NUMERICAL ANALYSIS ON NEURAL NETWORK PROJECTED SCHEMES FOR APPROXIMATING ONE DIMENSIONAL WASSERSTEIN GRADIENT FLOWS

XINZHE ZUO, JIAXI ZHAO, SHU LIU, STANLEY OSHER, AND WUCHEN LI

Abstract. We provide a numerical analysis and computation of neural network projected schemes for approximating one dimensional Wasserstein gradient flows. We approximate the Lagrangian mapping functions of gradient flows by the class of two-layer neural network functions with ReLU (rectified linear unit) activation functions. The numerical scheme is based on a projected gradient method, namely the Wasserstein natural gradient, where the projection is constructed from the L^2 mapping spaces onto the neural network parameterized mapping space. We establish theoretical guarantees for the performance of the neural projected dynamics. We derive a closed-form update for the scheme with well-posedness and explicit consistency guarantee for a particular choice of network structure. General truncation error analysis is also established on the basis of the projective nature of the dynamics. Numerical examples, including gradient drift Fokker-Planck equations, porous medium equations, and Keller-Segel models, verify the accuracy and effectiveness of the proposed neural projected algorithm.

1. Introduction

Simulating gradient flows of free energies is a central problem in the computational physics of complex systems [8] and data science [1, 2]. In physics, gradient flows often arise from first-order principles, such as the Onsager principle [32]. The Onsager gradient flows are widely used in phase fields, chemistry, and biology modeling. In recent years, a particular type of Onsager gradient flow, known as Wasserstein gradient flow, has been widely studied in optimal transport communities [3, 33, 37]. It studies an infinite-dimensional pseudo-Riemannian metric in the probability distribution space known as the density manifold. The gradient flow in the Wasserstein space naturally captures the free energy dissipation properties. Depending on the choices of free energies, the Wasserstein gradient flow contains a vast class of differential equations, such as gradient drift Fokker-Planck equations, porous medium equations, and Keller-Segel models. These models are widely used in population dynamics and sampling-related optimization problems.

Key words and phrases. Optimal transport; Information Geometry; Natural gradient; Neural network functions; Convergence analysis.

Xinzhe Zuo and Jiaxi Zhao contributed equally. Jiaxi Zhao, Xinzhe Zuo, and Shu Liu are partially supported by AFOSR YIP award No. FA9550-23-1-008; Xinzhe Zuo, Shu Liu, and Stanley Osher are partially funded by AFOSR MURI FA9550-18-502 and ONR N00014-20-1-2787; Wuchen Li's work is partially supported by AFOSR YIP award No. FA9550-23-1-008, NSF DMS-2245097, and NSF RTG: 2038080.

In recent years, machine learning has brought a class of new methods in computational physics, where free energies are identified with the loss functions [11, 30]. Meanwhile, computing Wasserstein gradient flows of loss functions in terms of samples also finds their various applications, such as generative artificial intelligence [4] and transport map-based sampling methods [35]. In these applications, one often relies on the Lagrangian mapping functions to describe the Wasserstein gradient flows and deep neural networks to approximate the mapping functions due to their high expressivity and adaptivity from the compositional structure. While empirical successes of this framework have been observed in various applications [4, 35], very few theoretical results exist to explain the underlying mechanism.

Moreover, projected dynamics in neural network space are widely used to approximate Wasserstein gradient flows [14, 25]. These dynamics restrict the space of probabilities onto a finite-dimensional subspace parameterized by neural network mapping functions. For this reason, we call it the neural projected gradient dynamics. This approach originates from the natural gradient method in information geometry [1] and extends the framework set by [20]. Some basic questions about its accuracy and efficiency remain: *Even in one-dimensional space, how well do the neural projected dynamics approximate the Wasserstein gradient flow? What is the accuracy of the neural network approximation in Lagrangian mapping functions?*

