Simulating infinite-dimensional diffusion bridges using score operator matching

Gefan Yang

March 7, 2024

1 Introduction

In the previous paper[3], we discoverd the possibility of using Denoising Score Matching (DSM) to learn the score $\nabla p(x_t, t \mid x_0, 0)$ for non-linear diffusion process using the algorithm proposed by[4]. However, we found that using only a fully-connected neural network to approximate high-dimensional score functions (more rigorously, the vector fields) can be nontrivial, specially when the learned objectives are Fourier coefficients of the score functions, the real and imaginary parts of coefficients need to be fed into the network individually as real vectors, which doubles the dimensions of the input and output. This drawback leads to the difficulty of scaling up the framework to high-dimensional cases. Besides that, since the shape is function in Hilbert spaces, the score should be defined in the context Hilbert spaces as well, which leads to the formulation of it as an operator that maps functions to functions, that is, we are going to match an operator.

Neural operators are proposed as a family of novel frameworks of neural network, which generalized the Universal Approximate Theorem of network over finite-dimensional spaces (values of functions) to infinite-dimensional function spaces. This formulation can overcome the inconsistency under different forms of discretization without additional training, and seems to show impressive performances on physical PDE solving. The essential idea behind neural operator is to formulate the unknown operator in terms of integral kernel operator. Depending on the choices of kernel function, typical neural operators are Graphic Neural Operator (GNO)[8], Low-rank Neural Operator (LNO)[11], Multipole Graphic Neural Operator (MGNO)[7] and Fourier Neural Operator (FNO)[9]. In our case, we shall mainly focus on FNO, since we are also representing functions under Fourier basis, which coincides with the FNO framework.

In addition, we need to properly define the infinite-dimensional score as either operator or function (it turns out that these two are equivalent in the later discussion). Luckily, [10] has already given a well-defined form of infinite-dimensional DSM, we shall use their definition and combine with [4]'s algorithm to develope a framework for infinite-dimensional diffusion bridge scheme.

2 Background

2.1 Score-based diffusion model and diffusion bridge

Score-matching techniques are initially developed to sample for generative purpose, i.e., sample from an unknown distribution, $p_{\text{data}}(\mathbf{x})$, which is a marginal distribution at a certain time in the context of SDEs. However, in diffusion bridge simulation, we would like to sample from, essentially, a transition distribution or its variant. The latter is accessed by Doob's h-transform, as $h(\mathbf{x}, t) := \nabla_{\mathbf{x}(t)} \log q_{tT}(\mathbf{x}(T) \mid \mathbf{x}(t))$. In the rest of notes, we shall denote $q_{st}(\mathbf{x}(t) \mid \mathbf{x}(s))$ as the transition density from $\mathbf{x}(s)$ to $\mathbf{x}(t)$ for $0 \leq s < t \leq T$. Therefore, it is necessary to think of what the objectives of the common score-based diffusion model are and how they can fit with our goals.

2.1.1 Score-based diffusion generative models

The essence of generative models is to approximate an unknown distribution $p_{\text{data}}(\mathbf{x})$. In [5], the author first proposed the concept of "score-matching", which is commonly referred as implicit score matching in the later literatures

Implicit score matching (ISM):

$$L_{\text{ISM}}(\theta) := \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \left[\frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right] \tag{1}$$

For $\theta^* = \arg\min_\theta L_{\text{ISM}}(\theta)$, $\mathbf{s}_{\theta^*}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$.

Since ISM requires the gradient of $\mathbf{s}_\theta(\mathbf{x})$, which is computationally costly. [14] proposed the denoising score matching (DSM) approach, which is a gradient-free loss:

Denoising score matching (DSM): Given the perturbation kernel $q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) := \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma^2 \mathbf{I})$, and define $q_\sigma(\tilde{\mathbf{x}}) := \int q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}$,

$$L_{\text{DSM}}(\theta) := \frac{1}{2} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \left[\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2 \right] \tag{2}$$

For $\theta^* = \arg\min_\theta L_{\text{DSM}}(\theta)$, $\mathbf{s}_{\theta^*}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x})$, and when $\sigma \to 0$, $q_\sigma(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$.

Then [12] found that DSM suffers from poor approximations in the regions where the data is sparse, therefore, they proposed to perturb the data with different perturbation kernels $q_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x})$ instead of a fixed one, which leads to multi-level denoising score matching (they refer it as noise-condition score network (NCSN) in their paper)

Multi-level denoising score matching (MDSM): Given perturbation kernels $q_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) := \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma_i^2 \mathbf{I})$ with a known sequence $\{\sigma_i\}_{i=1}^L$, where $\sigma_1 > \sigma_2 > \cdots > \sigma_{L-1} > \sigma_L \approx 0$, and define $q_{\sigma_i}(\tilde{\mathbf{x}}) := \int q_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}$

$$L_{\text{MDSM}}(\theta) := \frac{1}{L} \sum_{i=1}^{L} \lambda(\sigma_i) \mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x})} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \left[\|s_\theta(\tilde{\mathbf{x}}, \sigma_i) - \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2 \right] \tag{3}$$

For $\theta^* = \arg\min_\theta L_{\text{DSM}}(\theta)$ and $\forall \sigma \in \{\sigma_i\}_{i=1}^L$, $\mathbf{s}_{\theta^*}(\mathbf{x}, \sigma) \approx \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x})$, and when $\sigma \to 0$, $q_\sigma(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$.

