynamics, and in doing so we focus on general manifolds defined by holonomic constraints, by considering the constraint manifold as an embedded manifold in Euclidean space. The approach is compared to a continuous normalizing "Moser flow" model, and is demonstrated on a problem of human pose and generation.

Acknowledgments

I would like to thank my project supervisor, Professor Benedict Leimkuhler, and also Peter Whalley and Katerina Karoni

Own Work Declaration

I declare that the following work is my own except where otherwise noted.

Contents

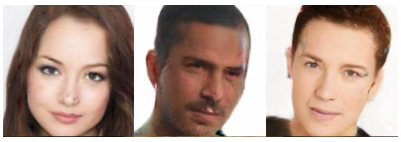
List of Tables

List of Figures

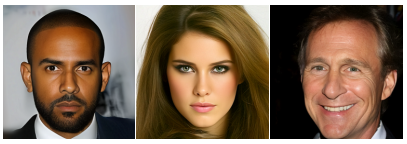
1 Introduction

Generative modelling aims to train a generative model to sample from the same distribution as the data. There are many applications of generative modelling, including image syntheses and inpainting [5, 33] and speech and music synthesis [24], but there are also applications in many other fields, since sampling from an unknown distribution is a common problem.

Early generative modelling techniques could be grouped into two categories; likelihood-based models and implicit generative models. Likelihood-based models, as demonstrated in Figure 1b such as Variational Auto-Encoders (VAEs) learn the probability density function using maximum likelihood [9]. Likelihood-based models require a restricted model to ensure a tractable normalizing constant, which commonly means that VAEs are based on fitting Gaussian-based models. Implicit generative models generate data without explicitly learning the density function, and are usually in the form of General Adversarial Networks (GANs), in which one neural network generates the data and another called the "discriminator" aims to discriminate between true and artificial data. However training for GANs is famously unstable [35].



(a) Images generated by GAN (BEGAN) [5].



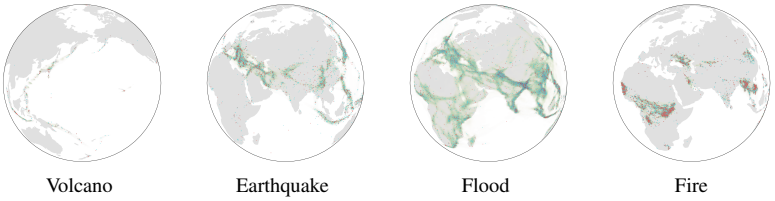
(b) Facial images generated by VAE (NVAE) [33].

Figure 1: Facial images generated by implicit and explicit generative models.

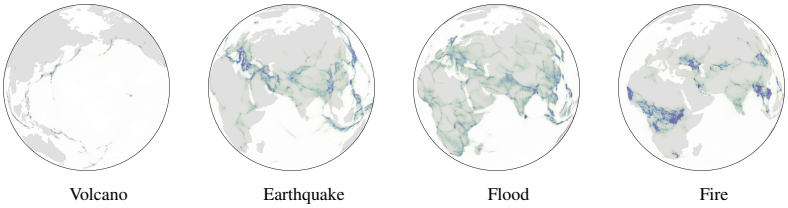
In recent years, diffusion models have been shown to be very effective, and have become an active area of research, outperforming GANs at a number of tasks. Diffusion models are a family of generative model which perturb data by injecting noise, and then learn to reverse the noising process in order to generate samples from the original data distribution [35]. SBD is based on learning the score function of the noising process, the gradient of the log of the marginal probability density function, which eliminates the need for the normalizing constant of the density function. Another important fact about score-based models is that they allow a model for the score to be trained without explicit knowledge of the true score function, by minimizing the Fisher divergence, which can be done by sampling from the data distribution [11, 30]. Thus the only information we need to train a score-based model is available from the data samples. While score matching can be performed without adding noise to the data, the motivation for the noising process comes from the difficulty in training a score-based model in areas of low density, where there may be very few data points. Adding noise aids the training process in these regions by preventing overfitting and counteracting biases that may be introduced in these regions [31].

Figure 2: Facial images generated by a diffusion model [10].

Constrained data arises in many areas, including protein modelling and image recognition as well as many other fields [27, 21]. Some that have been examined in the context of diffusion models are geoscience, in which there are many events which occur on the surface of the earth, as shown in Figure 3, and human pose generation, in which the length of the bones are constrained to be constant [9]. In settings like these, the data often lie on Riemannian manifolds rather than in Euclidean space. This motivates the use of constrained diffusion algorithms, in order to ensure that the samples lie on the manifold in question, which wouldn't be the case with standard diffusion models.



(a) Results of a Moser flow model [25].



(b) Results of a score-based model [8].

Figure 3: Moser flow and score-based generative models trained on earth sciences data gathered by Mathieu and Nickel [8]. Blue and red dots represent training and test datapoints, respectively, while the learned density is green-blue, with blue indicating larger values.

There are two main methods of generative modelling which have been used for modelling on a constraint manifold; SBD and Continuous Normalizing Flows (CNFs). Recent research into SBD on Riemannian manifolds has shown promising results, with results from Euclidean models often having direct analogs on Riemannian manifold [8]. In terms of CNFs on manifolds there are generally two different ways of generating a normalizing flow on the manifold. For manifolds which are topologically equivalent to Euclidean space, there are methods which involve the pushforward of a Euclidean flow along an invertible map $f : \mathbb{R}^n \to \mathcal{M}$, normally taken to be the exponential map, however, such a map is not necessarily available unless $\mathcal{M}$ is topologically equivalent to Euclidean space. Alternatively, Riemannian geometry, specifically the Riemannian divergence, can be used to define flows on the manifold directly [22]. This method is more general in that it can be used to construct CNFs on manifolds which are not topologically equivalent to Euclidean space. A third option for embedded submanifolds of Euclidean space is to focus on the embedding from Euclidean space to define a CNF on the manifold, as touched upon by Rozen et al [25]. This method will allow us to work with holonomic constraints of the same form as those used in constrained Langevin dynamics, which will allow a comparison between the two methods. While overdamped Langevin dynamics have often been used in SBD, underdamped Langevin dynamics has been used less frequently, and in particular in the case of constrained SBD, has not been studied. Using underdamped Langevin dynamics may bring certain advantages, since it has been observed to converge more quickly to the invariant distribution [7].

1.1 Outline of thesis

This report is structured as follows. In Section 2 we introduce some background concepts related to constrained Hamiltonian systems. In Section 3, we introduce Langevin dynamics with

constraints and the g-BAOAB integrator, and study its properties on a simple distribution. In Section 4 we introduce SBD on Euclidean space and generalized to manifolds, and the time-reversal of the constrained underdamped Langevin dynamics. In Section 5 we relate the SBD model to CNFs, and discusses the limitations of underdamped Langevin dynamics in this area. We also introduce Moser flows, a type of CNF which is particularly well suited to certain Riemannian manifolds, and compare it with the SBD method. In Section 6, underdamped Langevin dynamics and g-BAOAB are used to generate samples from a distribution of human poses. Section 7 summarizes some alternative methods for generative modelling, and how they relate to the methods discussed in this paper.

2 Background

2.1 Hamiltonian Systems

A Hamiltonian system is a system of differential equations characterized by its Hamiltonian function $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q} = \nabla_{\mathbf{p}} H(\mathbf{q}, \mathbf{p}), \tag{2.1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = -\nabla_{\mathbf{q}} H(\mathbf{q}). \tag{2.2}$$

Then when the Hamiltonian is the total kinetic plus potential energy, $H(\mathbf{q}, \mathbf{p}) = \frac{\mathbf{p}^T \mathbf{M}^{-1} \mathbf{p}}{2} + U(\mathbf{q}, \mathbf{p})$, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q} = \mathbf{M}^{-1}\mathbf{p}, \tag{2.3}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = -\nabla_{\mathbf{q}} U(\mathbf{p}, \mathbf{q}). \tag{2.4}$$

In this paper we will omit the mass matrix by setting $\mathbf{M} = \mathbf{I} \in \mathbb{R}^{n \times n}$, since this is often done in data science settings when we are no longer dealing with actual physical systems.

2.1.1 Symplectic Integrators

The evolution of a Hamiltonian system is symplectic, which means that it has an area form $\omega = dp \wedge dq$ in the phases space, that measures areas of phase space,

$$A = \int_S \omega, \tag{2.5}$$

for a given region S. Symplectic integrators are designed to maintain the symplectic structure of the phase space, which gives them the desirable property of having bounded energy fluctuations. A numerical method is called symplectic if the area form is preserved exactly:

$$d\mathbf{q}_{n+1} \wedge d\mathbf{p}_{n+1} = d\mathbf{q}_n \wedge d\mathbf{p}_n. \tag{2.6}$$

While we can also construct integrators designed specifically to conserve energy, their performance is often poorer than a simpler symplectic method [16]. As shown in Figure 4 by comparing the symplectic Euler method to the forward Euler method for the simple harmonic oscillator, the energy fluctuates for both methods but the fluctuations remain bounded by the symplectic method.

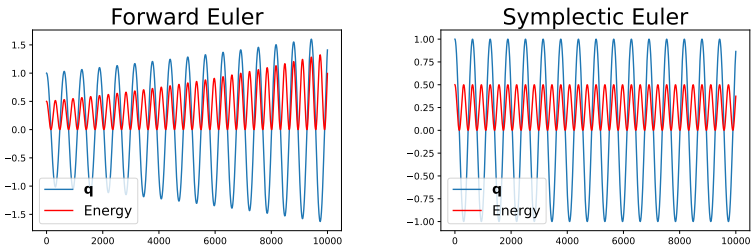


Figure 4: Forward and symplectic Euler integrator for the simple pharmonic oscillator with mass $m = 1$.

2.1.2 Splitting methods

Splitting methods provide a way to create integrators for Hamiltonian systems and in particular, a way to create symplectic integrators. Suppose we can split the Hamiltonian H into a sum of Hamiltonians,

$$H(\mathbf{q},\mathbf{p}) = \sum_{i=1}^{k} H_i(\mathbf{q},\mathbf{p}), \tag{2.7}$$

with each of the Hamiltonians H_i giving an explicitly solvable system. Since explicitly solving the equations gives a symplectic flow map, and compositions of symplectic maps are themselves symplectic maps, this gives a symplectic integrator. In Hamiltonian systems, this often involves splitting the Hamiltonian into kinetic and potential energy. First we introduce the flow map. Consider an initial value problem:

$$\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}),\ \mathbf{q}(0) = \mathbf{q}_0. \tag{2.8}$$

Then the *flow map* map $\mathcal{F}_t$ maps points in the phase space according to the time-evolution of the problem:

$$\mathcal{F}_t(\mathbf{q}_0) = \mathbf{q}_t. \tag{2.9}$$

If $H(\mathbf{p},\mathbf{q}) = H_1(\mathbf{p},\mathbf{q}) + H_2(\mathbf{p},\mathbf{q})$, and H_1 and H_2 have flow maps $\mathcal{F}^1$ and $\mathcal{F}^2$, then it can be shown that the map given by their composition,

$$\mathcal{F}^{1+2} = \mathcal{F}^1 \circ \mathcal{F}^2, \tag{2.10}$$

is a second order approximation of the true flow $\mathcal{F}$ [14].

2.2 Constrained Dynamics

We will begin by introducing mechanical systems with holonomic constraints. For a constrained system, a particle is subject to two types of force: applied forces which are defined using the potential energy function U, and constraint forces which are the forces which force the particle to remain on the constraint surface [16]. D'Alembert's principle states that the constraining force acts along the normal direction to the constraint surface, therefore if $\mathbf{F}_g$ is the constraint force we can write $\mathbf{F}_g = \lambda\nabla_{\mathbf{q}}$ for some scalar λ. λ is unknown, and it can be solved for by requiring that the particle remains on the constraint surface.

Consider algebraic constraints of the form

$$g_i(\mathbf{q}) = 0,\ i = 1,\dots,m, \tag{2.11}$$

where $g_i : \mathbb{R}^n \to \mathbb{R}$ is smooth for all $i \in [1,\dots m]$. This mechanical system is holonomic and constant in time. If the Hamiltonian is again given by $H(\mathbf{q},\mathbf{p}) = \frac{\mathbf{p}^T\mathbf{p}}{2} + U(\mathbf{q},\mathbf{p})$, then we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q} = \mathbf{p} \tag{2.12a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = \mathbf{F} - \sum_{i=1}^{m} \lambda_i g_i(q) \tag{2.12b}$$

$$g_i(\mathbf{q}) = 0,\ i = 1,\dots,m, \tag{2.12c}$$

where we have omitted the mass matrix and let $\mathbf{F} = -\nabla_{\mathbf{q}} U$. We can rewrite this system as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q} = \mathbf{p}, \tag{2.13a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = \mathbf{F}(\mathbf{q}) - \mathbf{G}^T(\mathbf{q})\mathbf{\Lambda}, \tag{2.13b}$$

where $\mathbf{g}(\mathbf{q})$ is the vector of constraints $\mathbf{g}(\mathbf{q}) = \begin{bmatrix} g_1(\mathbf{q}) & g_2(\mathbf{q}) & \dots & g_m(\mathbf{q}) \end{bmatrix}^T$, $\mathbf{\Lambda} = \begin{bmatrix} \lambda_1, \dots \lambda_m \end{bmatrix}^T$ and $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^{m \times n}$ represents the Jacobian matrix of the constraint vector evaluated at $\mathbf{q}$.

We will assume that $\{\nabla_{\mathbf{q}} g_i(\mathbf{q})\}_{i=1,\dots,m}$ forms a linearly independent set for each $\mathbf{q} \in \mathbb{R}^n$, or equivalently that $\mathbf{G}(\mathbf{q})$ has full rank for each $\mathbf{q} \in \mathbb{R}^n$. This lets us consider the constraint surface as a smooth algebraic manifold $\mathcal{M}$, which implies that it can be equipped with a Riemannian metric. While there are many results which make use of Riemannian geometry, for a general set of holonomic constraints the Riemannian metric will not be known, so we will avoid the use of Riemannian metrics and local coordinates, and instead focus on the manifold as a submanifold of Euclidean space.

2.2.1 Example: The Simple Pendulum

Consider the simple pendulum shown below, where the bob has mass 1 and the length of the rod is 1. We let $g_1(\mathbf{q})$ be the constraint so that $g_1(\mathbf{q}) = ||\mathbf{q}||_2^2$. The tension then acts along the normal of the constraint, so we can write $T = \lambda \nabla_{\mathbf{q}} g_1$, which gives the following equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q} = \mathbf{p} \tag{2.14a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = \mathbf{F} - \lambda \nabla_{\mathbf{q}} g_1 \tag{2.14b}$$

$$g_1(\mathbf{q}) = 0. \tag{2.14c}$$

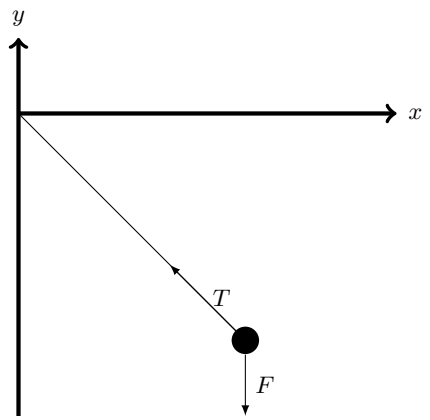


Figure 5: The simple pendulum.