In this paper, we study the numerical analysis and computational neural network projected schemes for one-dimensional Wasserstein gradient flows. The main result is sketched below. By formulating gradient flows in Lagrangian coordinates, the proposed numerical scheme takes the form of a 'preconditioned' gradient descent, where the preconditioner is the metric tensor of the statistical manifold of the parameter space. Theoretically, we first provide the derivation of the analytic solution for the inverse neural mapping metric. It is based on a special class of the ReLU network in theorem 2. We use the analytic form of the projected gradient flow formula to prove the consistency of the numerical scheme. Then, we prove in theorem 3 that the numerical schemes derived from the neural projected dynamics are of first or second-order consistency for the general Wasserstein gradient directions. These include cases of the heat flow and the Fokker-Planck equation. Furthermore, viewing our neural network model as a moving mesh method, we show in proposition 7 that the mesh will not degenerate during the simulation.

In numerics, the advantages of the proposed method are twofold. First, using a two-layer neural network as our basis function, the proposed method can be regarded as a 'moving-mesh' method in Lagrangian coordinate, which demonstrates very promising performance even when the number of parameters of the neural network is very limited. In particular, our numerical examples can achieve an accuracy of 10^{-3} with less than 100 neurons. Second, using the Wasserstein gradient flow formulation, the proposed method is very easy to implement since it can make use of the automatic differentiation feature from popular machine learning libraries such as PyTorch.

Nowadays, the computation of Wasserstein gradient flows (WGFs) has attracted great interests from researchers in various communities such as mathematics, physics, statistics, and machine learning. Classical numerical methods [9] have been introduced to directly

evaluate the probability density function. Recently, algorithms that approximate the Lagrangian mapping functions associated with WGFs have been invented. We refer the readers to [8] and references therein for related discussions. These treatments automatically preserve non-negativity and total mass. Together with the fast-developing deep learning techniques, they inspire a series of research on composing scalable, sampling-friendly computational methods for WGFs in higher-dimensional spaces [13, 16, 18, 25, 27]. Recently, deep learning-based algorithms for computing the Lagrangian coordinates of the Wasserstein Hamiltonian flows, or more generally mean field control problems, have also been introduced in [28, 34, 38].

Our treatment of projecting the WGFs onto the parameter space is also known as the natural gradient method, which are first introduced in [1] (w.r.t. Fisher-Rao metric) and [10] (w.r.t. Wasserstein metric). Here the projected matrix is often named information matrix, namely Fisher information matrix and Wasserstein information matrix, depending on the usage of metrics in probability space. This method recently finds its application in large-scale optimization problems [29]. In recent research [7, 12, 14], the authors aim to calculate general evolution equations by directly leveraging the neural network representation of the time-dependent solution. They endow the evolution of the equation in the functional space into the parameter space of the neural network to obtain a finite-dimensional ordinary differential equation, which can be readily integrated via the Runge-Kutta solvers. Numerical properties of the ReLU neural network families have been investigated in [15].

Compared to previous studies, we study the numerical analysis of neural network projected dynamics for approximating WGFs. In one-dimensional space, we provide the error analysis for the neural projected dynamics with a two-layer neural network. We numerically verify the proposed error analysis. In particular, we formulate a class of explicit schemes from the neural network projected dynamics. This study continues the study of the Wasserstein information matrix on neural network models; see related discussions in [21, 22, 25].

The paper is organized as follows. In Section 2, we briefly review the formulation of Wasserstein gradient flows of free energies in both Eulerian and Lagrangian coordinates. We formulate the projected Wasserstein gradient flows over neural network models in Section 3. In Section 4, we conduct the numerical analysis of the proposed neural projected dynamics in two-layer neural network functions. In Section 5, we verify the accuracy of the proposed algorithm with numerical examples in Fokker-Planck equations, porous medium equations, and Keller-Segel models.

2. Review of Wasserstein gradient flows and Lagrangian coordinates

In this section, we prepare the theoretical foundations of Wasserstein gradient flows with a focus on Lagrangian description (diffeomorphism mapping functions) and the associated microscopic particle dynamics. See details in [3, 37].