[13] generalized the discrete multi-level perturbation into a continuous scenario, which leads to the formulation under the SDE framework.

SDE Score matching (SDESM): Let $\mathbf{x}(t)$ be a random variable follows the SDE:

$$d\mathbf{x} = f(\mathbf{x}, t)dt + g(\mathbf{x}, t)d\mathbf{w} \tag{4}$$

and denote by $p_t(\mathbf{x})$ the probability density of $\mathbf{x}(t)$ and use $q_{st}(\mathbf{x}(t) \mid \mathbf{x}(s))$ to denote the transition density as the statement before, and let $p_{\text{data}}(\mathbf{x}(0)) = p_{\text{data}}(\mathbf{x})$.

$$\begin{aligned} L_{\text{SDE}}(\theta) :=& \mathbb{E}_{t\sim\mathcal{U}(0,T)}\lambda(t)\mathbb{E}_{\mathbf{x}(t)\sim p_{0t}(\mathbf{x}(t)|\mathbf{x}(0))} \\ & \mathbb{E}_{\mathbf{x}(0)\sim p_{\text{data}(\mathbf{x}(0))}} \left[\|\mathbf{s}_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log q_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))\|_2^2\right] \end{aligned} \tag{5}$$

For $\theta^* = \arg\min_\theta L_{\text{SDE}}(\theta)$, $\mathbf{s}_{\theta^*}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$.

Note that the connection between MDSM and SDESM is natural, since in SDESM, $p_t(\mathbf{x}) = \int q_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))p_{\text{data}}(\mathbf{x}(0))d\mathbf{x}(0)$, which aligns with the definition in MDSM. We can see that *the score-matching techniques can only approximate the transition density from before to current moment.* While in the bridge simulation, we require the transition density from the future to current moment.

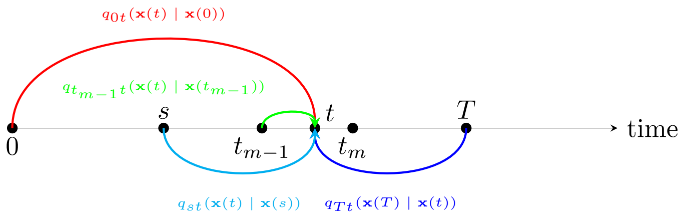
If the drift term $f(\cdot, t)$ in 4 is affine and the diffusion term $g(\cdot, t) = g(t)$ is linear, the transition density $p_{st}(\mathbf{x}(t) \mid \mathbf{x}(s))$ are always Gaussian and therefore $\nabla_{\mathbf{x}(t)} \log q_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ has a close form. For more general cases, [4] extends it to nonlinear cases.

Suspicious, should it be $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ here as before? Does it imply that $p(\mathbf{x}(0)) = 1$?

Nonlinear SDE Score matching (NSDESM): For any partition $\{t_m\}_{m=0}^M$ of the interval $[0, T]$, the loss function is defined as:

$$\begin{aligned} L_{NSDE}(\theta) := \frac{1}{2}\sum_{m=1}^{M}\int_{t_{m-1}}^{t_m} & \mathbb{E}_{\mathbf{x}(0)\sim p_{\text{data}}(\mathbf{x}(0))}\big[\|\mathbf{s}_\theta(\mathbf{x}(t), t) \\ & - \nabla_{\mathbf{x}(t)} \log q_{t_{m-1}t}(\mathbf{x}(t) \mid \mathbf{x}(t_{m-1}))\|^2_{g^2(\mathbf{x}(t),t)}\big]dt \end{aligned} \tag{6}$$

For $\theta^* = \arg\min_\theta L_{\text{NSDE}}(\theta)$, $\mathbf{s}_{\theta^*}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log q_{0t}(\mathbf{x} \mid \mathbf{x}(0))$.