Using the fact that $\frac{\mathrm{d}}{\mathrm{d}t} g_1 = 0$ gives:

$$\frac{\mathrm{d}}{\mathrm{d}t} g_1 = \frac{\mathrm{d}}{\mathrm{d}\mathbf{q}} g_1 \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q} + \frac{\mathrm{d}}{\mathrm{d}\mathbf{p}} g_1 \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = 0 \tag{2.15}$$

$$\implies \ddot{g_1} = 2\dot{\mathbf{q}}^T\dot{\mathbf{q}} + 2\mathbf{q}^T\ddot{\mathbf{q}} = 0 \tag{2.16}$$

$$\implies \mathbf{q}^T(\mathbf{F} - 2\lambda\mathbf{q}) + \dot{\mathbf{q}}^T\dot{\mathbf{q}} = 0 \tag{2.17}$$

$$\implies 2\lambda\mathbf{q}^T\mathbf{q} = \left(\mathbf{q}^T\mathbf{F} + \dot{\mathbf{q}}^T\dot{\mathbf{q}}\right) \tag{2.18}$$

$$\implies \lambda = \frac{\mathbf{q}^T\mathbf{F} + \dot{\mathbf{q}}^T\dot{\mathbf{q}}}{2\mathbf{q}^T\mathbf{q}}. \tag{2.19}$$

In this case this gives

$$\lambda = \frac{1}{2}\left(\frac{\begin{bmatrix} x & y \end{bmatrix}^T \mathbf{F} + \dot{x}^2 + \dot{y}^2}{x^2 + y^2}\right),$$

where we have replaced the vectors $\mathbf{q}$ and $\mathbf{p}$ with their x and y components. In the general case, we have a function for $\mathbf{\Lambda}$, given by:

$$\Lambda(\mathbf{q}, \mathbf{p}) = -(\mathbf{G}\mathbf{G}^T)^{-1}[\mathbf{G}\nabla_q U(\mathbf{q}) - g_{\mathbf{qq}} < \mathbf{p}, \mathbf{p} >], \tag{2.20}$$

where $\mathbf{G}$ is used as shorthand for $\mathbf{G}(\mathbf{q})$, $\mathbf{G}\mathbf{G}^T$ is non-singular since $\mathbf{G}$ has full rank and $g_{\mathbf{qq}} < \mathbf{p}, \mathbf{p} >$ represents a vector with m elements, with the ith element given by

$$(g_i)_{\mathbf{qq}} < \mathbf{p}, \mathbf{p} >= \sum_{k=1}^{n}\sum_{j=1}^{n} \frac{\mathrm{d}^2 g_i}{\mathrm{d}\mathbf{q}_k \mathrm{d}\mathbf{q}_j}\mathbf{p}_k\mathbf{p}_j. \tag{2.21}$$

However, applying a numerical integration scheme based on a direct discretisation of Equations 2.13 and 2.20 leads to issues, in that the solutions won't satisfy the constraint exactly, and errors could propagate forward leading to solutions which lie far from the constraint surface. This motivates another class of integrators designed to deal with this problem.

2.2.2 Constraint-preserving integrators

This section introduces a number of examples of constraint-preserving integrators, which aim to explicitly and exactly preserve the constraints at each step. Constraint-preserving integrators generally use iterative methods to find solutions that satisfy the constraints. Certain numerical integration methods can be modified so that solutions exactly preserve the constraints.

2.3 RATTLE and SHAKE

Two popular methods for constrained integration of ODEs are RATTLE and SHAKE. Although SHAKE is not technically symplectic since the velocities aren't tangent to the constraint manifold [18], it was later improved by RATTLE, which is symplectic. SHAKE was proposed by Ryckaert et al., and based on a standard leapfrog discretisation of the ODE [26].

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t \mathbf{p}^{n+1/2}, \tag{2.22a}$$
$$\mathbf{p}^{n+1/2} = \mathbf{p}^{n-1/2} - \Delta t \nabla_{\mathbf{q}} U(\mathbf{q}^n) - \Delta t \mathbf{G}(\mathbf{q}^n)^T \lambda^n, \tag{2.22b}$$
$$g_i(\mathbf{q}^{n+1}) = \mathbf{0},\ i = 1, \ldots, m, \tag{2.22c}$$
$$\mathbf{p}^n = \frac{1}{2}(\mathbf{p}^{n+1/2} + \mathbf{p}^{n-1/2}), \tag{2.22d}$$

where λ is chosen so that the solutions satisfy the cotangency constraints.

RATTLE was proposed later by Andersen to correct SHAKE by also enforcing the tangency constraint, the requirement that the velocity should be perpendicular to the normal of the constraint manifold [3].

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t \mathbf{p}^{n+1/2}, \tag{2.23a}$$
$$\mathbf{p}^{n+1/2} = \mathbf{p}^n - \frac{\Delta t}{2}\nabla_{\mathbf{q}} U(\mathbf{q}^n) - \frac{\Delta t}{2}\mathbf{G}(\mathbf{q}^n)^T \lambda_r^n, \tag{2.23b}$$
$$g_i(\mathbf{q}^{n+1}) = 0,\ i = 1, \ldots, m, \tag{2.23c}$$
$$\mathbf{p}^{n+1} = \mathbf{p}^{n+1/2} - \frac{\Delta t}{2}\nabla_q U(\mathbf{q}^{n+1}) - \frac{\Delta t}{2}\mathbf{G}(\mathbf{q}^{n+1})^T \lambda_v^{n+1}, \tag{2.23d}$$
$$0 = \mathbf{G}(\mathbf{q}^{n+1})\mathbf{p}^{n+1}, \tag{2.23e}$$

where λ_r and λ_v are chosen so that the solutions satisfy the position and tangency constraints,

respectively. However it has been shown that the two methods are algebraically equivalent, and the two methods produce identical positions with only the velocity approximations differing, so it has been suggested that an efficient process would use SHAKE steps wherever the velocity was not output and one RATTLE step before returning a velocity [18]. Both are second-order methods and third order accurate locally [3]. They are also both second order convergent [18].

2.3.1 Iterative methods

In order to solve the constraining equations in RATTLE and SHAKE we use iterative methods to find approximate solutions. The general form of the iterative methods used to enforce the hidden constraint given in Equation 2.22d is given by the following algorithm, which constitutes Gauss-Newton iteration. In order to solve the equation

$$\mathbf{g}\left(\mathbf{q}_{n+1} - \mathbf{G}(\mathbf{q}_n)^T\Lambda\right) = \mathbf{0}, \tag{2.24}$$

the updates can be written as [14]:

$$\Lambda^{(i+1)} = \Lambda^{(i)} + \left[\mathbf{G}(\mathbf{Q}^{(i)})\mathbf{G}(\mathbf{q}_n)^T\right]\mathbf{g}(\mathbf{Q}^{(i)}), \tag{2.25}$$

$$\mathbf{Q}^{(i)} = \mathbf{q}_{n+1} - \mathbf{G}(\mathbf{q}_n)^T\Lambda^{(i)}. \tag{2.26}$$

We can initialize this method with $\Lambda^{(0)} = \mathbf{0}$. The method can more conveniently be written as

$$\mathbf{E}^{(l)}\Delta\mathbf{\Lambda}^{(l)} = \mathbf{g}(\mathbf{Q}^{(l)}), \tag{2.27}$$

$$\mathbf{E}^{(l)} = \mathbf{G}(\mathbf{Q}^{(l)})\mathbf{G}(\mathbf{q}_n)^T \tag{2.28}$$

where $\Delta\mathbf{\Lambda}^{(l)} = \mathbf{\Lambda}^{(l+1)} - \mathbf{\Lambda}^{(l)}$. When the solution is unique, the constraints are [sufficiently] smooth and $\mathbf{E}$ is well conditioned, the method converges rapidly; for values $\mathbf{\Lambda}^{(i)}$ sufficiently near the solution we have

$$||\mathbf{\Lambda}^{(l+1)} - \mathbf{\Lambda}_*|| \leq C||\mathbf{\Lambda}^{(l)} - \mathbf{\Lambda}_*||^2, \tag{2.29}$$

although if the initial value isn't sufficiently close to the solution there is no guarantee of convergence. The above method deals with the positional constraints in the RATTLE and SHAKE algorithms, while in RATTLE we have an additional velocity constraint Equation 2.22d which can be satisfied by $\mathbf{\Lambda}_v^{n+1}$ solving the linear system:

$$\left[\mathbf{G}(\mathbf{q}_n)\mathbf{G}(\mathbf{q}_n)^T\right]\mathbf{\Lambda}_v^{n+1} = \mathbf{G}(\mathbf{q}_n)\left(\frac{2}{\Delta t}\mathbf{v}^{n-1/2} - \nabla_q U(\mathbf{q}_n)\right). \tag{2.30}$$

2.4 Probability measures on manifolds

Due to the assumption that $\mathbf{g}$ is smooth and that $\mathbf{G}$ as full row-rank everywhere, $\mathcal{M}$ is a smooth embedded submanifold of Euclidean space $\mathbb{R}^n$, and it has a metric $\bar{g}$ which is induced by the Euclidean matric in $\mathbb{R}^n$. Leimkuhler et al. then showed that the Riemannian measure on $\mathcal{M}$ was equal to the surface measure, and satisfies a Poincaré inequality [17], which is needed to show that the Markov diffusion process does in fact converge to equilibrium.

3 Langevin Dynamics

The Langevin equation is derived from the equations of motion of a Brownian particle suspended in a liquid, under the assumption that it experiences two forces:

1. A viscous drag, $\gamma\ \dot{\mathbf{q}}(t)$ representing friction.
2. A force $\eta(t)$ representing the impacts of the fluid's molecules on the particle.

where $\eta(t)$ is white noise as used in the Ornstein-Uhlenbeck process to model the collision [12]. In the case where the particle is subjected to an external potential force field, the equations of motion become [14]

$$\mathrm{d}\mathbf{q} = \mathbf{p}\ \mathrm{d}t, \tag{3.1}$$

$$\mathrm{d}\mathbf{p} = -\nabla_{\mathbf{q}} U(\mathbf{q})\ \mathrm{d}t\ - \gamma\mathbf{p}\ \mathrm{d}t\ + \sqrt{2\gamma k_B T}\ \mathrm{dW}(t), \tag{3.2}$$

where $\mathrm{W}(t)$ is a Wiener process, k_B is the Boltzmann constant and T is the temperature. We will let $\beta = (k_B T)^{-1}$ for simplicity. This is an Itô process, a stochastic differential equation (SDE) for which the solution can be represented as the sum of integrals with respect to time and the Wiener process:

$$\mathbf{q}(t) = \mathbf{q}(0) + \int_0^t \mathbf{p}(t)\ \mathrm{d}t, \tag{3.3}$$

$$\mathbf{p}(t) = \mathbf{p}(0) - \int_0^t \nabla_{\mathbf{q}} U(\mathbf{q}(t) + \gamma\mathbf{p}(t)\mathrm{d}t + \int_0^t \sqrt{2\beta^{-1}}\ \mathrm{dW}(t), \tag{3.4}$$

where the second integral in Equation 3.4 is an Itô integral. These equations can also be written as

$$\dot{\mathbf{q}} = \mathbf{p}, \tag{3.5}$$

$$\dot{\mathbf{p}} = \mathbf{F} - \gamma\mathbf{p} + \sqrt{2k_B T\gamma}\ \eta(t), \tag{3.6}$$

where $\eta(t)$ is a stationary zero-mean Gaussian process, which is the form that will be used in the remainder of the paper. It can be shown that the invariant distribution of $\mathbf{q}$ is proportional to $e^{-\beta U(\mathbf{q})}$. This gives Langevin dynamics a use case outside of the physical setting of the derivation, as a means of sampling from the invariant distribution. The equations given above are called the *underdamped* Langevin diffusion equations. The *overdamped* Langevin diffusion equation refers to the SDE

$$\dot{\mathbf{x}} = -\gamma^{-1}\nabla_{\mathbf{q}} U(\mathbf{x}) + \sqrt{\gamma^{-1}\beta^{-1}}\ \eta(t), \tag{3.7}$$

which is the high friction limit of the underdamped equations. Clearly both have the same invariant distribution, since the invariant distribution is independent of γ. The overdamped equation, being more straightforward, is often used in data science applications [31, 29]. γ is called the *damping coefficient* for the motion; an increase in γ results in slower moving particles, and since $\sqrt{\gamma}$ also weights the random term an increase in gamma also results in larger fluctuations. The effect of an increase in β is then to amplify the effect of the random element on the dynamics, and thus alter the equilibrium state.

3.1 Splitting methods for Langevin dynamics

Now we consider a family of splitting methods for the Langevin dynamics given in 3.5. The Ornstein-Uhlenbeck process alone preserves the invariant distribution, and since Hamiltonian dynamics preserve any function of the Hamiltonian, they also preserve the invariant distribution which is proportional to $\exp^{-\beta H}$. This motivates a splitting method which involves the Ornstein-Uhlenbeck process and the Hamiltonian dynamics, however since the Hamiltonian dynamics are

not integrable, they are split into the kinetic and potential term as described above. These steps are labeled $\mathcal{A}, \mathcal{B}$ and $\mathcal{O}$, referring to the kinetic energy update, the potential energy update and the Ornstein-Uhlenbeck update respectively. The steps are given below [14]:

$$\hat{\mathcal{F}}^{\mathcal{A}}_{\Delta t}(\mathbf{q}, \mathbf{p}) = (\mathbf{q} + \Delta t \mathbf{p},\ \mathbf{p}), \tag{3.8}$$

$$\hat{\mathcal{F}}^{\mathcal{B}}_{h}(\mathbf{q}, \mathbf{p}) = (\mathbf{q}, \mathbf{p} + \Delta t \mathbf{F}), \tag{3.9}$$

$$\hat{\mathcal{F}}^{\mathcal{O}}_{\Delta t}(\mathbf{q},\ \mathbf{p}) = \left(\mathbf{q},\ e^{-\gamma \Delta t}\mathbf{p} + \sqrt{k_b T(1 - e^{-2\gamma \Delta t})}\mathbf{R}(t)\right), \tag{3.10}$$

where $\mathbf{R}$(t) is a vector of i.i.d normal random numbers. This splitting strategy defines a family of schemes based on compositions of the individual components. These schemes are named after the order in which the elements are composed, for example the OBABO scheme is given by

$$\hat{\mathcal{F}}^{\mathcal{O}}_{\Delta t/2} \circ \hat{\mathcal{F}}^{\mathcal{B}}_{\Delta t/2} \circ \hat{\mathcal{F}}^{\mathcal{B}}_{\Delta t} \circ \hat{\mathcal{F}}^{\mathcal{B}}_{\Delta t/2} \hat{\mathcal{F}}^{\mathcal{O}}_{\Delta t/2}, \tag{3.11}$$

noting that steps which are performed twice are given a half-step of $h/2$. Of these methods, BAOAB is considered preferable as it has shown to be stable at larger step-sizes than alternative methods [13].

3.2 Constrained Langevin dynamics

We can write the Langevin dynamics with $\mathbf{q}$ constrained such that $\mathbf{g}(\mathbf{q}) = 0$ as:

$$\dot{\mathbf{q}} = \mathbf{p}, \tag{3.12a}$$

$$\dot{\mathbf{p}} = \mathbf{F} - \gamma \mathbf{p} + \sqrt{2\beta^{-1}\gamma}\ \eta(t) - \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{q}), \tag{3.12b}$$

$$0 = g_j(\mathbf{q}),\ j = 1, 2, ..., m, \tag{3.12c}$$

$$0 = \nabla g_j(\mathbf{q})^T \mathbf{p},\ j = 1, 2, ...m, \tag{3.12d}$$

where m is the number of constraints and $\mathbf{F} = -\nabla U(\mathbf{q})$ is the force. The co-tangency constraint in this formulation, Equation 3.12d, is redundant in theory, but it will be useful in discretising the equations. It can be shown that in the constrained case above, the invariant density can be written as [15]

$$\rho^g \propto \exp\left(-\beta H(\mathbf{q}, \mathbf{p})\right) \prod_{ij} \delta^m_{i=1}[g_i(\mathbf{q})] \prod_{i=1}^{m} \delta[\nabla g_i(\mathbf{q})^T \mathbf{p}]. \tag{3.13}$$

The invariant distribution of $\mathbf{q}$ is then:

$$\rho(\mathbf{q}) \propto \int_{\mathbf{p}'} \exp\left(-\beta(U(\mathbf{q}) + \frac{1}{2}||\mathbf{p}'||^2)\mathbb{P}(\mathbf{p} = \mathbf{p}')\right) \prod_{i=1}^{m} \delta_i[g_i(\mathbf{q})] \prod_{i=1}^{m} \delta[\nabla g_i(\mathbf{q})^T \mathbf{p}] d\mathbf{p}' \tag{3.14}$$

$$= \exp\left(-\beta U(\mathbf{q})\right) \prod_{ij} \delta^m_{i=1}\left[g_i(\mathbf{q})\right]\ C \tag{3.15}$$

for some constant C, since $\{\nabla g_i(\mathbf{q}) : i \in [1, \dots m]\}$ are linearly independent, which gives:

$$\rho(\mathbf{q}) \propto \exp\left(-\beta U(\mathbf{q})\right) \tag{3.16}$$

for all $\mathbf{q}$ such that $g_i(\mathbf{q}) = 0$ for all $i \in \{1, \dots, m\}$.