2.1. **Wasserstein gradient flows.** Suppose Ω is a domain in the Euclidean space R^d. Denote the probability space

$$\mathsf{P}(\Omega) = \left\{ p(\ ) \ 2 \ C^1 \ : \ \int_\Omega p(x)dx = 1, \quad p(\ ) \quad 0 \right\}.$$

Given an energy functional $\mathsf{F}(\ ) : \Omega \ ! \ \mathsf{R}$, we consider the following evolution equation associated with $\mathsf{F}(\ )$,

$$\partial_t p(t,x) = \ r_x \ (p(t,x) \, r_x \frac{\delta}{\delta p}\mathsf{F}(p)), \quad p(\ ,0) = p_0, \tag{1}$$

with Neumann boundary condition $p(t,x) \, r_x \frac{}{p}\mathsf{F}(p) \ \boldsymbol{n} = 0$ where $\boldsymbol{n}$ is the outward pointing vector on boundary $\partial\Omega$. $\frac{}{p}$ is the L^2 first variation operator w.r.t. density variable p. The mass of $p(t,\ )$ is conserved and always equals 1. An important fact about (1) is that this equation can be treated as the gradient flow of F on $\mathsf{P}(\Omega)$. To be more specific, by endowing the probability space $\mathsf{P}(\Omega)$ with the L^2 Wasserstein metric g_W, we can view $(\mathsf{P}(\Omega), g_W)$ as a Riemannian manifold, and (1) is the gradient flow on such manifold with respect to g_W.

Let us briefly review several facts. We first define the metric g_W at arbitrary $p \ 2 \ \mathsf{P}(\Omega)$, which is identified via the continuity equation (that is, tangent vectors) whose driving vector field belongs to the closure of all gradient fields $r_x\psi : \Omega \ ! \ \mathsf{R}^d$ with $\psi \ 2 \ C^1(\Omega)$ in $L^2(p)$-norm. Consider a smooth curve $p_i(t,\ )_{t2(\ ;\ )}$ $(i = 1,2)$ passing through p at $t = 0$ on $\mathsf{P}(\Omega)$. Suppose the probability evolution $p_i(t,\ )$ is driven by the gradient field $r_x\psi_i(\ )$ at $t = 0$, i.e., $\psi_i(\ )$ solves

$$\partial_t p_i(0,x) + \ r_x \ (p_i(0,x) \, r \, \psi_i(x)) = 0, \quad i = 1,2.$$

We define the L^2 Wasserstein metric $g_W(\ ,\ )$ at p as a symmetric, positive-definite bilinear form,

$$g_W(\partial_t p_1(0,\ ), \partial_t p_2(0,\ )) = \int_\Omega r_x\psi(x) \quad r_x\psi_2(x)p(x) \ dx.$$

Recall the definition of the gradient of a smooth function f on a Riemannian manifold (M,g) as

$$g(\mathrm{grad} f(x), \dot{x}(0)) = \frac{d}{dt} f(x(t)),$$

for any smooth curves $x(t)_{t \ 2 \ (\ \epsilon,\epsilon)}$ passing through x at $t = 0$. Switching back to our case, for the functional F defined on $(\mathsf{P}(\Omega), g_W)$, we define the gradient of F w.r.t. Wasserstein metric g_W at p as

$$g_W(\mathrm{grad}_W \mathsf{F}(p), \partial_t p(0,\ )) = \frac{d}{dt}\mathsf{F}(p(t,\ ))\bigg|_{t=0}.$$

Here $p(t,\ )_{t2(\ ;\ )}$ is arbitrary curve on $\mathsf{P}(\Omega)$ with $p(0,\ ) = p(\ )$. Suppose $p(t,\ )$ is guided by the gradient field $r_x\psi$ at time $t = 0$, Then the right-hand side can be computed as

$$\begin{aligned}\frac{d}{dt}\mathsf{F}(p(t,\ )) = \int_\Omega \frac{\delta \mathsf{F}(p(0,\ ))}{\delta p}(x)\partial_t p(0,x) \ dx &= \int_\Omega \frac{\delta \mathsf{F}(p)}{\delta p}(x)(\quad r_x \ (p(x) \, r_x\psi(x))) \ dx \\ &= \int_\Omega r \, \frac{\delta \mathsf{F}(p)}{\delta p}(x) \quad r_x\psi(x)p(x)dx.\end{aligned}$$