$q_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$
$q_{t_{m-1}t}(\mathbf{x}(t) \mid \mathbf{x}(t_{m-1}))$
s
t
T
0
t_{m-1}
t_m
time
$q_{st}(\mathbf{x}(t) \mid \mathbf{x}(s))$
$q_{Tt}(\mathbf{x}(T) \mid \mathbf{x}(t))$

[1] shows an important theorem of the time-reversed SDE of a given SDE

Anderson's time-reversed diffusion theorem: Let X_t be a stochastic process described by the SDE:

$$d\mathbf{x} = f(\mathbf{x}, t)dt + g(\mathbf{x}, t)d\mathbf{w} \tag{7}$$

The reverse time model for $\mathbf{x}(t)$ is:

$$d\mathbf{x} = \bar{f}(\mathbf{x}, t)dt + g(\mathbf{x}, t)d\bar{\mathbf{w}} \tag{8}$$

where,

$$\bar{f}(\mathbf{x}, t) = f(\mathbf{x}, t) - \frac{1}{p_t(\mathbf{x})}\nabla_{\mathbf{x}}[p_t(\mathbf{x})g(\mathbf{x}, t)g^T(\mathbf{x}, t)] \tag{9}$$

and $p_t(\mathbf{x}) := \int q_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))p(\mathbf{x}(0))d\mathbf{x}(0)$ as defined above.

2.2 Diffusion shape process in function spaces

We are more interested in the diffusion bridge process of shape evolution, i.e, how the shapes evolve stochastically along with time, and especially, it should be conditioned on hitting a target shape (or surroundings with infinitesimal distances) within finite time. Rigorously, a shape $\mathcal{S}$ is usually treated as a function from an underlying manifold $\mathcal{M}$ to Euclidean space $\mathbb{R}^d$ and two shapes $\mathcal{S}, \mathcal{S}'$ are equivalent up to a representation if there exists a diffeomorphism $\phi \in \mathrm{Diff}(\mathcal{M})$ such that $\mathcal{S} = \mathcal{S}' \circ \phi$[2]. Suppose we have a finite collection of observations of $\mathcal{S}$ in $\mathbb{R}^d$, denote as $s^{(1)}, \ldots, s^{(N)}, s^{(i)} \in \mathbb{R}^d$, which are often referred as *landmarks*. Given the landmark representations of initial shape $\mathcal{S}_i$ and target shape $\mathcal{S}_t$, $s_i \in \mathbb{R}^{N\times d}$ and $s_t \in \mathbb{R}^{N\times d}$, the difference $x = s_t - s_i \in \mathbb{R}^{N\times d}$ are the N observations of a function $X \in L^2(\mathcal{M}, \mathbb{R}^d)$. Take the time into account and $X_t \in L^2(\mathcal{M} \times [0, T], \mathbb{R}^d)$. We can also defined an unconditional SDE for functions in Hilbert spaces, which follows similarly from:

$$dX_t = F(X_t, t)dt + B(X_t, t)dW_t, \tag{10}$$

and apply infinite-dimensional Doob's h-transform[3] to get the conditional SDE:

$$dX_t^* = \{F(X_t^*, t) + B(X_t^*, t)B^*(X_t^*, t)\nabla \log h(X_t^*, t)\}dt + B(X_t^*, t)dW_t \tag{11}$$

where $h : H \times [0, T] \to \mathbb{R}$ and $\nabla \log h : H \times [0, T] \to H$, we refer the latter as the *score operator*. We further show that under a certain choice of basis of H, a finite collection of the projections of $\nabla \log h$, denote as $(\nabla \log h)_N$ is a real-valued function that maps $H_N \times [0, T] \to \mathbb{R}^N$, where H_N is a subspace of H with the dimension of N, which is essentially $\mathbb{R}^N$, so it would make sense to match the first N components of $\nabla \log h$ as real functions between Euclidean spaces. However, in the further discussion, we will see it is possible to directly matching the infinite-dimensional score operator without projecting it into finite dimensional spaces, which requires the framework of neural operators.

2.3 Fourier neural operator

Neural operator formulation: Neural operators are designed to learn a mapping between two infinite dimensional spaces by using a finite collection of observations of input-output pairs from this mapping. Let $\mathcal{A}$ and $\mathcal{U}$ be Banach spaces of functions defined on bounded domains $D \subset \mathbb{R}^d$ and $D' \subset \mathbb{R}^{d'}$. The neural operator $\mathcal{G}_\theta$ is a parametric approximation of $\mathcal{G}_\dagger : \mathcal{A} \to \mathcal{U}$. Given the input function $a \in \mathcal{A}$ are $\mathbb{R}^{d_a}$-valued and output function $u \in \mathcal{U}$ are $\mathbb{R}^{d_u}$-valued. The neural operator is constructed as followed[6]:

1. **Lifting** ($\mathcal{Q} : \mathbb{R}^{d_a} \to \mathbb{R}^{d_{v_0}}$): Map the input $\{a : D \to \mathbb{R}^{d_a}\} \mapsto \{v_0 : D \to \mathbb{R}^{d_{v_0}}\}$ to its first hidden representation, by choosing $d_{v_0} > d_a$, $\mathcal{Q}$ is a lifting operator.
2. **Iterative Kernel Integration**: For each layer index by $l = 0, 1, \ldots, L-1$, map each hidden representation from last layer to the next $\{v_l : D_l \to \mathbb{R}^{d_{v_l}}\} \mapsto \{v_{l+1} : D_{l+1} \to \mathbb{R}^{d_{v_{l+1}}}\}$ via the action of the sum of a local linear operator (matrix) $W_l \in \mathbb{R}^{d_{v_{l+1}} \times d_{v_l}}$, a non-local integral kernel operator $\mathcal{K}_l : \{v_l : D_l \to \mathbb{R}^{d_{v_l}}\} \mapsto \{v_{l+1} : D_{l+1} \to \mathbb{R}^{d_{v_{l+1}}}\}$, and a bias function $b_l : D_l \to \mathbb{R}^{d_{v_{l+1}}}$, where $D_0 = D, D_L = D'$, and activated by a nonlinear function $\sigma_l : \mathbb{R}^{d_{v_{l+1}}} \to \mathbb{R}^{d_{v_{l+1}}}$.
3. **Projection** ($\mathcal{P} : \mathbb{R}^{d_{v_L}} \to \mathbb{R}^{d_u}$): Map the last hidden representation $\{v_L : D' \to \mathbb{R}^{d_{v_L}}\} \mapsto \{u : D' \to \mathbb{R}^{d_u}\}$ to the output function, by choosing $d_{v_L} > d_u$, $\mathcal{P}$ is a projection map.

In summary, the above description can be represented using a single formulation:

$$\mathcal{G}_\theta := \mathcal{P} \circ \sigma_L(W_{L-1} + \mathcal{K}_{L-1} + b_{L-1}) \circ \cdots \circ \sigma_1(W_0 + \mathcal{K}_0 + b_0) \circ \mathcal{Q} \tag{12}$$

The essence of neural operators is the integral kernel operator $\mathcal{K}_l$, as defined as followed:

$$(\mathcal{K}_l(v_l))(x) = \int_{D_l} \kappa^{(l)}(x, y) v_l(y) d\nu_l(v), \quad \forall x \in D_{l+1} \tag{13}$$

where $\kappa^{(l)} : D_{l+1} \times D_l \to \mathbb{R}^{d_{v_{l+1}} \times d_{v_l}}$ and ν_l is usually chosen as the Lebesgue measure on $\mathbb{R}^{d_l}$.
Fourier neural operator: Let's focus on a specific layer and omit the index l and assume $d\nu(y) = dy$. Then 13 gives the update of a single layer:

$$u(x) = \int_D \kappa(x, y) v(y) dy, \quad \forall x \in D \tag{14}$$

Let $D = \mathbb{T}^d$ is the unit torus and $\mathcal{F} : L^2(D; \mathbb{C}^{d_v}) \to \ell^2(\mathbb{Z}^d; \mathbb{C}^{d_v})$ be the Fourier transform of a complex-valued function $v : D \to \mathbb{C}^{d_v}$ and $\mathcal{F}^{-1}$ be its inverse, for $v \in L^2(D; \mathbb{C}^{d_v})$ and $w \in \ell^2(\mathbb{Z}^d; \mathbb{C}^{d_v})$, we have:

$$(\mathcal{F}v)_j(k) = \langle v_j, \psi_k \rangle_{L^2(D;\mathbb{C})}, \quad j \in \{1, \ldots, n\}, k \in \mathbb{Z}^d \tag{15}$$

$$(\mathcal{F}^{-1}w)_j(x) = \sum_{k \in \mathbb{Z}^d} w_j(k) \psi_k(x), \quad j \in \{1, \ldots, n\}, x \in D \tag{16}$$

where, for each $k \in \mathbb{Z}^d$, define,

$$\psi_k : D \to \mathbb{C} \tag{17}$$

$$x \mapsto \exp(-i2\pi(k_1 x_1 + \cdots + k_d x_d)) \tag{18}$$

By letting $\kappa(x, y) = \kappa(x-y)$ for some periodic function $\kappa : D \to \mathbb{C}^{d_u \times d_v}$ and applying convolution theorem, we find:

$$u(x) = \mathcal{F}^{-1}(\mathcal{F}(\kappa) \cdot \mathcal{F}(v))(x), \quad \forall x \in D \tag{19}$$

Therefore, we shall directly parametrize κ by its Fourier coefficients:

$$u(x) = \mathcal{F}^{-1}(R_\phi \cdot \mathcal{F}(v))(x), \quad x \in D \tag{20}$$

For each frequency mode $k \in \mathbb{Z}^d$, we have $\mathcal{F}(\kappa)(k) = R_\phi(k) \in \mathbb{C}^{d_u \times d_v}$, that is, for each k, $R_\phi(k)$ is a $d_u \times d_v$ complex matrix. We pick a finite-dimensional collection of frequency