3.2.1 Geodesic Integration

This section introduces a constraint-preserving integration scheme for constrained Langevin dynamics which is symplectic in the deterministic case where $\gamma = 0$. Splitting the constrained Langevin equations into potential and kinetic energies and the Ornstein-Uhlenbeck process gives [14]:

$$\begin{bmatrix} \mathrm{d}\mathbf{q} \\ \mathrm{d}\mathbf{p} \\ \mathbf{0} \end{bmatrix} = \underbrace{\begin{bmatrix} M^{-1}\mathbf{p}\ \mathrm{d}t \\ -\mathbf{G}(\mathbf{q})^T\lambda\ \mathrm{d}t \\ \mathbf{g}(\mathbf{q}) \end{bmatrix}}_{A} + \underbrace{\begin{bmatrix} \mathbf{0} \\ -\nabla U(\mathbf{q}))\ \mathrm{d}t - \mathbf{G}(\mathbf{q})^T\lambda\ \mathrm{d}t \\ \mathbf{g}(\mathbf{q}) \end{bmatrix}}_{B} + \underbrace{\begin{bmatrix} \mathbf{0} \\ -\gamma\mathbf{p}\ \mathrm{d}t + \sqrt{2\gamma k_B T\mathbf{M}}\ \mathrm{d}\mathbf{W} - \mathbf{G}(\mathbf{q})^T\lambda\ \mathrm{d}t \\ \mathbf{g}(\mathbf{q}) \end{bmatrix}}_{O}, \tag{3.17a}$$

where the $\mathcal{A}$ step comes from the kinetic energy, the $\mathcal{B}$ step from the potential and the $\mathcal{O}$ step from the Ornstein-Uhlenbeck process. The solution of part $\mathcal{A}$ is a geodesic curve, meaning that it has no force applied and it runs along the constraint surface, and it can be approximated using SHAKE or RATTLE [15]. Multiple steps of these algorithms can be taken within one step of the proposed algorithm, adding relatively little computational load, since they don't require an evaluation of the force function which is in practice the most expensive part of the algorithm.

For the solution of the $\mathcal{B}$-step, we get [15]

$$0 = \mathbf{G}(\mathbf{q})\mathbf{p} \tag{3.18}$$

$$\implies 0 = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{G}(\mathbf{q})\mathbf{p} + \mathbf{G}(\mathbf{q})\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} \tag{3.19}$$

$$\implies \mathbf{G}(\mathbf{q})\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = 0 \tag{3.20}$$

$$\implies \lambda = -\left[\mathbf{G}\mathbf{G}^T\right]^{-1}\mathbf{G}\nabla U(\mathbf{q}) \tag{3.21}$$

Where we used $\mathbf{G}$ as shorthand for $\mathbf{G}(\mathbf{q})$. This implies that the $\mathcal{B}$-step can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = -\mathbf{\Pi}\nabla U(\mathbf{q}), \tag{3.22}$$

where $\mathbf{\Pi}$ is a projector onto the cotangent space, $\mathbf{\Pi} : \mathcal{M} \to T\mathcal{M}$, defined by

$$\mathbf{\Pi} = \mathbf{I} - \mathbf{G}^T\left[\mathbf{G}\mathbf{G}^T\right]^{-1}\mathbf{G}. \tag{3.23}$$

Therefore the $\mathcal{B}$-step can be solved exactly as

$$\mathbf{q}(t) = \mathbf{q}(0) \tag{3.24}$$

$$\mathbf{p}(t) = \mathbf{p}(0) - t\mathbf{\Pi}(\mathbf{q}(0))\nabla U(\mathbf{q}) \tag{3.25}$$

which is called a projected "kick" of the force onto the tangent space of $\mathcal{M}$.

Similarly for the O-step we get an SDE of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p} = \gamma\mathbf{\Pi}\mathbf{p} + \sqrt{2\gamma k_B T}\mathbf{\Pi}\ \eta(t), \tag{3.26}$$

which can be weakly solved by

$$\mathbf{p}(t) = e^{-\gamma t\mathbf{\Pi}}\mathbf{p}(0) + \sqrt{k_B T}\left[I - e^{-2\gamma t\mathbf{\Pi}}\right]^{1/2}\mathbf{\Pi}\mathbf{R}(t), \tag{3.27}$$

where $\mathbf{R}(t)$ is a vector of i.i.d normal random variables. Since $\mathbf{\Pi}^2 = \mathbf{\Pi}$, we have

$$e^{-\gamma t \mathbf{\Pi}} = \mathbf{I} + (e^{\gamma t} - 1)\mathbf{\Pi} \tag{3.28}$$

$$\implies \left[I - e^{-2\gamma t \mathbf{\Pi}}\right]^{1/2} = \sqrt{1 - e^{-2\gamma t}}\mathbf{\Pi} \tag{3.29}$$

and if we choose the initial velocity to satisfy the tangency condition so that $\mathbf{\Pi p}(0) = \mathbf{p}(0)$, then we have

$$\mathbf{p}(t) = e^{-\gamma t}\mathbf{p}(0) + \sqrt{\beta^{-1}(1 - e^{-2\gamma t})\mathbf{\Pi}}\mathbf{R}(t). \tag{3.30}$$

We can now use these solutions of the individual steps to construct integrators based on their compositions.

3.2.2 g-BAOAB

The geodesic BAOAB method uses a similar splitting to that used in unconstrained BAOAB, but ensures that both constraints are satisfied for each step. The name geodesic BAOAB comes from the fact that the solution of the $\mathcal{A}$ step is a geodesic along the surface of the manifold. The algorithm is defined as follows:

Algorithm 1 g-BAOAB Algorithm

$\mathbf{p} \leftarrow \mathbf{p} + \frac{dt}{2}F(\mathbf{q}) + \sum_j \mu_j G_j(\mathbf{q}) \in T^*_q\mathcal{M}$ ▷ $\frac{1}{2}\mathcal{B}$−step
for k from 1 to K_r **do**
  $(\mathbf{q}, \mathbf{p}) \leftarrow A(\mathbf{q}, \mathbf{p}, \frac{dt}{2K_r})$
end for ▷ $\frac{1}{2}\mathcal{A}$−step
$\mathbf{p} \leftarrow a_2\mathbf{p} + b_2\mathbf{M}^{1/2}R + \sum_j \mu_j G_j(\mathbf{q}) \in T^*_q\mathcal{M}$ ▷ $\mathcal{O}$−step
for k from 1 to K_r **do**
  $(\mathbf{q}, \mathbf{p}) \leftarrow A(\mathbf{q}, \mathbf{p}, \frac{dt}{2K_r})$
end for ▷ $\frac{1}{2}\mathcal{A}$−step
$\mathbf{p} \leftarrow \mathbf{p} + \frac{dt}{2}F(\mathbf{q}) + \sum_j \mu_j G_j(\mathbf{q}) \in T^*_q\mathcal{M}$ ▷ $\frac{1}{2}\mathcal{B}$−step.

The solution of the $\mathcal{A}$-step is calculated using the RATTLE algorithm with a force of zero using a total of K_r steps. As shown by Leimkuhler and Matthews [15], a low value of K_r is sufficient for g-BAOAB and larger values don't provide a significant difference in stability or error.

3.3 Example

The g-BAOAB algorithm was used to generate the samples used in Figure 6. The potential function used was $U(x, y, x) = (x^2 - 1)^2 + (y^2 - 1)^2 + (z^2 - 1)^2$. The samples used in Figure 6 were generated using $50,000$ steps for 6 different initial points on the surface of the sphere, for a total of $300,000$ samples.

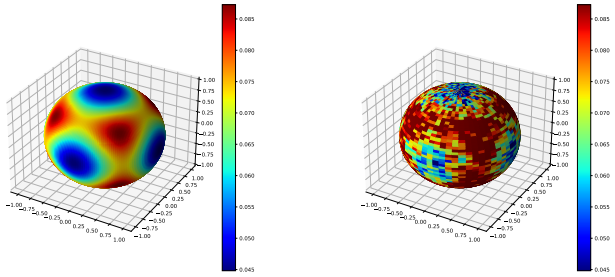


Figure 6: Invariant density compared to g-BAOAB for the potential function $U(x, y, x) = (x^2 - 1)^2 + (y^2 - 1)^2 + (z^2 - 1)^2$

In Figure 7 we can see that the algorithm converges to the correct theoretical values of the expectations of the observables. The speed of convergence is slower and the bias is larger, somewhat counter intuitively, for lower stepsizes. This is particularly surprising when we remember that due to the fact that the $x-$ axis represents time rather than iterations, meaning that with lower step-sizes we can converge more slowly to the true value than a larger step-size even while taking more steps.

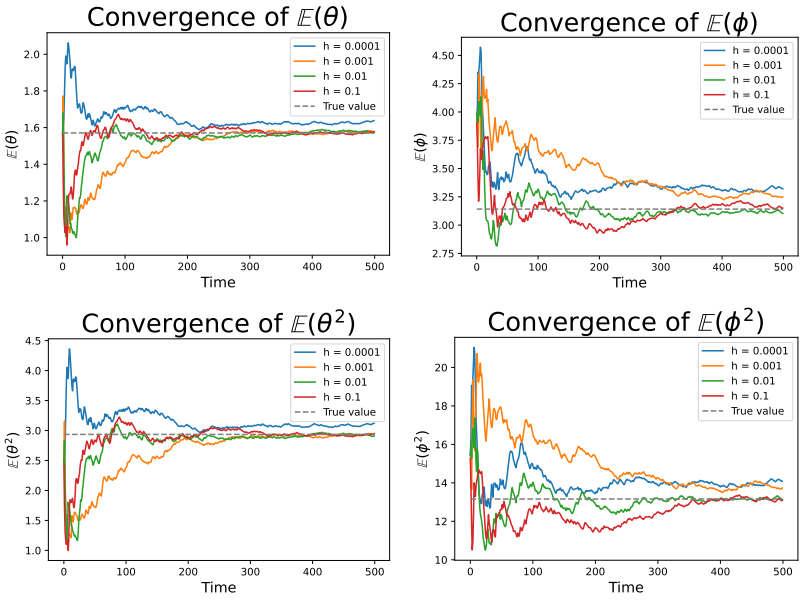


Figure 7: Convergence of ϕ, θ, ϕ^2 and θ^2 for different stepsizes h. (note plot with math h)

The number of RATTLE iterations performed per $\mathcal{A}$-step, K_r, was seen to have effects on how accurately the solutions satisfied the constraints, as shown in Figure 8. The errors decrease rapidly with higher values of K_r, with $K_r = 2$ already bringing the error in the constraint below machineprecision. The iterations in Figure 8 were run with a step size of 0.2, $\gamma = 1$ and $\beta = 1$. While in general it's preferable to run the Newton iterations with an error tolerance as a stopping criterion, this may not always be practical.

Again slightly counter intuitively, we see from Figure 9 that convergence is also much slower for lower values of γ. There are two possible causes for this stemming from the fact that γ is a factor in two terms in the equations. A low γ in the damping term could mean that the particles end up moving too quickly and essentially overshoot high density areas. On the other hand, the lower weighting on the random term could mean that the particles get trapped in local maxima in low-density areas. In this case, the latter are not present in the density function, so we can conclude that it is the due to the former explanation.

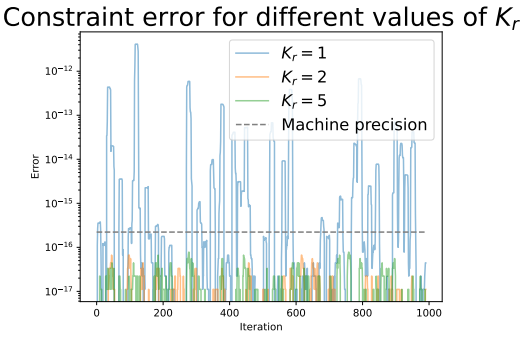


Figure 8: 10 point rolling average of error per iteration for different values of K_r

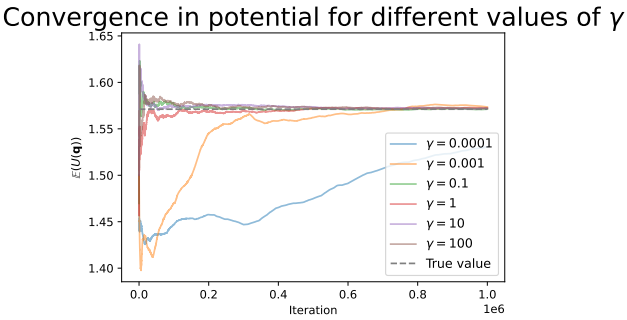


Figure 9: Convergence in expected potential $U(\mathbf{q})$ for different values of γ

4 Score based diffusion

Now that we have seen how the g-BAOAB integrator can be used for solving SDEs subject to holonomic constraints, we move into the area of SBD, where g-BAOAB can be used to noisify data on the constraint manifold, and reverse the noising process while remaining on the constraint manifold. Key to this process is approximating the score function of the marginal distribution.

4.1 Score matching

For an SDE of the form given in Equations 3.12, our aim is to construct an estimator of the score function of the marginal distribution of $\mathbf{q}$, $\hat{\mathbf{s}}_\theta(\mathbf{q}, t) \approx \nabla_{\mathbf{q}} \log \rho_t(\mathbf{q})$, where ρ_t is the distribution at time t in the noising process. The score function is intractable, and needs to be approximated rather than calculated exactly. The fact that the score-based model $s_\theta(\mathbf{x})$ is independent of the normalizing constant allows us to use a variety of models, since the normalizing constant does not need to be tractable. Usually, training the model score function involves minimizing the Fisher divergence [30]:

$$\mathcal{L}(\hat{s}_\theta) = \int_0^T \mathbb{E}_\rho \left[\nabla_{\mathbf{q}} \log \rho_t(\mathbf{x}) - \hat{s}_\theta(\mathbf{x}, t) ||_2^2 \right] \, \mathrm{d}t, \tag{4.1}$$

where $\mathbb{E}_\rho$ signifies that the expectation is taken over $\rho(\mathbf{x})$. Although the Fisher divergence includes the *true* score function which is intractable, Hyvärinen showed that it can remarkably be written purely in terms of the *model* score function, plus a constant. In the following theorem we deal a score model which is not time-dependent, and write $\hat{s}_\theta(\mathbf{x})$ rather than $\hat{s}_\theta(\mathbf{x}, t)$, but the time-dependent version will follow.