Recall the definition of the metric g_W, it is not difficult to verify that the gradient field associated with $\mathrm{grad}_W \mathsf{F}(p)$ is $\nabla_x \frac{1}{p} \mathsf{F}(p)$. Thus,

$$\mathrm{grad}_W \mathsf{F}(p) = \nabla_x \cdot (p(t,x) \nabla_x \frac{\delta \mathsf{F}(p)}{\delta p}(x)),$$

and the Wasserstein gradient flow $\partial_t p = \mathrm{grad}_W \mathsf{F}(p)$ can be formulated as equation (1).

We provide several examples of WGFs. In these examples, we assume $\Omega = \mathbb{R}^d$.

(Fokker-Planck equation) Consider

$$\mathsf{F}(p) = \int_\Omega V(x)p(x)dx + \gamma \int_\Omega p(x) \log p(x) dx.$$

Then the Wasserstein gradient of F equals

$$\begin{aligned} \mathrm{grad}_W \mathsf{F}(p) &= \nabla_x \cdot (p(x) \nabla_x (V(x) + \gamma(\log p(x) + 1))) \\ &= \nabla \cdot (p(x) \nabla_x V(x)) - \gamma \Delta_x p(x). \end{aligned}$$

The corresponding WGF is the Fokker-Planck equation

$$\partial_t p(t,x) = \nabla_x \cdot (p(t,x) \nabla_x V(x)) + \gamma \Delta_x p(t,x). \tag{2}$$

(Porous medium equation) Consider

$$\mathsf{F}(p) = \frac{p^m}{m - 1}.$$

One computes

$$\mathrm{grad}_W \mathsf{F}(p) = \nabla_x \cdot (p(t,x) \nabla_x (\frac{m}{m-1} p(x)^{m-1})) = \nabla_x \cdot (\nabla_x (p(x)^m)) = \Delta_x p(x)^m.$$

Thus, the corresponding WGF yields the porous medium equation

$$\partial_t p(t,x) = \Delta_x p(t,x)^m. \tag{3}$$

(Keller-Segel equation) Another well-known WGF is by choosing F as the sum of the internal energy and the interaction energy

$$\mathsf{F}(p) = \int_\Omega U(p(x))\, dx + \frac{1}{2} \iint_{\Omega \times \Omega} W(x - y) p(x) p(y)\, dxdy,$$

where U is a certain smooth function defined on $\mathbb{R}_+$, and $W(\cdot) \in C(\mathbb{R}_+; \mathbb{R})$ is a kernel function.

We calculate

$$\mathrm{grad}_W \mathsf{F}(p) = \nabla_x \cdot (p(x) \nabla_x (U'(p(x)) + W * p(x))),$$

where we denote the convolution $W * p(x) = \int_\Omega W(x - y) p(y)\, dy$. The WGF associated with this functional is the Keller-Segel equation

$$\partial_t p(t,x) = \nabla_x \cdot (p(t,x) \nabla_x U'(p(t,x))) + \nabla_x \cdot (p(t,x) \nabla_x (W * p_t(x))). \tag{4}$$

2.2. **Lagrangian coordinates & Particle dynamics.** Consider a mapping function $T\colon Z \to \Omega$. Here $z \in Z$ is an input space, $\Omega \subset \mathbb{R}^d$ is the domain on which WGF is defined. To alleviate our discussion, we assume $Z = \Omega$. Let us further assume $T \in C^1(Z,\Omega)$, and the Jacobian matrix $D_z T(z)$ is non-singular for all $z \in Z$, i.e., $\det(D_z T(z)) \neq 0$ on Z. This also guarantees that T is injective. Given a smooth reference probability density $p_{\mathrm{r}} \in \mathcal{P}(Z)$, we denote the pushforwarded probability density of p_r by T as

$$p = T_{\#} p_r,$$

where $T_{\#} : \mathcal{P}(Z) \to \mathcal{P}(\Omega)$ is the pushforward operator defined as

$$\int_\Omega f(x) T_{\#} p_r(x)\, dx = \int_Z f(T(z)) p_r(z)\, dz, \quad \text{for all } f \circ T \in L^1(p_r).$$