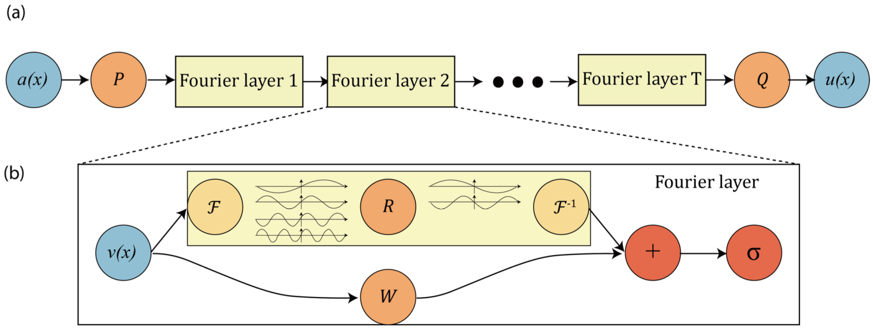


Figure 1: (a) The overall architecture of Fourier Neural Operator; (b) The detailed Fourier layer

modes by truncating the Fourier series at a maximal number of modes $k_{\max} = |Z_{k_{\max}}| = \left|\{k \in \mathbb{Z}^d : |k_j| \leq (\mathfrak{k}_{\max})_j, j = 1, \ldots, d, \mathfrak{k}_{\max} \in \mathbb{Z}^d_+\}\right|$, then we can parametrize R_ϕ directly as a $k_{\max} \times d_u \times d_v$ complex tensor and therefore drop off the parameter ϕ. The actual form of R depends on the choice of periodic function κ. In addition, if the mapping happens between the real-valued functions, we shall impose that κ is also real-valued by enforcing conjugate symmetry in the parameterization, i.e.

$$R(-k)_{m,n} = R^*(k)_{m,n}, \quad \forall k \in Z_{k_{\max}} \subset \mathbb{Z}^d, \quad m = 1, \ldots, d_u,\ n = 1, \ldots, d_v \tag{21}$$

A pictorial representation of FNO is shown in 2.3
Discrete case and FFT: The content above is the continuous case, we shall talk about the discrete implementation of FNO and apply FFT to accelerate the Fourier transformation. Let the domain D be discretized into $N \in \mathbb{N}$ points with resolution of $N_1 \times \cdots \times N_d$ for $N_1, \ldots, N_d \in \mathbb{N}$ and $\prod_{i=1}^d N_i = N$. Then we shall treat $v \in \mathbb{C}^{N_1 \times \cdots \times N_d \times d_v}$ and $\mathcal{F}(v) \in \mathbb{C}^{N_1 \times \cdots \times N_d \times d_v}$. Suppose we truncate at the position $Z_{k_{\max}} \in \mathbb{Z}^d$, and leave a subtensor of $\mathcal{F}(v) \in \mathbb{C}^{(\mathfrak{k}_{\max})_1 \times \cdots \times (\mathfrak{k}_{\max})_d \times d_v}$. We also truncate the Fourier coefficients of the kernel κ in the same way, that is, $R = \mathcal{F}(\kappa) \in \mathbb{C}^{(\mathfrak{k}_{\max})_1 \times \cdots \times (\mathfrak{k}_{\max})_d \times d_u \times d_v}$. The weighted multiplication in 20 can be written as:

$$(R \cdot \mathcal{F}(v))_{k_1,\ldots,k_d,m} = \sum_{n=1}^{d_v} R_{k_1,\ldots,k_d,m,n} (\mathcal{F}(v))_{k_1,\ldots,k_d,n}, \tag{22}$$

$$k_j = 1, \ldots, (\mathfrak{k}_{\max})_j \text{ for } j = 1, \ldots, d; \tag{23}$$

$$m = 1, \ldots, d_u \tag{24}$$

In the above multiplication, $\mathcal{F}(v)$ can be efficiently computed by d-dimensional DFT if the discretization is uniform. For $v \in \mathbb{C}^{N_1 \times \cdots \times N_d \times d_v}$, $k = (k_1, \ldots, k_d) \in \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_d}$, and $x = (x_1, \ldots, x_d) \in D$, the d-dimensional DFT $\hat{\mathcal{F}}$ and its inverse $\hat{\mathcal{F}}^{-1}$ are defined as:

$$(\hat{\mathcal{F}} v)(k) = \sum_{n_1=0}^{N_1} \cdots \sum_{n_d=1}^{N_d} v_{n_1,\ldots,n_d} \exp(-2i\pi \sum_{j=1}^{d} \frac{x_j}{N_j} k_j) \tag{25}$$

$$(\hat{\mathcal{F}}^{-1} v)(x) = \sum_{n_1=0}^{N_1} \cdots \sum_{n_d=1}^{N_d} v_{n_1,\ldots,n_d} \exp(2i\pi \sum_{j=1}^{d} \frac{k_j}{N_j} x_j) \tag{26}$$

Both 25 and 26 can be computed via Fast Fourier Transform (FFT). Note that when truncate the Fourier domain, the sum bound in 26 will end up to $Z_{k_{\max}}$ instead of N, and the truncation in this case is taking the "corner" of $(\mathcal{F}(v))$, which is similar to the computation of R.