Theorem 4.1. [11] *Assume that the model score function $\hat{s}(\mathbf{x})$ is differentiable, the data distribution is differentiable, the expectation of the squared model density and the expectation of the squared norm of the score are finite. Also assume that $\rho(\mathbf{q})\hat{\rho}(\mathbf{q}) \to 0$ as $||\mathbf{q}|| \to \infty$. Then the*

objective in Equation 4.1 can be expressed as

$$\mathcal{L}(\hat{\mathbf{s}}_\theta) = \mathbb{E}_\rho \left[Tr(\nabla_x \hat{\mathbf{s}}_\theta(\mathbf{x})) + \frac{1}{2}||\hat{\mathbf{s}}_\theta(\mathbf{x})||_2^2 \right] + C, \tag{4.2}$$

where $\nabla_x \hat{s}_\theta(\mathbf{x})$ *is the Jacobian matrix of the score model evaluated at* $\mathbf{x}$*.*

Thus we can train the score-based model by calculating its norm and divergence on samples of the data, without any explicit knowledge of the true score function. This fact is key to score matching and thus score-based generative modelling. The proof of the theorem is based on integration by parts; we have that

$$\mathcal{L}(\hat{\mathbf{s}}_\theta) = \int \rho(\mathbf{x}) \left[\frac{1}{2}||\hat{\mathbf{s}}_\theta(\mathbf{x})||^2 + \frac{1}{2}||\mathbf{s}(\mathbf{x})||^2 - \hat{s}_\theta(\mathbf{x})^T \mathbf{s}(\mathbf{x}) \right] \mathrm{d}\mathbf{x} \tag{4.3}$$

where the integration domain has been omitted for simplicity. Then it must be shown that

$$\int \rho(\mathbf{q}) \hat{s}_\theta^T \mathbf{s}(\mathbf{q}) d\mathbf{q} = \int \rho(\mathbf{q}) \nabla \hat{s}_\theta d\mathbf{q}. \tag{4.4}$$

The partial integration is based on a generalization of the following statement to multiple dimensions: Given a one dimensional density function ρ and a function f, we have

$$\int \rho(\mathbf{q})(\log \rho)'(\mathbf{q}) f(\mathbf{q}) d\mathbf{q} = -\int \rho(\mathbf{q}) f'(\mathbf{q}) d\mathbf{q}, \tag{4.5}$$

under the assumption that f is differentiable, and that the limit goes to zero at infinity as stated in the theorem. Using this theorem, we can use the following estimator of the Fisher divergence to train the score model:

$$\mathcal{L}(\hat{s}_\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\mathrm{Tr}\left(\nabla_x \hat{s}_\theta(\mathbf{x})\right) + \frac{1}{2}||\hat{s}_\theta(\mathbf{x})||_2^2 \right]. \tag{4.6}$$

Note that this theorem implies that the minimum value for the loss is $-C = -\mathbb{E}_\rho(\frac{1}{2}||\mathbf{s}(\mathbf{q})||^2)$, so losses can and generally should be negative but we won't know what the minimum value, or even what is an acceptable value, in advance. Finally, we can generalize this result to a continuous time-dependent vector field $\hat{s}_\theta$ which aims to minimize the loss function in Equation 4.1, where we get the loss function

$$\mathcal{L}(\hat{s}_\theta) \approx \frac{1}{T} \sum_{j=1}^{M} \frac{1}{N} \sum_{i=1}^{N} \left[\mathrm{Tr}(\nabla_{\mathbf{x}_i} \hat{s}_\theta(\mathbf{x}_i, t_j) + \frac{1}{2}||\hat{s}_\theta(\mathbf{x}_i, t_j)||_2^2 \right] \tag{4.7}$$

where the times $t_j,\ j \in [1, \ldots, M]$ are sampled uniformly. Note that we write $\hat{s}_\theta$ with θ representing a vector of parameters to indicate that $\hat{s}_\theta$ is representing an approximation which will be parameterized as a neural network.

4.1.1 Score-matching on Riemennian manifolds

It can be shown that on a Riemannian manifold, we get an expression which is directly analogous to the score-matching loss used in Euclidean space. We define the *denoising score-matching* loss function as

$$\mathcal{L}(\hat{s}_\theta) = \int_{\mathcal{M}^2} ||\nabla_{\mathbf{q}} \log \rho_{t|s}(\mathbf{q}_t|\mathbf{q}_s) - \hat{s}_\theta(\mathbf{q}_t, t)|| \mathrm{d}\mathbb{P}_{s,t}(\mathbf{q}_s, \mathbf{q}_t), \tag{4.8}$$

Then by a proof which is analogous to that of Theorem 4.1, to which is minimized by then we have the following theorem:

Theorem 4.2. [8] *Under sufficient regularity of $\rho_{t|s}(\mathbf{q}_t|\mathbf{q}_s)\hat{s}_\theta(\mathbf{q}_t)$ for all $t, s \in (0, T]$ with $t > s$, we have*

$$\mathcal{L}_{t|s}(\hat{s}_\theta) = \int_{\mathcal{M}} \left(\frac{1}{2} ||\hat{s}_\theta(\mathbf{q}_t, t)||^2 + \nabla \cdot \hat{s}_\theta(\mathbf{q}_t, t) \right) \, \mathrm{d}\mathbb{P}_t(\mathbf{q}_t) + C, \tag{4.9}$$

for some constant C.

The proof requires regularity conditions on $\rho_{t|s}$ which are discussed in Appendix C This theorem in combination with a relationship between the Riemannian and the Euclidean divergence will, under certain conditions, let us express the loss funtion in terms of the Euclidean divergence. First, we denote by $\mathcal{X}(\mathcal{M})$ the space of smooth tangent vector fields to $\mathcal{M}$. Then:

Theorem 4.3. [25] *If $u \in \mathcal{X}(\mathbb{R}^n)$, and the restriction of u on $\mathcal{M}$, $u|_{\mathcal{M}}$ is infinitesimally constant in the normal directions of $\mathcal{M}$, then for $\mathbf{q} \in \mathcal{M}$, $\nabla \cdot u(\mathbf{q}) = \nabla_E \cdot u(\mathbf{q})$, where ∇_E is the standard Euclidean divergence.*

In other words, given a vector field which is tangent to the manifold at each point, $u(\mathbf{q}) \in T_{\mathbf{q}}\mathcal{M} \; \forall \mathbf{q} \in \mathcal{M}$, its divergence is equal to its Euclidean divergence, and since we have an explicit projection matrix $\mathbf{\Pi}(\mathbf{q})$ onto $T_{\mathbf{q}}\mathcal{M}$, we can construct such a vector field using projections.

4.2 Perturbing data

For regions of low-density data, it can be difficult to accurately estimate the score. Since score matching aims to minimize the Fisher divergence (4.1), the low density regions of the [space] naturally contribute relatively little to the overall loss, since the norms are weighted by $\rho(\mathbf{x})$. To remedy this we can perturb the data points with noise, which will populate low density areas with points. Generally multiple scales of Gaussian noise have been used to ensure that we cover low-density areas [31]. A noise-conditional score-based model is then trained on the noisy data. The training objective for the noise-conditional score-based model is given by a weighted sum of Fisher divergences for different noise scales:

$$\sum_{i=1}^{L} \zeta(i) \mathbb{E}_{\rho_i} \left[\nabla_{\mathbf{x}} \log \rho_i(\mathbf{x}) - \hat{s}_\theta(\mathbf{x}, i) ||_2^2 \right] \tag{4.10}$$

where ρ_i is the distribution of $\mathbf{x}$ perturbed with noise with standard deviation σ_i and $\zeta(i)$ is some positive weighting function, usually taken to be $\zeta(i) = \sigma_i$. Note that outside of the Hamiltonian setting we will deviate from referring to the variable of interest as $\mathbf{q}$ and replace this with $\mathbf{x}$.

However recently, instead of using this discrete noising process, an SDE has been used to perturb the data, effectively generalizing the number of noise scales to infinity [31]. In perturbing the data using an SDE, we must bear in mind that the invariant distribution of the SDE used should be known, since we will need to sample from the invariant distribution as the *initial* distribution of the reverse process.

Choosing a suitable distribution on manifolds can present challenges. The uniform distribution seems a sensible choice, however a uniform distribution cannot be defined if the manifold is not closed. In that case, we would need to choose a potential function $U(\mathbf{q})$ so that the Langevin dynamics give a stationary distribution we can sample from. Two distributions proposed by De Bortoli et al. are the "Riemannian normal" and the "exponential wrapped" Gaussian [8]. For a closed manifold, the uniform distribution on the manifold is the invariant distribution given by Langevin dynamics with a force term of zero. In the overdamped limit this equation is a Brownian motion on the manifold and is given by

$$\dot{\mathbf{q}} = \sqrt{2k_B T \gamma} \mathbf{M}^{1/2} \eta(t) - \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{q}), \tag{4.11}$$

$$g_i(\mathbf{q}) = 0, \; i = 1, \ldots, m. \tag{4.12}$$

However we can also use a system of underdamped Langevin dynamics, and since the invariant distribution is proportional to $e^{-U(\mathbf{q})}$, letting $U(\mathbf{q}) = 0$ gives $p(\mathbf{q}) \propto e^{-0} = 1$ which is a uniform distribution on the manifold. This gives us the familiar noising process

$$\dot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{p}, \tag{4.13}$$

$$\dot{\mathbf{p}} = -\gamma\mathbf{p} + \sqrt{2k_BT\gamma}\mathbf{M}^{1/2}\eta(t) - \sum_{i=1}^{m}\lambda_i\nabla g_i(\mathbf{q}), \tag{4.14}$$

$$0 = g_j(\mathbf{q}),\ j = 1, \ldots, m, \tag{4.15}$$

$$0 = \nabla g_j(\mathbf{q})^T M^{-1}\mathbf{p},\ j = 1, \ldots, m, \tag{4.16}$$

from which we can generate samples using the g-BAOAB algorithm.

4.3 Time Reversal

Central to the idea of SBD is the reverse SDE [31], which allows us to move backwards in time from a known distribution to the unknown data distribution. Now that we have a noising process using constrained underdamped Langevin dynamics, we aim to construct the reverse SDE corresponding to our formulation of constrained Langevin dynamics. This will allow us to move backwards from noised data to data which follows the data distribution.

4.3.1 Time reversal of constrained Langevin dynamics

As shown by [4], any SDE in Euclidean space

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) + \mathbf{g}(t)\eta(t) \tag{4.17}$$

yields a time-reversed SDE, given by

$$\dot{\mathbf{x}} = \mathbf{f}(x, t) - \mathbf{g}^2(t)\nabla_x \log \rho_t(x) + \mathbf{g}(t)\eta(t), \tag{4.18}$$

where can avoid explicitly reversing the time by imagining $d\mathbf{x}$ as another process moving forward through time, where $p_t(\mathbf{x})$ is the distribution of $\mathbf{x}$ at time t in the *noising* process [4].

In order to perform SBD on the manifold however, we must find the reverse-time formulation of the constrained Langevin dynamics SDE given in Equations 3.12. As shown in by Lelievre et al., if $\mathcal{G}_t^f$ is the generator of the forward dynamics, then the backward dynamics can be defined through its generator by first reversing the momentum in the initial condition, then reversing it *back* after the time evolution [20]:

$$\mathcal{G}_{t'}^b = \mathcal{R}\mathcal{G}_{T-t'}^f\mathcal{R} \tag{4.19}$$

where $\mathcal{R} : \phi \to \phi \circ S$ where S is an operator which flips the momentum, $S(\mathbf{q}, \mathbf{p}) = S(\mathbf{q}, -\mathbf{p})$

This means that constrained Langevin dynamics of the form given in Equations 3.12 yield the reverse equations:

$$\dot{\mathbf{q}} = -\mathbf{M}^{-1}\mathbf{p}, \tag{4.20}$$

$$\dot{\mathbf{p}} = -\mathbf{F}(\mathbf{q}) - \gamma\mathbf{p} + \sqrt{2k_BT\gamma}\eta(t) - \sum_{i=1}^{m}\lambda_i\nabla g_i(\mathbf{q}), \tag{4.21}$$

$$0 = g_j(\mathbf{q}),\ j = 1, 2, \ldots, m, \tag{4.22}$$

$$0 = \nabla g_j(\mathbf{q})^T M^{-1}\mathbf{p},\ j = 1, 2, ...m. \tag{4.23}$$

Modifying the g-BAOAB algorithm for the reversed dynamics, we have that the $\mathcal{B}-$step can be

solved exactly as

$$\mathbf{q}(t) = \mathbf{q}(0),\ \mathbf{p}(t) = \mathbf{p} - t\mathbf{\Pi}(\mathbf{q}(0))\mathbf{F}. \tag{4.24}$$

The $\mathcal{O}$-step doesn't change, and the $\mathcal{A}$-step is simply given by reversing the sign of the velocity. For completeness, the algorithm is given below:

Algorithm 2 Reverse g-BAOAB algorithm

$\mathbf{p} \leftarrow \mathbf{p} - \frac{dt}{2}F(\mathbf{q}) + \sum_j \mu_j G_j(\mathbf{q}) \in T^*_q\mathcal{M}$ ▷ $\frac{1}{2}\mathcal{B}$−step
for k from 1 to K_r **do**
  $(\mathbf{q}, -\mathbf{p}) \leftarrow A(\mathbf{q}, -\mathbf{p}, \frac{dt}{2K_r})$
end for ▷ $\frac{1}{2}\mathcal{A}$−step
$\mathbf{p} \leftarrow a_2\mathbf{p} + b_2\mathbf{M}^{1/2}R + \sum_j \mu_j G_j(\mathbf{q}) \in T^*_q\mathcal{M}$ ▷ $\mathcal{O}$−step
for k from 1 to K_r **do**
  $(\mathbf{q}, -\mathbf{p}) \leftarrow A(\mathbf{q}, -\mathbf{p}, \frac{dt}{2K_r})$
end for ▷ $\frac{1}{2}\mathcal{A}$−step
$\mathbf{p} \leftarrow \mathbf{p} - \frac{dt}{2}F(\mathbf{q}) + \sum_j \mu_j G_j(\mathbf{q}) \in T^*_q\mathcal{M}$ ▷ $\frac{1}{2}\mathcal{B}$−step

5 Connection with Ordianry Differential Equations

One issue with SBD is that it doesn't give a likelihood for the density of the model, and it also doesn't necessarily maximize the likelihood of the observations. In order to link the SBD SDE to the likelihood, we use CNFs, which are ODEs describing the rate of change of a probability density function.

5.1 Normalizing flows

Normalizing flows provide a mechanism for defining probability distributions based on a base distribution. In unconstrained problems, we express points $\mathbf{x}$ as a transformation of a real vector $\mathbf{u}$ sampled from the base distribution $\rho_u(\mathbf{u})$. For a flow based model, T must be invertible and T^{-1} and T must be differentiable. Then we can use the change of variables formula

$$\rho_x(\mathbf{x}) = \rho_u(\mathbf{T^{-1}}(\mathbf{x}))|\det J_T(\mathbf{u})|^{-1}, \tag{5.1}$$

where J_T is the Jacobian matrix of T, using a change of variables [23]. The name *normalizing* flow refers to the fact that the change of variables gives a normalized density.

It's been shown by Papamakarios et al. that for any pair of sufficiently well behaved distributions τ and ρ, there is exists a diffeomorphism ϕ which maps τ to ρ, where the requirements on the distributions are only that they are differentiable and strictly positive [23]. We will later use τ and ρ to refer to our known initial distribution and the data distribution, respectively, but here they can be thought of as any general probability distributions.