The density function of p satisfies

$$p(T(z))\det(D_z T(z)) = p_{\mathrm{r}}(z). \quad \forall\, z \in Z. \quad \text{i.e., } p(x) = \frac{p_r}{\det(D_z T)} \circ T^{-1}(x) \quad \forall\, x \in \Omega. \tag{5}$$

Such pushforward map T used for constructing probability distribution p is usually called the *Lagrangian coordinate*. We now imitate the derivation of the WGF to help formulate its counterpart under the Lagrangian coordinate.

We denote $\mathcal{O}$ as the space of smooth, $L^2(p_r)$ integrable pushforward maps with non-zeros Jacobian, i.e.,

$$\mathcal{O} = \left\{ T \in C^1(Z,\Omega) \;:\; \det(D_z T) \neq 0,\ \int_Z |T(z)|^2 p_z(z)\, dz < \infty \right\}.$$

Then the pushforward operation $\# : \mathcal{O} \to \mathcal{P}(\Omega)$ introduces a submersion from the space of pushforward maps (diffeomorphisms) to the space of probability densities.

In order to derive the Wasserstein gradient flows (WGFs) on the space $\mathcal{O}$ of pushforward maps instead of the probability space $\mathcal{P}(\Omega)$, we first build up certain metric $\langle\cdot,\cdot\rangle$ on $\mathcal{O}$ that corresponds to the Wasserstein metric g_W. As illustrated in [33], g_W is obtained by pulling back the $L^2(p_r)$ norm on $\mathcal{O}$ via submersion $\#$. Thus, a way of choosing the metric is

$$\langle \mathbf{u}_1, \mathbf{u}_2 \rangle = \int_Z \mathbf{u}_1(z) \cdot \mathbf{u}_2(z) p_r(z)\, dz, \quad \forall\, \mathbf{u}_1, \mathbf{u}_2 \in L^2(p_r) \bigcap C^1(Z,\Omega).$$

Now for any smooth functional $F : \mathcal{P}(\Omega) \to \mathbb{R}$, the composition $F^{\#} = F \circ \# : \mathcal{O} \to \mathbb{R}$ defines its corresponding functional on $\mathcal{O}$. Follow similar arguments presented in 2.1, we compute the gradient of $F^{\#}$ with respect to the metric $\langle\cdot,\cdot\rangle$ as

$$\mathrm{grad}_{\langle\cdot,\cdot\rangle} F^{\#}(T) = \frac{1}{p_{\mathrm{r}}(\cdot)} \frac{\delta F^{\#}(T)}{\delta T}(\cdot).$$

Here, $\frac{\delta}{\delta T}$ is the $L^2(m)$ (m denotes the Lebesgue measure) first variational w.r.t. the pushforward map T.

Thus, the gradient flow of $F^{\#}$ on $\mathcal{O}$ is formulated as

$$\partial_t T(t,\cdot) = -\mathrm{grad}_{\langle\cdot,\cdot\rangle} F^{\#}(T(t,\cdot)) = -\frac{1}{p_{\mathrm{r}}(\cdot)} \frac{\delta F^{\#}(T(t,\cdot))}{\delta T}(\cdot).$$

The variation $\frac{\delta}{\delta T}$ is calculated as

$$\frac{\delta \mathsf{F}^{\#}(T)}{\delta T}(z) = \left(\nabla_x \frac{\delta \mathsf{F}(T_{\#}p_r)}{\delta p} \right) \quad T(z) p_r(z).$$

The above equation can also be written as

$$\partial_t T(t,z) = \quad \left(\nabla_x \frac{\delta \mathsf{F}(T(t,\ )_{\#}p_r)}{\delta p} \right) \quad T(t,z). \tag{6}$$

If we denote $p(t,\ ) = T(t,\ )_{\#}p_r$, one can verify that $p(t,\ )$ exactly solves equation (1) for WGF with $p_0 = T(0,\ )_{\#}p_r$, which justifies the equivalence between the gradient flow (6) in Lagrangian coordinates (i.e., the map $T(t,\ )$) and the WGF (1) expressed by using Eulerian coordinate (i.e., the density function $p(t,\ )$).