Choice for R: In general, R should be defined to depend on $\mathcal{F}(a)$, the Fourier transform of the input $a \in \mathcal{A} : D \to \mathbb{C}^{d_a}$, we shall define $R_\phi : \mathbb{Z}^d \times \mathbb{C}^{d_a} \to \mathbb{C}^{d_u \times d_v}$ as a parametric function that maps $(k, (\mathcal{F}(a))(k))$ to the Fourier coefficients of the appropriate Fourier modes, in [6], they proposed three choices of R_ϕ:

- *direct parameterization*: Define the parameters $\phi_k \in \mathbb{C}^{d_u \times d_v}$ for each wave number $k \in \mathbb{N}$:
$$R_\phi(k, (\mathcal{F}(a))(k)) := \phi_k \tag{27}$$
- *linear parameterization*: Define the parameters $\phi_{k_1} \in \mathbb{C}^{d_u \times d_v \times d_a}, \phi_{k_2} \in \mathbb{R}^{d_u \times d_v}$, for each wave number $k \in \mathbb{N}$:
$$R_\phi(k, (\mathcal{F}(a))(k)) := \phi_{k_1}(\mathcal{F}(a))(k) + \phi_{k_2} \tag{28}$$
- *feed-forward neural network*: Let $\Phi_\phi : \mathbb{Z}^d \times \mathbb{C}^{d_a} \to \mathbb{C}^{d_u \times d_v}$ be a neural network with parameters ϕ:
$$R_\phi(k, (\mathcal{F}(a))(k)) := \Phi_\phi(k, (\mathcal{F}(a))(k)) \tag{29}$$

In summary, the algorithm of computing Fourier layer is summarized in 1. Empirically, the direct and linear parameterization have similar performances and the feed-forward network has the worst one. Besides, the direct parameterization has the simplest computation complexity and least parameters required.

Algorithm 1 Fourier layer

```
Input: $v \in \mathbb{C}^{N_1 \times \cdots \times N_d \times d_v}$, $R \in \mathbb{C}^{N_1 \times \cdots \times N_d \times d_u \times d_v}$, $W \in \mathbb{C}^{d_u \times d_v}$, $b \in \mathbb{C}^{d_u}$
  $\tilde{v} = \hat{\mathcal{F}}(v) \in \mathbb{C}^{N_1 \times \cdots \times N_d \times d_v}$ by 25;
  Truncate $\tilde{v}$ at $Z_{k_{\max}}$, obtain $\tilde{v}' \in \mathbb{C}^{(\mathfrak{k}_{\max})_1 \times \cdots \times (\mathfrak{k}_{\max})_d \times d_v}$;
  $\tilde{v}' = R \cdot \tilde{v}' \in \mathbb{C}^{(\mathfrak{k}_{\max})_1 \times \cdots \times (\mathfrak{k}_{\max})_d \times d_u}$ by 22;
  $u = \hat{\mathcal{F}}^{-1}(\tilde{v})$ by 26;
  $u = \sum_{j=1}^{d_u} (u_{\ldots,j} W_{j,\ldots} + b_j)$;
  $u = \sigma(v)$
Output: $u \in \mathbb{C}^{(\mathfrak{k}_{\max})_1 \times \cdots \times (\mathfrak{k}_{\max})_d \times d_u}$
```

2.4 Infinite-dimensional score matching

2.4.1 Define infinite-dimensional score

Now back to our shape problem, we would like to model the stochastic evolution of the displacement function $X : \mathcal{M} \to \mathbb{R}^d$, let the shape defined in $\mathbb{R}^2$, $\mathcal{M}$ be the interval $[-\pi, \pi]$ with periodic boundaries, that is, $X : [-\pi, \pi] \to \mathbb{R}^2$. If in finite-dimensional case, the score s is defined as a vector field that maps $\mathbb{R}^d \to \mathbb{R}^d$. While in the infinite case, s is an operator that maps between Hilbert spaces. [10] dedicated to define such a score operator denoted by:

$$D_{H_{\mu_0}} \Phi = D_{H_{\mu_0}} \log \frac{d\nu}{d\mu_0}, \quad D_{H_{\mu_0}} \Phi : H \to H^*_{\mu_0} \tag{30}$$

where $\Phi : H \to \mathbb{R}$ is a Fréchet differentiable Borel measurable mapping, H_{μ_0} is Cameron-Martin space, which itself is a Hilbert space, ν is the the probability measure induced by the function perturbation

$$v = u + \eta, \quad u \sim \mu, \ \eta \sim \mu_0, \ u, \eta \in H \tag{31}$$

$D_{H_{\mu_0}}$ is the Fréchet derivative and Φ suffices:

$$\frac{d\nu}{d\mu_0}(w) = \exp(\Phi(w)), \quad \mu_0\text{-a.s.}, \ w \in H \tag{32}$$