5.1.1 Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence is one of the most popular objective functions used in flow-based models (cite uses); it is given by the expectation of the logarithmic difference between probabilities. The forward KL divergence between the target distribution ρ and the modelled distribution $\hat{\rho}$ is given by:

$$D_{KL}[\rho|\hat{\rho}] = \mathbb{E}\left[\log\left(\frac{\rho}{\hat{\rho}}\right)\right] \tag{5.2}$$

$$= -\mathbb{E}_\rho\left[\log \rho\mathbf{x}\right] + C \tag{5.3}$$

$$= \mathbb{E}_\rho\left[\log \rho_u(T^{-1}(\mathbf{x}))\right] + \log|\det J_{T^{-1}}(\mathbf{x})|] \tag{5.4}$$

The minimizer of the KL divergence can be approximated by minimizing the loss function:

$$\mathcal{L}(\rho, \tau) = -\frac{1}{N} \sum_{n=1}^{N} \left(\log \tau(T^{-1}(\mathbf{x_n})) + \log |\det J_{T^{-1}}(\mathbf{x})| \right), \tag{5.5}$$

which can be minimized in practice using gradient descent, using its gradients with respect to the parameters of ρ, θ_ρ and the parameters of τ, θ_τ:

$$\nabla_{\theta_\rho} \mathcal{L}(\rho, \tau) = -\frac{1}{N} \sum_{n=1}^{N} \nabla_{\theta_\rho} \log \tau(T^{-1}(\mathbf{x_n})) + \nabla_{\theta_\rho} \log |\det J_{T^{-1}}(\mathbf{x})|, \tag{5.6}$$

$$\nabla_{\theta_\tau} \mathcal{L}(\rho, \tau) = -\frac{1}{N} \sum_{n=1}^{N} \log \tau(T^{-1}(\mathbf{x_n})). \tag{5.7}$$

The KL divergence is useful since we can estimate it using samples of the target distribution, and it can be shown that maximising the likelihood is asymptotically equivalent to minimizing the KL divergence, ie. they are equivalent as the number of samples goes to infinity.

5.1.2 Continuous normalizing flows

Now considering a continuous time version of the transformation gives an ODE

$$\dot{\mathbf{z}} = g(t, \mathbf{z}) \tag{5.8}$$

where the function g is Lipschitz continuous, and continuous in t. The transformed variable $\mathbf{x}$ is then given by $\mathbf{x} = \mathbf{u} + \int_{t=0}^{t=1} g_\phi(t, \mathbf{z}_t) dt$. We define the *pushforward* map so that the pushforward of the density by the transformation is the density of the transformed variable, ie. $\mathbf{z} = \phi(\mathbf{z}_0)$ with $\mathbf{z}_0 \sim \tau \implies \mathbf{z} \sim \phi_* \tau$ [22]. The ODE can then be solved using, for example, a simple ODE discretisation such as Euler's method:

$$\mathbf{z}_{t+h} \simeq \mathbf{z}_t + h g(t, \mathbf{z}_t) \tag{5.9}$$

It can be shown that the rate of change of the log-density can be characterized in terms of the trace of the Jacobian matrix [23]:

$$\frac{\mathrm{d}}{\mathrm{d}t} \log \rho(\mathbf{z}_t) = -\mathrm{Tr}(J_{g_\phi(t,)}(\mathbf{z}_t), \tag{5.10}$$

which is obtained from a Fokker-Planck equation with a diffusion term of zero [23]. Thus we can evaluate both the transformation and the log-density by integrating:

$$\begin{bmatrix} \mathbf{x} \\ \log \rho(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \log \rho(\mathbf{u}) \end{bmatrix} + \int_{t=0}^{t=1} \begin{bmatrix} g(t, \mathbf{z}_t) \\ -\mathrm{Tr}(J_{g_\phi(t,)}(\mathbf{z}_t) \end{bmatrix} dt \tag{5.11}$$

using ODE solvers and then update the parameters based on a backward pass, in order to minimize the KL divergence.

5.2 Riemanninan normalizing flows

CNFs can also be constructed on Riemannian manifolds. Consider a particle $\mathbf{q}_t$ lying on the manifold $\mathcal{M}$. Then let f be a vector field, $f : \mathcal{M} \times \mathbb{R} \to T\mathcal{M}$, where $T_{\mathbf{q}}\mathcal{M}$ is the tangent space at $\mathbf{q}$ and $T\mathcal{M} = \cap_{\mathbf{q} \in \mathcal{M}} T_{\mathbf{q}}\mathcal{M}$ is the tangent bundle. We refer to $f(;t)$ as f_t, the vector field at time t. Then the particle's time-evolution according to f is given by [22]:

$$\frac{\mathrm{d}}{\mathrm{d}t} \mathbf{q}_t = f_t(\mathbf{q}_t). \tag{5.12}$$

Then when starting at an initial position $\mathbf{q}_0$, the flow operator $\phi : \mathcal{M} \times \mathbb{R} \to \mathcal{M}$ is defined so that

$$\phi(\mathbf{q}_0, t) = \mathbf{q}_t \tag{5.13}$$

gives the position of the particle at time t. Mathieu and Nickel give a generalization of the log-probability flow given in Equation 5.30 for the flow on a Riemannian manifold in the following theorem:

Theorem 5.1. [22] *Let $\mathbf{q}_t$ be a continuous manifold-valued random variable which is described by the ODE from 5.12 with probability density $\rho(\mathbf{q}_t)$. Then the change in log-probability follows a differential equation given by*

$$\frac{\mathrm{d}}{\mathrm{d}t} \log \rho(\mathbf{q}_t) = -\nabla_{\mathcal{M}} \cdot f_t(\mathbf{q}_t). \tag{5.14}$$

Therefore we can also solve for the change in log-density between two manifold-valued random variables, and notably we have that

$$\log\left(\frac{\rho}{\tau}\right) = -\int_0^1 \nabla_{\mathcal{M}} \cdot f_t(\mathbf{q}_t)\ \mathrm{d}t, \tag{5.15}$$

where $\nabla_{\mathcal{M}}$ is the divergence of the vector field on the manifold, which we will denote from now on simply by ∇, with ∇_{E} representing the Euclidean divergence This will allow us to evaluate the transformation by integrating using an ODE solver, and then to minimize the log-difference, or KL divergence, by backpropogating and updating the parameters.

5.3 Score-based flow

In the unconstrained case, score-based models can be much more efficient to train than CNFs, because minimizing the maximum likelihood while training CNFs requires running an expensive ODE solver for each optimization step. However an issue with SBD is that training the model doesn't necessarily maximize the likelihood, which may be desirable in certain cases. However, we can model the flow of the marginal distribution of the SDE, as shown by Song [29]. For a stochastic differential equation given by

$$\dot{\mathbf{x}} = f(\mathbf{x}, t) + \mathbf{g}(t)\eta(t), \tag{5.16}$$

there is a deterministic process sharing the same marginal densities as the SDE, which is an example of a CNF, given by

$$\dot{\mathbf{x}} = f(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \hat{s}_\theta(\mathbf{x}, t), \tag{5.17}$$

called the *probability flow ODE*. The ODE is a CNF, and it can be used to generate samples using numerical ODE solvers starting with a sample from the base distribution. This CNF also uses the score function, which can then either be taken from the diffusion model in order to calculate its likelihood, or trained using the maximum likelihood objective of the CNF. The main result from Song's paper was the bounding of the likelihood of *score-based* models through their relationship with the probability flow ODE. Let $\hat{\rho}_{\mathrm{SDE}}$ be the SDE and ODE models' approximation of the data density respectively. The results are then as follows:

Theorem 5.2. *Let $\rho(\mathbf{x})$ be the unknown data distribution and $\tau(\mathbf{x})$ be a known base distribution. If $d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)dW$, then under certain regularity conditions*

$$D_{KL}(\rho || \hat{\rho}_{SDE}) \leq \mathcal{J}_{SM}(\theta; g(\cdot)^2) + D_{KL}(\rho_T | \pi), \tag{5.18}$$

where $\mathcal{J}_{SM}$ is the score-matching loss:

$$\mathcal{J}_{SM}(\theta;\zeta) = \frac{1}{2}\int_0^T \mathbb{E}_{\rho_t}\left[\zeta(t)||\nabla_{\mathbf{x}}\log\rho_t - \mathbf{s}_\theta(\mathbf{x},t))||_2^2\right] dt, \tag{5.19}$$

for a some positive weighting function ζ.

The regularity conditions required are given in Appendix C. This theorem shows that training with a weighted combination of score-matching losses similarly to Equation 4.10 rather than training based on the maximum likelihood leads to a bounding of the KL divergence, of the score-based density. Note that the function g used in the score matching loss is the time-dependent weighting of the stochastic term in the SDE; if we have $\frac{\mathrm{d}}{\mathrm{d}t}g = 0$, then the score matching loss reduces to the Fisher divergence. Therefore in the Euclidean overdamped setting, we have a bound on the KL divergence and therefore the likelihood of the SDE density estimate $\hat{\rho}_{\mathrm{SDE}}$.

5.3.1 Score-based flow on Riemannian manifolds

Song's derivation of the probability flow ODE comes from the Fokker-Planck equation for the SDE. In order to examine the probability flow for underdamped Langevin dynamics subject to constraints, we will examine the Markov generator of the constrained process. In order to sample from the uniform distribution $\mathcal{M}$, De Bortoli et al. used the overdamped Langevin dynamics with a force $\mathbf{F} = \mathbf{0}$ to noise the data, which constitutes a Brownian motion on the manifold [8]. Using a Brownian motion yields a probability-flow ODE of

$$\frac{\mathrm{d}}{\mathrm{d}t}\rho_t(\mathbf{q}) = \frac{1}{2}\nabla\cdot\rho_t(\mathbf{q}), \tag{5.20}$$

which is a generalization of the Euclidean Fokker-Planck equation for Brownian motion. In the case of underdamped Langevin dynamics, in order to converge to the uniform distribution we can still use a force of $\mathbf{F} = \mathbf{0}$, but the $\mathbf{q}$ dynamics are no longer a Brownian motion. Therefore the probability flow will take a different form, which we will discuss below.

5.3.2 Score-based flow for constrained underdamped Langevin dynamics with zero force

In order to discuss the probability flow of constrained Langevin dynamics, we first introduce some new concepts and notation for constrained systems. The Poisson bracket $\{\cdot,\cdot\}$ of two smooth functions $f_1,\ f_2$ acting on $(\mathbf{q},\mathbf{p})$ is defined as

$$\{f_1, f_2\} = (\nabla_{\mathbf{q}} f_1)^T \nabla_{\mathbf{p}} f_2 - (\nabla_{\mathbf{p}} f_1)^T \nabla_{\mathbf{q}} f_2 \tag{5.21}$$

Let $\{\cdot,\cdot\}_\Xi$ be the Poisson bracket for the constrained system, defined by

$$\{\rho_1, \rho_2\} = \nabla\rho_1^T J_G \nabla\rho_2, \tag{5.22}$$

where J_G is the constrained symplectic matrix given by

$$J_G = J - J\nabla\Xi\left[\nabla\Xi^T J\nabla\Xi\right]^{-1}\nabla\Xi^T J, \tag{5.23}$$

J is the symplectic matrix $J \in \mathbb{R}^{m\times m}$

$$J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}, \tag{5.24}$$

and $\Xi(\mathbf{q}, \mathbf{p}) \in \mathbb{R}^m$ is a vector of constraints. In our case

$$\Xi(\mathbf{q}, \mathbf{p}) = \begin{bmatrix} \mathbf{g}(\mathbf{q}) \\ \mathbf{G}(\mathbf{q})\mathbf{p} \end{bmatrix}. \tag{5.25}$$

Now we will make use of the following theorem which gives the Markov generator of the constrained underdamped Langevin dynamics:

Theorem 5.3. [19] *For general constraints Ξ, the Markov generator of the constrained process 3.12 where the matrix $\mathbf{G}(\mathbf{q})$ is invertible for all $\mathbf{q}$ can be written*

$$\mathcal{G}_\Xi = \mathcal{G}_\Xi^{ham} + \mathcal{G}_\Xi^{thm}, \tag{5.26}$$

where the Hamiltonian generator is

$$\mathcal{G}_\Xi^{ham} = \{-, H\}_\Xi, \tag{5.27}$$

and the fluctuation-dissipation part can be written

$$\mathcal{G}_\Xi^{thm} = \frac{1}{\beta}\nabla_{\mathbf{p}} \cdot (e^{-\beta H}\mathbf{\Pi}\nabla_{\mathbf{p}}). \tag{5.28}$$

This gives a probability flow of

$$\frac{\mathrm{d}}{\mathrm{d}t}\rho(\mathbf{q}, \mathbf{p}) = [\mathcal{G}_\Xi]\,(\mathbf{q}, \mathbf{p}). \tag{5.29}$$

However, these equations do not give a probability flow in the same way as Song et al. and De Bortoli et al., where the flow can be written in terms of the score function [29, 8].

5.4 Moser Flows

Moser flows on Riemannian manifolds have been proposed to improve upon some issues with Riemannian CNFs, being that they can be slow to train, and they also can be slow to converge [25]. The idea behind Moser flows is to construct an interpolant between the source and target distributions, i.e. a map $\alpha : [0, 1] \times \mathcal{M} \to \mathbb{R}_{>0}$ such that $\alpha_0 = \tau$, $\alpha_1 = \rho$ and $\int_{\mathcal{M}} \alpha_t dV = 1$ for all $t \in [0, 1]$. In this way the Moser flow will generate a CNF, but by limiting the *flow space*, we can express the learned probability in a desirable way as the divergence of a vector field. Then the flow of the CNF is defined by

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi(\mathbf{q}) = v_t(\Phi_t), \tag{5.30}$$

where ϕ is a map on the manifold $\Phi : \mathcal{M} \to \mathcal{M}$, v_t is a time dependent vector field, $v_t : \mathcal{M} \to T\mathcal{M}$, defined so that the flow satisfies the continuous normalization equation:

$$\Phi_t^* \alpha_0 = \alpha_t, \tag{5.31}$$

meaning that the way the pushforward map transforms densities agrees with the interpolant we defined on the density space. For $t = 1$, we get the map Φ_1 such that $\hat{\rho} = \Phi_1^*(\tau)$. Now the following theorem shows that if we construct a vector field $u_t \in T\mathcal{M}$ such that

$$\nabla \cdot u_t = -\frac{\mathrm{d}}{\mathrm{d}t}\alpha_t, \tag{5.32}$$

and define the vector field v_t as

$$v_t = \frac{u_t}{\alpha_t}, \tag{5.33}$$

then the diffeomorphism Φ defined by u_t and v_t satisfies the normalization equation as required.

Theorem 5.4. *The diffeomorphism $\Phi = \Phi_1$, defined by the ODE in 5.30 and the vector field v_t in equation 5.33 solves the normalization equation 5.31.*

This theorem is derived from works by Moser, after whom the Moser flow is named. Essentially, it tells us the vector field which defines the flow induced by an interpolation in the probability distribution space. As an example, the simplest choice of interpolant α_t is

$$\alpha_t = (1-t)\tau + t\rho, \tag{5.34}$$

which means that the divergence of the vector field u_t should be constant:

$$\nabla \cdot u = -\frac{\mathrm{d}}{\mathrm{d}t}\alpha_t = \tau - \rho. \tag{5.35}$$

In this case the vector field v_t is given by

$$v_t = \frac{u}{(1-t)\rho + t\pi}, \tag{5.36}$$

and the model density $\hat{\rho}$ is defined by

$$\hat{\rho} = \tau - \nabla \cdot u. \tag{5.37}$$

Rozen et al. give a description of how Moser flows can be used for generative modelling over Euclidean submanifolds [25]. Specifically, the following theorem speaks to the universality of Moser flows for Euclidean submanifolds. By parameterizing the vector field u as the projection onto $T_x\mathcal{M}$ of an ambient vector field $\bar{u}(\mathbf{q})$, we construct our network in the same way as in Section 6.2, and define

$$u(\mathbf{q}) = \mathbf{\Pi}\bar{u}(\mathbf{q}). \tag{5.38}$$

Then from Theorem 4.3, since $u(\mathbf{x}) \in T_{\mathbf{x}}\mathcal{M}$ for all $\mathbf{x} \in \mathcal{M}$, the Riemannian divergence is given by the Euclidean divergence for all $\mathbf{x} \in \mathcal{M}$. The following theorem validates this approach:

Theorem 5.5. [25] *Given an orientable, compact, boundaryless, connected, differentiable n-dimensional submanifold $\mathcal{M} \in \mathbb{R}^d$, with $n < d$, and a continuous target density $\rho : \mathcal{M} \to \mathbb{R}_{>0}$, there exists for each $\epsilon > 0$ a vector field u so that $\hat{\rho}$ defined by 5.37 satisfies:*

$$\max_{\mathbf{q}\in\mathcal{M}} |\rho(\mathbf{q}) - \hat{\rho}(\mathbf{q})| < \epsilon. \tag{5.39}$$

It's worth noting here that not all manifolds given by the set of constraints 2.11 are connected, so the usefulness of this theorem is slightly limited in our case. Therefore we will aim to compare Moser flows to SBD on compact, connected manfolds, but SBD can be employed on a wider range of manifolds, as is done in Section 6.2.