Such gradient flow (6) on the space of diffeomorphisms also forms a microscopic picture of particle dynamics of the WGF (1). For any random reference sample $z \quad p_r$, by setting $\mathbf{x}_t = T(t,z)$, it is not hard to verify that $\mathbf{x}_t$ evolves w.r.t. the dynamic

$$\frac{d\mathbf{x}_t}{dt} = \quad \left(\nabla_x \frac{\delta}{\delta p} \mathsf{F}(p_t) \right)(\mathbf{x}_t), \quad \mathbf{x}_0 = T(0,z), \quad z \quad p_r. \tag{7}$$

Here we denote $p_t = T(t,\ )_{\#}p_r$. p_t can be equivalently treated as the probability density of the random particle $\mathbf{x}_t$. In this dynamic, the movement of a single agent $\mathbf{x}_t$ is determined by the instant population density p_t evaluated at $\mathbf{x}_t$. Such an approach offers a microscopic and deterministic interpretation of various diffusive processes possessing WGF structures.

The aforementioned examples of WGF can be formulated as the gradient flows under Lagrangian coordinates (6) as well as the particle dynamics (7). We summarize this in the following Table 1. We assume $T(0,\ )_{\#}p_r = p_0$ as the initial condition for (6), and $\mathbf{x}_0 \quad p_0$ as the initial distribution of the random particle $\mathbf{x}_t$ in (7). We denote $p_t = T(t,\ )_{\#}p_r$ in equation (6). Accordingly, we denote p_t as the probability density of the stochastic particle $\mathbf{x}_t$ in the dynamic (7).

WGF	Gradient flow in Lagrangian coordinates Particle dynamic
Fokker-Planck (2)	$\partial_t T(t,z) = \quad \nabla_x (V + \gamma \log p_t) \quad T(t,z)$ $\frac{d\mathbf{x}_t}{dt} = \quad \nabla_x V(\mathbf{x}_t) \quad \gamma \nabla_x \log p_t(\mathbf{x}_t)$
Porous-medium (3)	$\partial_t T(t,z) = \quad \frac{m}{m-1} p_t(T(t,z))^{m-1} \nabla_x p_t \quad T(t,z)$ $\frac{d\mathbf{x}_t}{dt} = \quad \frac{m}{m-1} p_t(\mathbf{x}_t)^{m-1} \nabla p_t(\mathbf{x}_t)$
Keller-Segel (4)	$\partial_t T(t,z) = \quad \nabla_x (U^0(p_t) + W \quad p_t) \quad T(t,z)$ $\frac{d\mathbf{x}_t}{dt} = \quad \nabla_x U^0(p_t(\mathbf{x}_t)) \quad \nabla_x W \quad p_t(\mathbf{x}_t)$

TABLE 1. Gradient flows under Lagrangian coordinates & Particle dynamics associated with the WGFs.

3. Neural projected Wassersetin gradient flows and their algorithms

As discussed in Section 2, instead of the direct evaluation of the density function of the Wasserstein gradient flow, it suffices to compute the time-dependent Lagrangian mapping $T(t,\ )$. In this research, we approximate $T(t,\ )$ via neural networks parametrized by time-dependent parameter $f\theta_t g$. The evolution of θ_t is obtained by projecting the gradient flow (6) onto the parameter space Θ. In this section, we briefly review the basic definitions of neural network mapping functions. We next study a metric space for neural mapping functions and formulate several neural mapping dynamics for $f\theta_t g$.