The *infinite-dimensional score* of $\nu \in H$ is $D_{H_{\mu_0}}\Phi : H \to H^*_{\mu_0}$, which is a mapping between Hilbert spaces, therefore is an *operator*, in the following text, we shall call it *score* for short.

2.4.2 Denoising score matching

Let $\mathcal{G}_\theta : H \to H^*_{\mu_0}$ be the parametric mapping and then the score matching loss is:

$$\min_\theta \mathbb{E}_{v\sim\nu} \left\| D_{H_{\mu_0}}\Phi(v) - \mathcal{G}_\theta(v) \right\|^2_{H^*_{\mu_0}}, \quad v \in H \tag{33}$$

which is intractable to compute and the denoising score matching scheme needs to be applied. Consider the Gaussian transition of $v \mid u$, $\mathcal{N}(u, C) := \gamma^u$ for μ-almost any $u \in H$, where $C : H \to H$ is the covariance operator. For μ_0-almost any $w \in H$ and μ-almost any $u \in H_{\mu_0}$,

$$\frac{d\gamma^u}{d\mu_0}(w) = \exp\left(\sum_{j=1}^{\infty} \lambda_j^{-1} \langle w, \varphi_j \rangle \langle u, \varphi_j \rangle - \frac{1}{2} \left\| C^{-1/2}(u) \right\|^2 \right) := \exp(\Psi(w; u)) \tag{34}$$

where $C\varphi_j = \lambda_j \varphi_j$ is the eigendecomposition of C. The score of each conditional γ^u is given as the Fréchet derivative of the *potential* $\Psi : H \times H_{\mu_0} \to \mathbb{R}$ in the direction of H_{μ_0}, then the *infinite-dimensional denoising score matching* loss is:

$$\min_\theta \mathbb{E}_{u\sim\mu} \mathbb{E}_{w\sim\gamma^u} \left\| D_{H_{\mu_0}}\Psi(w; u) - \mathcal{G}_\theta(w) \right\|^2_{H^*_{\mu_0}}, \quad u \in H_{\mu_0}, \ w \in H. \tag{35}$$

The first term in inf-dim DSM can be computed by

$$D_{H_{\mu_0}}\Psi(w; u) = \sum_{j=1}^{\infty} \lambda_j^{-1} \langle u, \varphi_j \rangle \varphi_j, \quad D_{H_{\mu_0}}\Psi : H \times H_{\mu_0} \to H^*_{\mu_0} \tag{36}$$

Furthermore, for $z \in H_{\mu_0}$,

$$D_{H_{\mu_0}}\Psi(w; u)(z) = \sum_{j=1}^{\infty} \lambda_j^{-1} \langle z, \varphi_j \rangle \langle u, \varphi_j \rangle \tag{37}$$

Since in, the "score" $s(X^*_t, t)$ should be also a function in H in the context of $X^* \in H$, while the learned $\mathcal{G}_\theta$ lives in the dual space. In order to derive a form that lives in H as well, the *Riesz representation theorem* shall be applied:

Riesz representation theorem: Let H be a Hilbert space equipped with inner product $\langle \cdot, \cdot \rangle : H \times H \to \mathbb{R}$. For every continuous linear function $\varphi \in H^*$, there exists a unique vector $f_\varphi \in H$, s.t.

$$\varphi(x) = \langle x, f_\varphi \rangle, \quad \forall x \in H \tag{38}$$

and the map $R : \varphi \mapsto f_\varphi, H^* \to H$, is a canonical isometric isomorphism.

By applying the Reisz map R on $\mathcal{G}_\theta$, we obtain the operator $R\mathcal{G}_\theta : H \to H$, also using the isometric property (distance preserving), we find:

$$\left\|D_{H_{\mu_0}}\Psi(v;u) - \mathcal{G}_\theta(v)\right\|^2_{H^*_{\mu_0}} = \left\|C^{-1/2}(u - R\mathcal{G}_\theta(v))\right\|^2_{H_{\mu_0}}, \quad v \in H, u \in H_{\mu_0} \tag{39}$$

where in the R.H.S, the norm is taken in H_{μ_0} instead of $H^*_{\mu_0}$. And finally, the simpler inf-dim DSM loss is:

$$\min_\theta \mathbb{E}_{u\sim\mu}\mathbb{E}_{\eta\sim\mu_0}\left\|C^{-1/2}(u - R\mathcal{G}_\theta(u+\eta))\right\|^2 \tag{40}$$

which is the infinite-dimensional analog to the original DSM loss in [14]:

$$\min_\theta \mathbb{E}_{x\sim p(x)}\mathbb{E}_{\eta\sim\mathcal{N}(0,\sigma^2 I)}\left\|\frac{\eta}{\sigma^2} + s_\theta(x+\eta)\right\|^2_2 \tag{41}$$