5.4.1 Training the model

The paper notes that $\hat{\rho}$ integrates to 1 over $\mathcal{M}$ by construction, since we know that ρ integrates to 1 over the manifold, and Stokes' theorem proves that the integral of $\nabla \cdot \mathbf{u}$ over $\mathcal{M}$ is zero. In order to ensure positivity over the manifold we choose a small value ϵ and then define

$$\hat{\rho}_+(\mathbf{q}) = \max\{\epsilon, \hat{\rho}(\mathbf{q})\} \tag{5.40}$$

$$\hat{\rho}_-(\mathbf{q}) = \min\{\epsilon, \hat{\rho}(\mathbf{q})\} \tag{5.41}$$

so that $\hat{\rho}_+ + \hat{\rho}_- = \hat{\rho}$. Here $\hat{\rho}_-$ is used to encourage $\hat{\rho} > \epsilon$, as the term is zero for all $\hat{\rho} \leq \epsilon$. Then the loss can be given by

$$\mathcal{L}(\hat{\rho}) = -\mathbf{E}_\rho \left[\log \hat{\rho}_+(\mathbf{q})\right] + \lambda \int_{\mathcal{M}} \hat{\rho}_- \mathrm{d}V, \tag{5.42}$$

where $\mathrm{d}V$ is the Riemannian volume form. The first term is the negative log-likelihood of the observations and can be approximated by

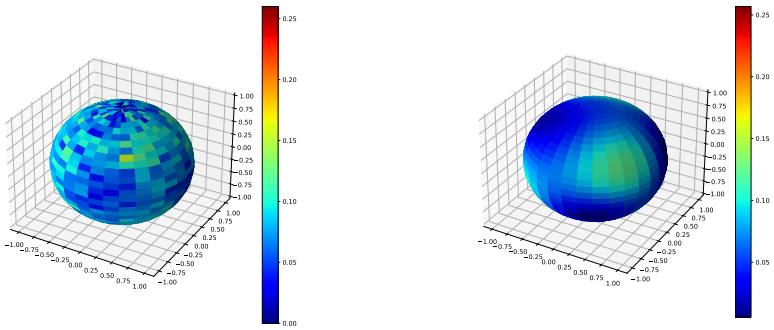
$$\mathbb{E}_\rho \log \hat{\rho}_+(\mathbf{x}) \tag{5.43}$$

and the integral in the second term is approximated by a Monte Carlo estimate over some distribution τ over $\mathcal{M}$:

$$\int_{\mathcal{M}} \hat{\rho}_- dV \approx \frac{1}{l} \sum_{i=1}^{l} \frac{\hat{\rho}_-(\mathbf{q}_i)}{\kappa(\mathbf{q}_i)}, \tag{5.44}$$

for some probability distribution κ on the manifold. In this way training seeks to maximize the expectation of the observations using the first term of Equation 5.42, while ensuring that the density is strictly positive using the second term.

It is noted in [25] that the additional term does not change the fact that the unique minimum of the loss is the true density ρ, for $\lambda > 1$ and sufficiently small ϵ, and since the loss used is the KL divergence, this implies that $\hat{\rho}$ is the maximum likelihood estimator within the Moser flow framework. Figure 10 demonstrates a Moser flow model on a low-dimensional compact and connected manifold. The neural network, shown in Figure 11, used to parameterise the vector field u was constructed similarly to the network used in Section 6.2.



(a) Data samples used to train the Moser flow model

(b) Moser flow model fit to the data

Figure 10: Moser flow model trained on data distributed from the distribution function $e^{-U(x,y,z)}$ with $U(x, y, z) = (x^2 - 1)^2 + (y^2 - 1)^2 + (z^2 - 1)$

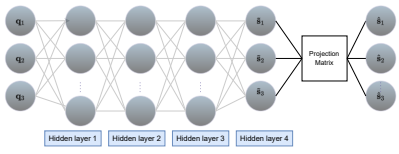


Figure 11: Neural network architecture used for the Moser flow model

5.5 Comparison

To compare the SBGM with a Moser flow model, both models can be parameterized using of a vector field projected onto the tangent space of the constraint manifold. In the case of the Moser flow model, this divergence itself effectively represents the probability distribution, by $\nabla \cdot u = \tau - \rho$. This results in the divergence being effectively minimized in the case of a uniform prior, under the condition that the density remains positive, for the data points. The SBGM also effectively minimizes the divergence of the vector field, although the norm of the vector field is added so the divergence is not quite minimized. However despite the relationship between the parameterisations and training, in theory these models are approaching the problem in different ways. While both can be seen as minimizing the divergence of the vector field on the data points, the Moser flow model uses this divergence directly to assign a probability to a specific point. The SBGM on the other hand is concerned with the vector field itself, rather than the divergence, and uses it to model the score function of the marginal density in the diffusion process. As noted by De Bortoli et al., both SBGMs and Moser flow models interpolate between the base density π and the data density ρ [8]. The key difference is that while Moser flows interpolate on the density space by defining a map $MF(\rho, \pi)$ sending two density functions to a new density function, the SBGM instead interpolates by defining a path on $\mathcal{M} \times \mathcal{M} \to \mathcal{M}$. This path then generates an interpolation in the sample space, rather than the density space. Moser flows may be very effective in low-dimensional spaces, however in high dimensional spaces, the number of samples needed for the Monte Carlo estimator, which ensures positivity, grows rapidly, with De Bortoli et al. unable to train a model with dimension greater than 10 [8]. In addition, Moser flows are restricted to a narrow class of Riemannian manifolds as described in Section 5.4.

6 Experiments

Human pose estimation is the active research area in computer vision concerned with predicting the joint locations of a human body, usually based on $2D$ images. There are a wide range of applications for $3D$ pose estimation including autonomous driving, human-computer interaction and biomechanics [34].

6.1 3D Human Poses Estimation

This experiment aims to estimate the $3D$ coordinates of human skeleton joints based on a $2D$ image captured by a camera with known coordinates and focal lengths. The constraint manifold is defined to ensure that the lengths of the limbs remain constant:

$$||q^i - q^j||_2^2 = l_{i,j}^2 \ \forall (i,j) \in \mathcal{J}, \tag{6.1}$$

where q^i is the $3D$ position of joint i, $l_{i,j}$ is the length of the limb, and $\mathcal{J}$ is the set of limbs. Assuming that we are given noisy $2D$ locations x^i of the joints and the matrix A of camera parameters, Brubaker et al. use a linear pose model based on eigenposes obtained from the training data [28, 6]. Essentially the model fit to the distribution is a normal distribution on the principal component analysis (PCA) of the data. The eigenposes are obtained from the PCA as the columns of the matrix U in the singular value decomposition (SVD) of the data matrix X, where each column of X is a pose q in the training data, which has been de-meaned, meaning that the mean of each joint coordinate is subtracted from the data before the SVD is applied.

The eigenposes capture the most dominant directions of deviation from the mean of the poses. The density is then defined as

$$\begin{aligned} \pi(\mathbf{q}|\mathbf{x}) &\propto p(\mathbf{x}|\mathbf{q}) \cdot p(\mathbf{q}) \\ &\propto \prod_{i=1}^{N} \exp(-||x(q^i) - x^i||^2/\sigma_m^2) \prod_{i=1}^{3N} \exp(-(P_j^T(q - q_0))^2/\sigma_j^2), \end{aligned} \tag{6.2}$$

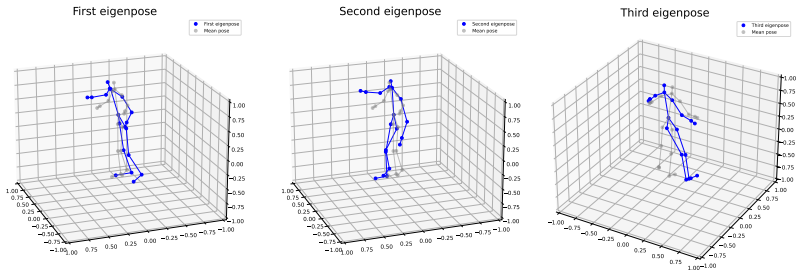


Figure 12: First, second and third eigenpose.

where $x(q^i)$ is the $2D$ projection of joint i, P_j is the jth eigenpose and σ_j^2 is the associated eigenvalue. The first term in the density function describes the probability of a $2D$ observation $\mathbf{x}$ given a $3D$ pose $\mathbf{q}$. Since we have added Gaussian noise to our observations this is a simple Gaussian distribution. The second term describes the prior distribution that was fit by Brubaker et al. to the HumanEva data [6]. Consider the matrix X with columns given by the data poses, $X = \begin{bmatrix}\mathbf{q}_1, \mathbf{q}_2, \dots \mathbf{q}_m\end{bmatrix}$. Let $\hat{X}$ be the de-meaned observations, so that $\hat{X} + \mu \otimes \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix} = X$ where μ is the vector of the row-wise means. Then the PCA is applied using an SVD of $\hat{X}$, $\hat{X} = U\Sigma V^T$. The transformed data is then modelled using a multivariate Gaussian distribution with zero covariance (since the SVD removes any correlation, zero covariance is a reasonable assumption, since the relationship between joint positions on the skeleton is naturally linear). The data used to train and test the model was generated according to the variational auto-encoder (VAE) used by Graham et al [9]. It was trained using the *PosePrior* motion-capture dataset, and generates angles according to a log-normal model that was fit to the data. For the purposes of this paper, the generated lengths were kept constant (at the mean of the model fit to the data), although there is some variation in the *perceived* bone lengths since the skeleton we use does not include all of the joints used to generate the data. The data was centred with the pelvis at $(0, 0, 2)$ so that the manifold was closed.

A known camera matrix was used to project the $3D$ data to a $2D$ image. The $2D$ image was then perturbed randomly by adding a random variable with a standard deviation of 0.1 to each joint's coordinates. The integrator was initialized at a random point in the training dataset for each frame of the test dataset, and run for $2,000$ steps with a stepsize of 0.01.

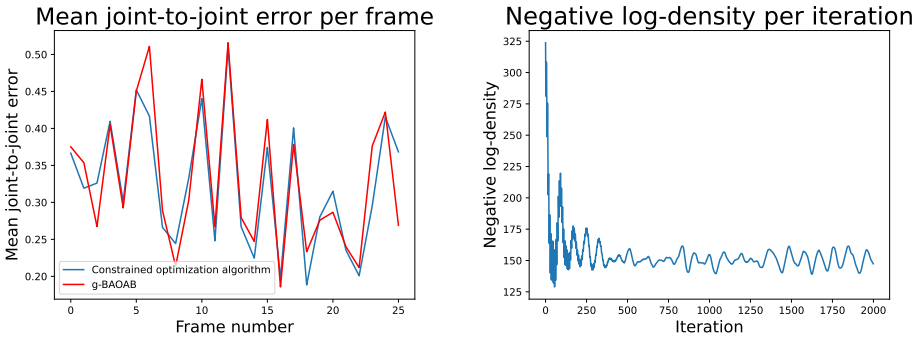


Figure 13: Mean joint-to-joint error per frame and negative log density per iteration

The g-BAOAB estimates of the $3D$ poses were compared to estimates obtained from a gradient-based constrained optimization algorithm which minimized the negative log-density, as was done by De Bortoli et al. [8]. The error was measured as the average Euclidean distance between the joints. The algorithm used was sequential least-squares, a method similar to Newton's method but which can minimize nonlinear functions subject to constraints. The g-BAOAB estimate was taken to be the *intrinsic* mean of the positions generated by the algorithm, since the Euclidean mean will not necessarily lie on the manifold - this was obtained by converting to intrinsic coordinates, taking the average, and converting the average back to

Euclidean coordinates. As shown in Figure 13, the g-BAOAB estimate was roughly as effective as the sequential least-squares estimate, underperforming the optimization algorithm for some frames but outperforming it for others.

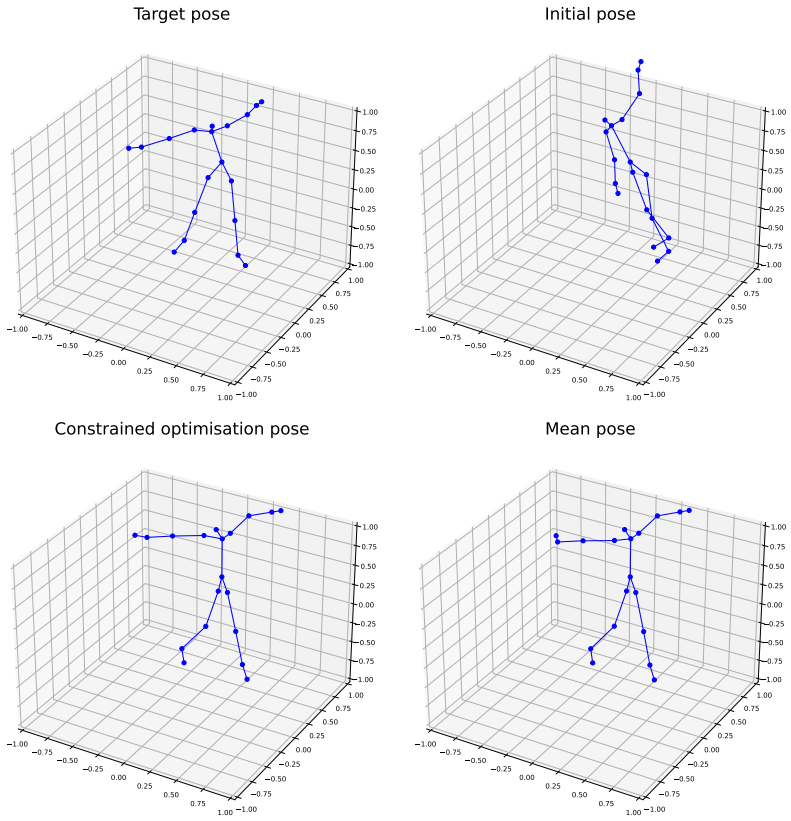


Figure 14: Target pose, constrained optimization algorithm pose and g-BAOAB mean pose after 2,000 g-BAOAB iterations.

6.2 3D Human Poses Generation

Next, we aim to use the SBD method described in Section 4 to noisify $3D$ poses from our dataset in order to train a score-based model to de-noisify poses, in order to sample from the pose distribution.

6.2.1 Adding noise

We add noise to the data with constrained underdamped Langevin dynamics using the g-BAOAB algorithm, as described in Section 4.2. We fix one joint of the poses so that our constraint manifold is closed. This lets us perturb the data using the g-BAOAB algorithm with a force $\mathbf{F} = \mathbf{0}$, which will give a uniform stationary distribution. We can sample from the uniform distribution on the manifold by sampling the angle of each bone from a uniform spherical distribution, which will make every point on the manifold equally likely:

$$\tau(\mathbf{q}) = \mathbb{P}(\mathbf{q_1} = \mathbf{q}_1^*)\mathbb{P}(\mathbf{q_2} = \mathbf{q}_2^* | \mathbf{q}_1^*) \cdots \mathbb{P}(\mathbf{q_n} = \mathbf{q}_n^* | \mathbf{q}_1^*, \mathbf{q}_2^* \dots \mathbf{q}_{n-1}^*) \tag{6.3}$$

$$\implies \tau(\mathbf{q}) = \prod_{i=2}^{n} \frac{1}{4\pi l_i^2} \tag{6.4}$$

$$\tag{6.5}$$

where i indexes the bones and l_i is the length of bone i. Note that since we have fixed the initial point, $\mathbb{P}(\mathbf{q}_1 = \mathbf{q}_1^*) = \mathbf{0}$. As a note on the uniform spherical distribution, it is incorrect to

uniformly sample $\phi \in [0, 2\pi]$ and $\theta \in [0, \pi]$. Due to the fact that the area element is given by $\mathrm{d}\Omega = \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi$, this method would not give a uniform distribution and points would instead more likely be sampled nearer to the poles. Points are instead generated by sampling $\phi \in [0, 2\pi)$ and $\theta = \cos^{-1}(2v - 1)$, where v is uniformly distributed on $[0, 1)$.