3.1. Neural network activation functions.

We first provide the definition of a neural network mapping function. Consider a mapping function

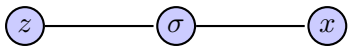
$$f\colon Z \quad \Theta \ ! \ \Omega,$$

where Z R^l is the latent space, Ω R^d is the sample space and Θ R^D is the parameter space. In this paper, we consider the following network structure

$$f(\theta, z) = \frac{1}{N}\sum_{i=1}^{N} a_i \sigma\Big(z \quad b_i\Big),$$

where $\theta = (a_i, b_i)$ 2 R^D, $D = (l+1)N$. Here N is the number of hidden units (neurons). a_i 2 R is the weight of unit i. b_i 2 R^l is an offset (location variable). $\sigma\colon \mathsf{R}$! R is an activation function, which satisfies $\sigma(0) = 0$, 1 2 $\partial\sigma(0)$. From now on, we assume that f is invertible, monotone, and is continuous w.r.t. both z and θ variables.

For example, let $N = d = 1$, and $b_1 = 0$. Define a two layer neural network by



The following neural network mapping functions have been widely used.

Example 1 (Linear). *Denote* $\sigma(x) = x$. *Consider*

$$f(\theta, z) = \theta z, \quad \theta \ 2 \ \mathsf{R}_+.$$

Example 2 (ReLU). *Denote* $\sigma(x) = \max f x, 0 g$. *Consider*

$$f(\theta, z) = \theta \max f z, 0 g, \quad \theta \ 2 \ \mathsf{R}_+.$$

Example 3 (Sigmoid). *Denote* $\sigma(x) = \frac{1}{1+e^{\ 2x}}$. *Consider*

$$f(\theta, z) = \frac{\theta}{1 + e^{\ 2z}}, \quad \theta \ 2 \ \mathsf{R}_+.$$

In Section 4 (theoretical results) and Section 5 (numerical examples), we focus mainly on the case where $l = d = 1$, $D = 2N$. And $\sigma(\ )$ is the ReLU activation function.

3.2. **Neural mapping models and energies.** In this subsection, we consider the following probability density functions generated by neural network mapping functions. We call them the neural mapping models.

Definition 1 (Neural mapping models)**.** *Let us de ne a xed input reference probability density* $p_{\mathrm{r}} \in \mathsf{P}(Z) = \left\{p(z) \in C^1(Z) \colon \int_Z p_{\mathrm{r}}(z)dz = 1,\ p(z) \geq 0\right\}$*. Denote a probability density generated by a neural network mapping function by the pushforward operator:*

$$p = f_{\#}p_{\mathrm{r}} \in \mathsf{P}(\Omega),$$

In other words, p *satis es the following Monge-Ampere equation by*

$$p(f(\theta, z))\det(D_z f(\theta, z)) = p_{\mathrm{r}}(z)\,, \tag{8}$$

where $D_z f(\theta, z)$ *is the Jacobian of the mapping function* $f(\theta, z)$ *w.r.t. variable* z*.*

Definition 2 (Neural mapping energies)**.** *Given an energy functional* $\mathsf{F} \colon \mathsf{P}(\Omega) \to \mathbb{R}$*, we can construct a neural mapping energy* $F \colon \Theta \to \mathbb{R}$ *by*

$$F(\theta) = \mathsf{F}(f_{\#}p_{\mathrm{r}}).$$

Many applications in machine learning and scientific computing can be cast into the following optimization problem

$$\min_{\theta \in \Theta} F(\theta).$$

Here, F often measures the closeness between the neural mapping model and the target or data density distribution. Several concrete examples of neural mapping energies F are given below. For simplicity of presentation, we often write the integration operator w.r.t. density p_{r} over domain Z by the expectation operator $\mathsf{E}_{z \sim p_{\mathrm{r}}}$. Later in Section 3.5, we provide several examples of the energy functional F including the potential, the interaction (E.g. maximum mean discrepancy) and the internal (information entropy/divergence) functionals. They are commonly used in machine learning and optimal transport communities; see details in [3, Section 9].

To summarize, the neural mapping energies are functionals F written in terms of the mapping functions $f(\theta, z)$. This allows us to perform optimization on the finite dimensional space Θ instead of the infinite dimensional space $\mathsf{P}(\Omega)$.