2.4.3 Data perturbation

31 shows the general form of perturbing function data. In [10], the author propose to perturb the data by structured noise instead of purely independent white noise, which is available as a *Gaussian process*, also called *Gaussian random field (GRF)* in some cases. Mathematically, a Gaussian process is defined as:

Gaussian process GP: Let X be a set, and let $f : X \to \mathbb{R}$ be a random function. We say that $f \sim \mathrm{GP}(\mu, k)$ if for any $n \in \mathbb{N}$ and any finite set of points $\mathbf{x} \in X^n$, the random vector $\mathbf{f} = f(\mathbf{x}) \in \mathbb{R}^n$ is n-dimensional multivariate Gaussian with prior mean vector $\mu = \mu(\mathbf{x}) \in \mathbb{R}^n$ and covariance matrix $\mathbf{K}_{\mathbf{xx}} = k(\mathbf{x}, \mathbf{x})$. The function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is called the *kernel* of Gaussian process.

A Gaussian process is uniquely determined by the mean function $\mu : X \to \mathbb{R}$ and the kernel function $k : X \times X \to \mathbb{R}$. There are many different choices of kernels, including two commonly used ones: Matérn-type kernel and radius basis function, they are defined as followed:

$$k_{\text{Matérn}}(x, x'; \sigma, \kappa, \nu) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\frac{\|x - x'\|}{\kappa}\right)^\nu K_\nu\left(\sqrt{2\nu}\frac{\|x - x'\|}{\kappa}\right) \tag{42}$$

$$k_{\text{RBF}}(x, x'; \sigma, \kappa) = \sigma^2 \exp\left(-\frac{\|x - x'\|^2}{2\kappa^2}\right) \tag{43}$$

The RBF can be treated as the limit of Matérn kernel as $\nu \to \infty$. There are many ways to sample from a GP, we here so far use the simplest and most intuitive way: Cholesky decomposition , essentially, it computes the square root of the covariance matrix computed by the kernel, which is the standard deviation, and multiply this on a discrete white noise field.

Maybe use different efficient methods, like the discrete convolution.

3 Research problems

The ultimate goal of this project is to *simulate the diffusion bridge of infinite-dimensional shape displacements*. In order to achieve this, we need to address the following issues:

- **Implement Fourier neural operator** Neural operators were initially developed for solving PDEs and there seems no usable implementation of score operator matching (in [10] used FNO but they didn't provide the source code). Also, the most comprehensive neural operator code base is neuraloperator which is written in PyTorch. A good start shall be rewriting it using JAX, but for some very simple demos, to get some experiences on the implementation side, especially the computation of loss functions. After getting know about how exactly it works, we then develope the structure used for our purpose.

- **Derive the inf-dim DSM loss for the transition density of bridge** is the generalization of original DSM (matching normal density) to bridges (matching transition density), while is also used to match normal "density" of functions, we need to derive the corresponding variation of matching the "transition density" of the functions as well.
- **Add additional conditions to the score approximator** The existing framework only allows us to bridge a single pair of points, to bridge other pairs requires additional training. We could incorporate extra condition into the network, for example, $s_\theta(x_t, x_0, t)$, just like conditional diffusion model, to generalize to different bridges without extra training.

References

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

[2] Elizabeth Baker, Thomas Besnier, and Stefan Sommer. A function space perspective on stochastic shape evolution, 2023.

[3] Elizabeth Louise Baker, Gefan Yang, Michael L. Severinsen, Christy Anna Hipsley, and Stefan Sommer. Conditioning non-linear and infinite-dimensional diffusion processes, 2024.

[4] Jeremy Heng, Valentin De Bortoli, Arnaud Doucet, and James Thornton. Simulating diffusion bridges with score matching, 2022.

[5] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[6] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces, 2023.

[7] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations, 2020.

[8] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations, 2020.

[9] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[10] Jae Hyun Lim, Nikola B. Kovachki, Ricardo Baptista, Christopher Beckham, Kamyar Azizzadenesheli, Jean Kossaifi, Vikram Voleti, Jiaming Song, Karsten Kreis, Jan Kautz, Christopher Pal, Arash Vahdat, and Anima Anandkumar. Score-based diffusion models in function space, 2023.

[11] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021.

[12] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[13] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021.

[14] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.