The data were perturbed using the g-BAOAB algorithm, with $h = 0.01$, $\gamma = \beta = 1$, and $T = 10$ giving 1000 steps. Figure 15 shows how perturbing using the g-BAOAB algorithm changes the pose. As the time approaches infinity, this will approach the distribution defined above.

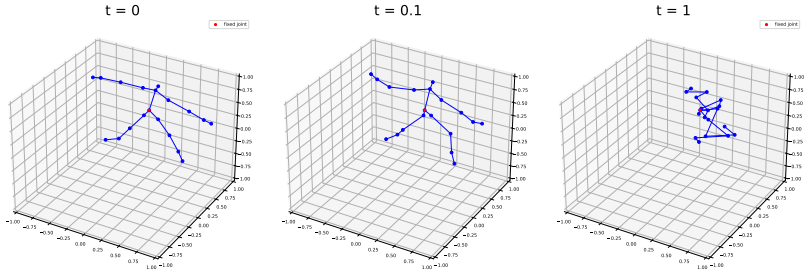


Figure 15: Initial pose being perturbed using the g-BAOAB algorithm, with a step size of 0.01.

6.2.2 Score Matching

Similarly to De Bortoli et al's experiments in Riemannian SBGM [8], a 5-layer multilayer perceptron was used to parameterize the score based network. The time was encoded in a manner similar to Song's method using Gaussian projections [31], which have been used to enable MLPs to learn high-frequency functions [32].

As shown in Section 4.1.1, we can train the network by minimizing the loss function:

$$\mathcal{L}(\hat{s}_\theta) = \int_{\mathcal{M}} \frac{1}{2} ||\hat{s}_\theta(\mathbf{q}, t)||^2 + \Delta \hat{s}_\theta(\mathbf{q}, t) d\mathbf{q} + C. \tag{6.6}$$

We approximate the minimizer of this loss by

$$\mathcal{L}(\hat{s}_\theta) \simeq \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{2} ||\hat{s}_\theta(\mathbf{q}, t)||_2^2 + \Delta \hat{s}_\theta(\mathbf{q}, t) \right). \tag{6.7}$$

where x_t are the data from the distribution. In this case, x_t are the noisified data points.

The neural network is constructed as shown by De Bortoli et al., using the projection onto the tangent space [8]:

$$\hat{s}_\theta(\mathbf{q}, t) = \mathbf{\Pi}(\mathbf{q}) \tilde{\mathbf{s}}_\theta(\mathbf{q}, t)) \tag{6.8}$$

where $\tilde{s} : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ is an ambient vector field. Then using Theorem 4.3, the Riemannian divergence $\nabla \cdot \hat{s}_\theta(\mathbf{q}, t)$ is equal to the Euclidean divergence $\nabla \cdot_E \hat{s}_\theta$. This allows us to calculate the loss without knowledge of the Riemannian metric, which is useful for constraints of the form given in Equation 3.12c.

By an application of Stokes' lemma, if our neural network is remaining tangent to the manifold at each point and the divergence is calculated accurately, we should have that

$$\int_{\mathcal{M}} \Delta \hat{\mathbf{s}}(\mathbf{q}) d\mathbf{q} = 0. \tag{6.9}$$

In order to check that this is satisfied, we use Monte Carlo integration using a uniform distri-

bution over the manifold. Then we should have that

$$\sum_{\mathbf{q}\in\mathcal{M}} \Delta\hat{\mathbf{s}}(\mathbf{q}) \approx 0 \tag{6.10}$$

$$\implies \frac{1}{N}\sum_{i=1}^{N} \Delta\hat{\mathbf{s}}(\mathbf{q}_i) \approx 0 \tag{6.11}$$

$$\implies \frac{1}{N}\sum_{i=1}^{N} (1 - \Delta\hat{\mathbf{s}}(\mathbf{q}_i)) \approx 1, \tag{6.12}$$

so we can validate that the neural network is behaving as expected using this final value. Indeed, as shown in Figure 16, the approximation approaches 1 as the number of samples is increased.

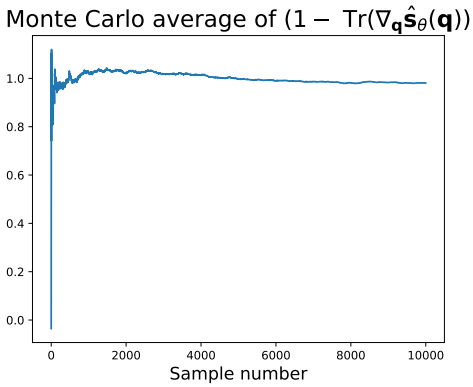


Figure 16: Monte Carlo average of $(1 - \nabla\hat{\mathbf{s}})$

The specific construction of the neural network is shown in Figure 17. The main structure of the Multi-Layer Perceptron (MLP) was similar to that used by De Bortoli et al., while the structure of the time embeddings was taken from Song et al. [8, 31]. The time was mapped to 58 different nodes, each taking a value of either $\sin(2\pi\omega t)$ or $\cos(2\pi\omega t)$ for a random variable ω for each node which was fixed at initialisation. The arrows from the time embedding indicate element-wise addition (after the application of the activation function). Do to computational constraints, the neural network was trained on $1,700$ data points from the data distribution, with 100 points serving as a test dataset. 4 spatial hidden layers were used rather than 5 in order to both speed up training and reduce the risk of overfitting due to having less computational resources and so a lower number of samples.

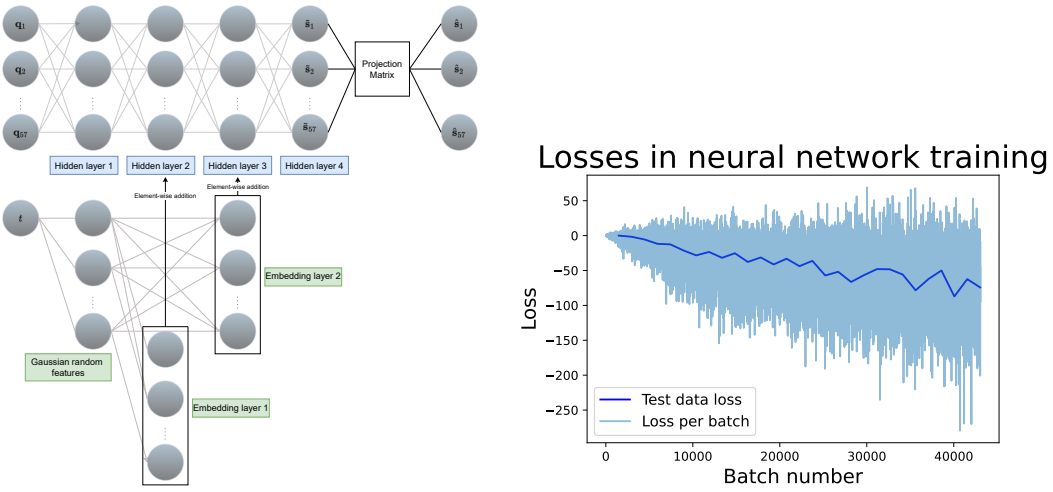


Figure 17: Score-based diffusion neural network architecture and training loss progression

The model was trained on 1,700 initial data points, although each datapoint was noised randomly, so in effect the model was trained on many more points than this. The learning rate used was 10^{-4}. In each epoch, for each sample from the data distribution, a time t was randomly chosen, the data was noised from 0 to t, and the loss was updated based on the score function given at time t. The full experiment algorithms are given below, using a step-size of h and a time of T:

Algorithm 3 Score-based training using g-BAOAB

for $n \in \{0, \dots, N_{\text{epochs}} - 1\}$ **do**
  for $\mathbf{q}_{\text{batch}} \in$ dataset **do** ▷ A batch is selected from the shuffled data
    for $\mathbf{q} \in \mathbf{q}_{\text{batch}}$ **do**
      $\mathcal{L}_{\text{batch}} = 0$
      $t \sim \mathcal{U}[1, \lfloor \frac{T}{h} \rfloor]$ ▷ ts is sampled uniformly from integers between 1 and $\lfloor \frac{T}{h} \rfloor$
      $\mathbf{p} = \mathbf{0}$ ▷ Velocities are initialized at zero
      for $i \in \{0, t-1\}$ **do**
        $\mathbf{q}, \mathbf{p} = \text{gBAOAB}(\mathbf{q}, \mathbf{p}, \text{Force} = \mathbf{0}, \text{step} = h)$ ▷ Data is noised using g-BAOAB
      end for
      $\mathcal{L}(\hat{s}_\theta) = \frac{1}{2}||\hat{s}_\theta(\mathbf{q}, t)||^2 + \nabla \cdot \hat{s}_\theta(\mathbf{q}, t)$
      $\mathcal{L}_{\text{batch}} = \mathcal{L}_{\text{batch}} + \mathcal{L}(\hat{s}_\theta)$ ▷ The loss is calculated and added to the batch loss
    end for
    $\theta = \text{optimizer.update}(\theta)$ ▷ The neural network parameters are updated
  end for
end for

Algorithm 4 Score-based diffusion time-reversal using g-BAOAB

$\mathbf{q} \sim \pi$ ▷ $\mathbf{q}$ is sampled from the base (uniform) distribution
$\mathbf{p} = \mathbf{0}$
for $t \in \{1, \dots (\lfloor \frac{T}{h} \rfloor)\}$ **do**
  $\mathbf{q}, \mathbf{p} = \text{reverse-gBAOAB}(\mathbf{q}, \mathbf{p}, \text{Force} = \hat{s}_\theta(\mathbf{q}, T - ht), \text{step} = h)$ ▷ Data is denoised using g-BAOAB
end for

6.3 Results

The results of the SBD sampling process using g-BAOAB are presented here, compared with samples from the true data distribution.

6.4 Discussion

The samples which were generated by the SBGM are shown in Figure 19. We can see that the noising process has been reversed; it appears that the samples obtained are close to the data distribution, relative to the uniform distribution which was used as the base distribution. However there are certain noticeable defects in the samples. This is likely due to issues in training the neural network rather than with the time-reversal process. These issues are discussed in Appendix A.

7 Related work

7.1 Stochastic interpolants

Stochastic intepolants have been proposed as a framework that can unify flows and diffusions. Similarly to SDB, they involve interpolation on the event space [2, 1]. Given two densities ρ_0

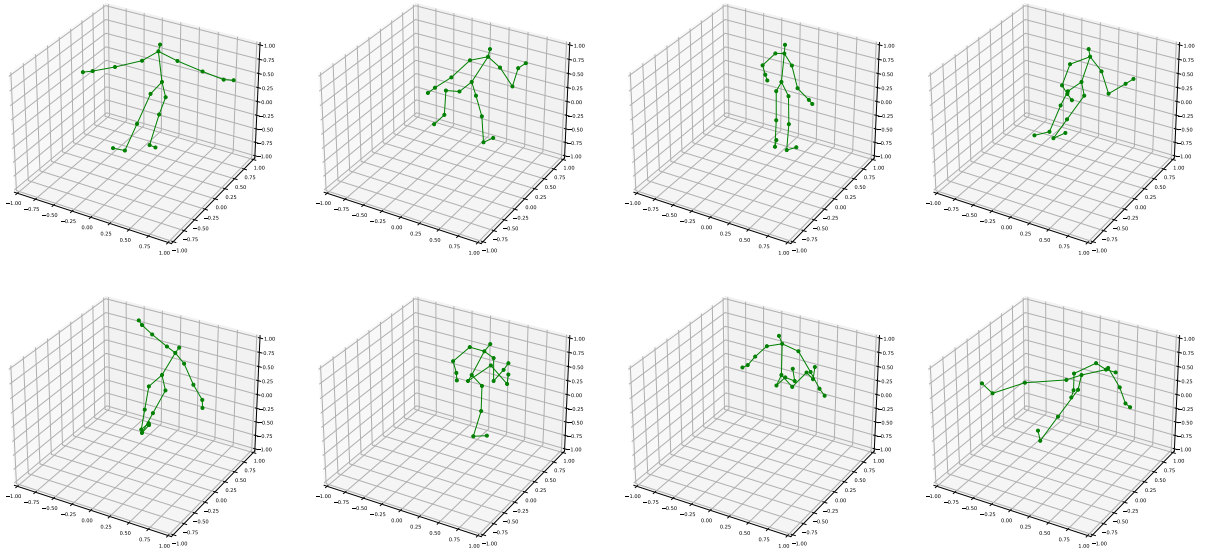


Figure 18: Samples from the data distribution

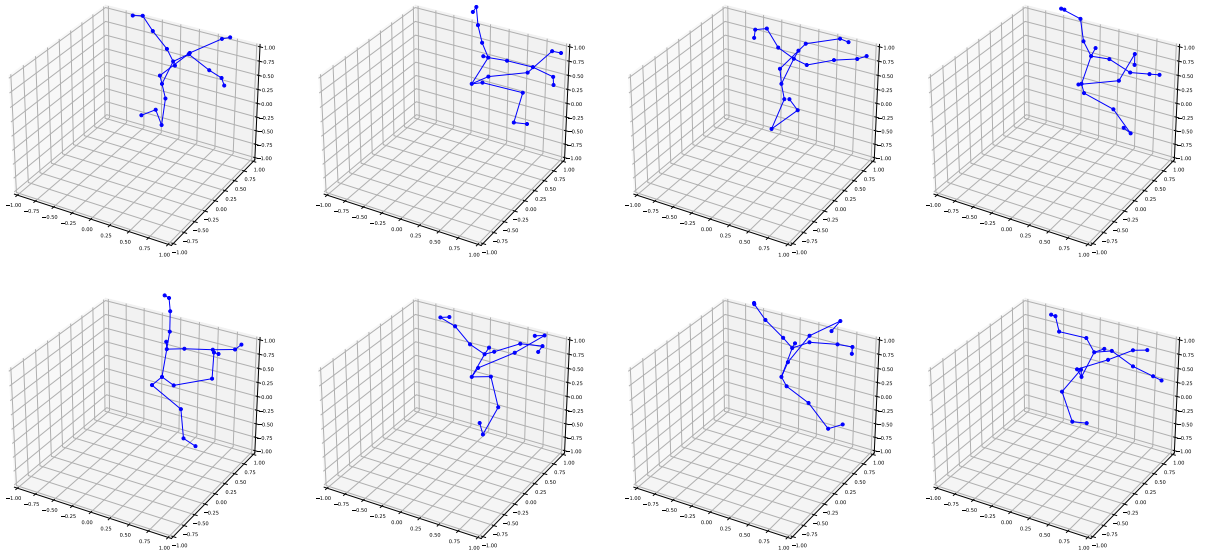


Figure 19: Samples from a score-based diffusion model using g-BAOAB

and $\rho_1 : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, a stochastic interpolant is a stochastic process

$$\mathbf{x}_t = I(t, \mathbf{x}_0, \mathbf{x}_1), \tag{7.1}$$

for $t \in [0,1]$, where $I(0, \mathbf{x}_0, \mathbf{x}_1) = \mathbf{x}_0, I(1, \mathbf{x}_0, \mathbf{x}_1) = \mathbf{x}_1$, $\mathbf{x}_0 \sim \rho_0$, $\mathbf{x}_1 \sim \rho_1$ and where I is twice continuously differentiable in t and $\mathbf{x}$, and satisfies the conditions $I(0, \mathbf{x}_0, \mathbf{x}_1) = \mathbf{x}_0$ and $I(1, \mathbf{x}_0, \mathbf{x}_1) = \mathbf{x}_1$, as well as

$$\exists C_1 < \infty : \ |\delta_t I(t, \mathbf{x}_0, \mathbf{x}_1)| \leq C_1 |\mathbf{x}_0 - \mathbf{x}_1| \ \forall (t, \mathbf{x}_0, \mathbf{x}_1) \in [0,1] \times \mathbb{R}^n \times \mathbb{R}^n. \tag{7.2}$$

The first condition on I is simply that it does what we would expect of an interpolant, while the second requires that its output doesn't move too quickly relative to its input.

The velocity is then defined as the vector field $v : \mathbb{R}^n \times [0,1] \to \mathbb{R}^n$ such that:

$$\dot{\mathbf{x}}_t = v_t(\mathbf{x_t}). \tag{7.3}$$

The stochastic interpolant $\mathbf{x}_t$ defined above with $I(t, \mathbf{x}_0, \mathbf{x}_1)$ satisfying the above conditions has a probability density that satisfies the continuity equations

$$\partial_t \rho_t(\mathbf{x}) + \nabla \cdot v_t(\mathbf{x}) \rho_t(\mathbf{x}) = 0, \tag{7.4}$$

which maps from ρ_0 at time $t = 0$ to ρ_1 at time $t = 1$, with its velocity given by the unique minimizer of

$$\mathcal{L}(\hat{v}) = \mathbb{E}\left[|\hat{v}(I(tx_0, x_1), t)|^2 - 2\partial_t I_t(\mathbf{x}_0, \mathbf{x}_1) \cdot \hat{v}_t(I_t(\mathbf{x}_0, \mathbf{x}_1))\right]. \tag{7.5}$$

Thus the velocity could be parameterised as a neural network and learned, as was done with Moser flows and SBGM. However, rather than maximizing the likelihood of this model, the stochastic interpolant approach constructs the map I based on a prior assumption. Then once we have a time-differentiable map I we can estimate the velocity field v, which gives us an ODE for the probability density.

In the more general case proposed later [2], the interpolant is generalized to include a *latent variable* and we have

$$\mathbf{x}_t = I(t, \mathbf{x}_0, \mathbf{x}_1) + \gamma(t)\mathbf{z}, \tag{7.6}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\gamma : [0, 1] \to \mathbb{R}$ is continuously differentiable. The addition of the latent variable smooths the density and velocity, and also allows stochastic interpolants to be more closely related to SBD. The latent variable allows the score of the marginal distribution to be written as

$$\nabla \mathbf{x} \log \rho_t(\mathbf{x}) = -\gamma(t)^{-1} \mathbb{E}(\mathbf{z} | \mathbf{x}_t = \mathbf{x}) \tag{7.7}$$

and the score associated with the model is the unique continuously differentiable minimizer of

$$\mathcal{L}(\hat{s}_\theta) = \int_0^1 \mathbb{E}\left(\frac{1}{2}|\hat{s}_\theta(t, \mathbf{x}_t)|^2 + \gamma^{-1}(t)\hat{s}_\theta(t, \mathbf{x_t})\right) \mathrm{d}t, \tag{7.8}$$

In particular, it has been shown that minimizing the loss functions given in Equations 7.8 and 7.5 maximizes the likelihood of a stochastic generative model with force given by a weighted sum of the score and velocity estimates

$$\mathbf{F}(\mathbf{x}, t) = \hat{v}_t(\mathbf{x}) + \kappa \hat{s}_\theta(\mathbf{x}, t) \tag{7.9}$$

for a constant κ. This links the stochastic interpolant framework with SBGM.

7.2 Pushforward of Euclidean normalizing flows

In this paper I focused on methods for constructing normalizing flows directly on manifolds in order to compare with constrained SBD, however another way to construct a normalizing flow on a $d-$dimensional manifold which is embedded in $\mathbb{R}^n$ is to use the embedding map $T : \mathbb{R}^d \to \mathbb{R}^n$. Then the embedding map induces a metric G on the tangent space of $\mathcal{M}$, which gives a volume $\mathrm{d}\nu = \sqrt{\det\, G(\mathbf{u})}\mathrm{d}\mathbf{u}$ on $\mathcal{M}$. Then the inverse mapping of $T|_{\mathcal{M}}$ as $T^{-1} : \mathcal{M} \to \mathbb{R}^n$, we arrive at the density on the manifold [23]:

$$\rho(\mathbf{q}) = \tau(T^{-1}(\mathbf{q}))\left[\det\, G(T^{-1}(\mathbf{q}))\right]^{-1/2}, \tag{7.10}$$

where τ is our base density in Euclidean space. This is a generalization of Equation 5.10 in the case where $\mathcal{M} = \mathbb{R}^n$. However, the diffeomorphism T above can only exist if the manifold $\mathcal{M}$ is topologically equivalent to Euclidean space, and without this map any embedding map will create singularities, which lead to numerical issues.

8 Limitations

As discussed in Section 5.3, the main limitation of using constrained overdamped Langevin dynamics in SBD model is the lack of an accompanying probability-flow model when compared to overdamped Langevin dynamics. While SBGMs can often outperform likelihood-based models, as pointed out by Song et al., there are a number of fields such as lossless compression in which likelihood is extremely important [29].

9 Conclusion

In summary, SBD is a powerful framework for generative modelling, in which data is noised using an SDE and denoised using the reverse SDE to return to the data distribtion. This paper first introduces constrained Langevin dynamics, a formulation of standard underdamped Langevin dynamics which is subject to holonomic algebraic constraints. This system of constraints allows the constraint surface to be viewed as a Riemannian manifold. Many problems in data science involve data which is supported on Riemannian manifolds as opposed to Euclidean space, so SBD on manifolds is an active area of research.

We then introduced a method for SBD modeling on constraint manifold, which is based on the g-BAOAB integrator for constrained Langevin dynamics. While previous SBD models have focused on overdamped Langevin dynamics, this method uses underdamped dynamics, which has been seen to converge faster than overdamped dynamics to the invariant distribution [7]. The g-BAOAB integrator, which is based on a BAOAB splitting of the Langevin dynamics with a geodesic step, is used as a constrained integrator, in order to noise and denoise the data. In order to perform the denoising step, the SDE was reversed, and the reversed SDE was integrated using the g-BAOAB algorithm.

Considering the constraint manifold as an embedded submanifold in Euclidean space, we demonstrate that the score model can be parameterized by a neural network and projected onto the tangent space of the constriaint manifold, allowing the divergence to be computed for a general system of holonomic constraints. This approach can be compared with a Moser flow model, which also parameterizes and projects a neural network onto the tangent space of a Euclidean submanifold. Moser flows are well suited to low-dimensional, connected, boundaryless manifolds, but the score based diffusion method we propose is more general.

CNFs play an important role in Euclidean score-based generative models, allowing the likelihood to be computed and maximized. We show that in the case of underdamped constrained Langevin dynamics, in contrast to constrained and unconstrained overdamped Langevin dynamics, a probability-flow ODE is not available, which is an important limitation of the method.

The algorithms were demonstrated on a $3D$ pose generation problem, which is an active area of research in computer vision, and successfully generated samples using the diffusion process. The results of the diffusion process could be improved through further work studying the properties of SBGM using constrained Langevin dynamics, and also through improved efficiency in implementation of the algorithms.

Appendices

A Implementation

The g-BAOAB algorithms were implemented using PyTorch and used automatic differentiation to compute the Jacobian matrices required for the projections onto the tangent space used in the g-BAOAB algorithm and in parameterizing the vector fields as part of the neural network. The implementation saw several bottlenecks, especially in computing the Jacobian matrices, and the experiments in Section 6 were limited in their computation by a lack of effieciency. Notably, the

neural network losses were still decreasing on the test dataset when the experiment was halted, due to slow training times.

There are several computational efficiencies that could be implemented in the future to make the method more feasable. For example, sliced score matching would allow the neural network to be trained more effieciently. The noising process also slowed the training substantially, and in the future many of the operations could be vectorized in order to be computed as efficiently as possible [30].

The code used to run the experiments is available here.

B Activation functions

Throughout the paper various vector fields are constructed using neural networks. In Section 5.4, the neural networks are required to be $l-$finite, since for a smooth target function it has been proven that there exists an MLP with $L-$finite smooth activation function that uniformly approximates the first m derivatives of f with an error of at most ϵ for a given ϵ. Therefore the activations chosen are **Tanh** and **Softplus**, which are $l-$finite for $l \geq 1$ and $l \geq 2$, respectively. A function is said to be $l-$finite if it is l times continuously differentiable, and satisfies $\int_{-\infty}^{\infty} |f^{(l)}| < \infty$. **Swish**, **ReLU** and **LeakyReLU** are not $l-$finte for any l. **ReLU** has been widely shown to be a stable activation function which helps to mitigate issues with exploding or vanishing gradients, however for this reason these functions were not used in this context.

C Assumptions

C.1 Theorem 4.2

The regularity conditions imposed on $\rho_{t|s}(\mathbf{x}_t|\mathbf{x}_s)$ are simply that

1. $\rho_{t|s}(\mathbf{q}_t|\mathbf{q}_s)\mathbf{s}_t(\mathbf{q})$ is a vector field in C_1 for all $\mathbf{x}_s$.
2. $|\rho_{t|s}(\mathbf{q}_t|\mathbf{q}_s)| \in L_1 \ \forall \mathbf{q}_s \in \mathcal{M}$.
3. $\nabla \cdot \left[\rho_{t|s}(\mathbf{q}_t|\mathbf{q}_s)\mathbf{s}_t(\mathbf{q}_t)\right] \in L_1 \ \forall \mathbf{q}_s \in \mathcal{M}$.

where C_1 is the set of all continuously differentiable vector fields, and $f \in L_1 \iff \int_{\mathcal{M}} |f| < \infty$.

C.2 Theorem 5.2

The assumptions made by Song et al. required for either Theorem 5.2 are given below.

1. $\rho(\mathbf{x}), \tau(\mathbf{x}) \in \mathcal{C}^2$ and $\mathbb{E}_\rho\left[||\mathbf{x}||_2^2\right], \mathbb{E}_\tau\left[||\mathbf{x}||_2^2\right] < \infty$.
2. $\mathbf{f}(\cdot, t) \in C_1$ for all $t \in [0, T]$, and there exists a constant $C > 0$ such that for all $\mathbf{x} \in \mathbb{R}^n$ and each $t \in [0, T]$, $||\mathbf{f}(\mathbf{x}, t)||_2 \leq C(1 + ||\mathbf{x}||_2)$.
3. There exists a constant $C > 0$ such that for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $||\mathbf{f}(\mathbf{x}, t) - \mathbf{f}(\mathbf{y}, t)||_2 \leq C||\mathbf{x} - \mathbf{y}||_2$.

References

[1] M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2023.

[2] M. S. Albergo and E. Vanden-Eijnden. Building normalizing flows with stochastic interpolants, 2023.

[3] H. C. Andersen. Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations. *Journal of Computational Physics*, 52(1):24–34, 1983.

[4] B. D. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[5] D. Berthelot, T. Schumm, and L. Metz. BEGAN: boundary equilibrium generative adversarial networks. *CoRR*, abs/1703.10717, 2017.

[6] M. A. Brubaker, M. Salzmann, and R. Urtasun. In N. D. Lawrence and M. A. Girolami, editors, *AISTATS*, volume 22 of *JMLR Proceedings*, pages 161–172. JMLR.org, 2012.

[7] X. Cheng, N. S. Chatterji, P. L. Bartlett, and M. I. Jordan. Underdamped langevin mcmc: A non-asymptotic analysis. In S. Bubeck, V. Perchet, and P. Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 300–323. PMLR, 06–09 Jul 2018.

[8] V. De Bortoli, E. Mathieu, M. Hutchinson, J. Thornton, Y. W. Teh, and A. Doucet. Riemannian score-based generative modelling. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 2406–2422. Curran Associates, Inc., 2022.

[9] M. M. Graham and A. J. Storkey. Asymptotically exact inference in differentiable generative models. *Electronic Journal of Statistics*, 11(2):5105 – 5164, 2017.

[10] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[11] A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

[12] B. Leimkuhler and C. Matthews. *The Langevin Equation: With Applications to Stochastic Problems in Physics, Chemistry and Electrical Engineering*. World Scientific Series in Contemporary Chemical Physics. World Scientific, September 2012.

[13] B. Leimkuhler and C. Matthews. Robust and efficient configurational molecular sampling via Langevin dynamics. *The Journal of Chemical Physics*, 138(17):174102, 05 2013.

[14] B. Leimkuhler and C. Matthews. *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Interdisciplinary Applied Mathematics. Springer, May 2015.

[15] B. Leimkuhler and C. Matthews. Efficient molecular dynamics using geodesic integration and solvent–solute splitting. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 472(2189):20160138, 2016.

[16] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, September 2004.

[17] B. Leimkuhler, T. Vlaar, T. Pouchon, and A. Storkey. Better training using weight-constrained stochastic dynamics, 2021.

[18] B. J. Leimkuhler and R. D. Skeel. Symplectic numerical integrators in constrained hamiltonian systems. *Journal of Computational Physics*, 112(1):117–125, 1994.

[19] T. Lelievre, M. Rousset, and G. Stoltz. *Free Energy Computations: A Mathematial Perspective*, volume 31 of *Springer Series in Computational Mathematics*. Imperial College Press, second edition, 2010.

[20] T. Lelievre, M. Rousset, and G. Stoltz. Langevin dynamics with constraints and computation of free energy differences. *Mathematics of computation*, 81(280):2071–2125, 2012.

[21] L. Liao, Y. Zhang, S. J. Maybank, Z. Liu, and X. Liu. Image recognition via two-dimensional random projection and nearest constrained subspace. *Journal of Visual Communication and Image Representation*, 25(5):1187–1198, 2014.

[22] E. Mathieu and M. Nickel. Riemannian continuous normalizing flows. *Advances in Neural Information Processing Systems*, 33:2503–2515, 2020.

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

[24] S. Pascual, G. Bhattacharya, C. Yeh, J. Pons, and J. Serrà. Full-band general audio synthesis with score-based diffusion. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[25] N. Rozen, A. Grover, M. Nickel, and Y. Lipman. Moser flow: Divergence-based generative modeling on manifolds. *Advances in Neural Information Processing Systems*, 34:17669–17680, 2021.

[26] J.-P. Ryckaert, G. Ciccotti, and H. J. Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics*, 23(3):327–341, 1977.

[27] M. Shapovalov and R. Dunbrack. A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. *Structure*, 19(6):844–858, 2011.

[28] J. D. Skufca, E. M. Bollt, R. Pilkar, and C. J. Robinson. Eigenposes: Using principal components to describe body configuration for analysis of postural control dynamics. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–3, 2010.

[29] Y. Song, C. Durkan, I. Murray, and S. Ermon. Maximum likelihood training of score-based diffusion models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1415–1428. Curran Associates, Inc., 2021.

[30] Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

[31] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *CoRR*, abs/2011.13456, 2020.

[32] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *CoRR*, abs/2006.10739, 2020.

[33] A. Vahdat and J. Kautz. NVAE: A deep hierarchical variational autoencoder. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[34] J. Wang, S. Tan, X. Zhen, S. Xu, F. Zheng, Z. He, and L. Shao. Deep 3d human pose estimation: A review. *Computer Vision and Image Understanding*, 210:103225, 2021.

[35] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang. Diffusion Models: A Comprehensive Survey of Methods and Applications. *arXiv e-prints*, page arXiv:2209.00796, Sept. 2022.