3.3. **Neural mapping metric space.** We next consider a mapping space parameterized by a neural mapping function $f(\theta, \ )$. We can measure the difference between two neural mapping functions by the L^2 distance thanks to the following definition.

Definition 3 (Neural mapping distance)**.** *De ne a distance function* $\mathrm{Dist}_{\mathrm{W}} \colon \Theta \times \Theta \to \mathbb{R}$ *as*

$$\begin{aligned}\mathrm{Dist}_{\mathrm{W}}(f_{\theta^0\#}p_{\mathrm{r}}, f_{\theta^1\#}p_{\mathrm{r}})^2 &= \int_Z \left\|f(\theta^0, z) - f(\theta^1, z)\right\|^2 p_{\mathrm{r}}(z)dz \\ &= \sum_{m=1}^{d} \mathsf{E}_{z \sim p_{\mathrm{r}}}\left[\left\|f_m(\theta^0, z) - f_m(\theta^1, z)\right\|^2\right],\end{aligned}$$

where θ^0, θ^1 2 Θ are two sets of neural network parameters and k k is the Euclidean norm in R^d.

In the above definition, $\mathrm{Dist}_{\mathrm{W}}$ represents a distance function for two given neural mapping functions $f(\theta^0, \ )$ and $f(\theta^1, \ )$. In fact, the L^2 distance between neural mapping functions induces a metric on neural network parameters. Similar Riemannian geometry for feed-forward neural networks is also studied in [31].

We next consider the Taylor expansion of the distance function. Let $\Delta\theta$ 2 R^D,

$$\begin{aligned}
&\mathrm{Dist}_{\mathrm{W}}(f_{\ +\Delta\ \#}p_{\mathrm{r}}, f_{\ \#}p_{\mathrm{r}})^2\\
=&\sum_{m=1}^{d} \mathsf{E}_{z\ \ p_{\mathrm{r}}}\Big[k f_m(\theta+\Delta\theta, z) \quad f_m(\theta, z)k^2\Big]\\
=&\sum_{m=1}^{d}\sum_{i=1}^{D}\sum_{j=1}^{D} \mathsf{E}_{z\ \ p_{\mathrm{r}}}\Big[\partial_{\ i} f_m(\theta, z)\partial_{\ j} f_m(\theta, z)\Big]\Delta\theta_i\Delta\theta_j + o(k\Delta\theta k^2)\\
=&\Delta\theta^{\mathsf{T}} G_{\mathrm{W}}(\theta)\Delta\theta + o(k\Delta\theta k^2).
\end{aligned}$$

Here G_{W} is a Gram-type matrix function. We summarize its definition below.

Definition 4 (Neural mapping metric)**.** *De ne a matrix function $G_{\mathrm{W}}\colon \Theta \ ! \ \mathsf{R}^{D\ \ D}$. Denote $G_{\mathrm{W}}(\theta) = (G_{\mathrm{W}}(\theta)_{ij})_{1\ \ i;j\ \ D}$, such that*

$$G_{\mathrm{W}}(\theta)_{ij} = \sum_{m=1}^{d} \mathsf{E}_{z\ \ p_{\mathrm{r}}}\Big[\partial_{\ i} f_m(\theta, z)\partial_{\ j} f_m(\theta, z)\Big].$$

We also write

$$G_{\mathrm{W}}(\theta) = \mathsf{E}_{z\ \ p_{\mathrm{r}}}\Big[\ r\ \ f(\theta, z)\ r\ \ f(\theta, z)^{\mathsf{T}}\Big],$$

where we denote $r\ \ f(\theta, z) = (\partial_{\ i} f_m(\theta, z))_{1\ \ i\ \ D;1\ \ m\ \ d}$ 2 $\mathsf{R}^{D\ \ d}$.

From now on, we call (Θ, G_{W}) the *neural mapping metric space*. Here we always assume that $G_{\mathrm{W}}(\theta)$ is a positive definite matrix in $\mathsf{R}^{D\ \ D}$.

3.4. **Neural mapping dynamics.** In this subsection, we derive some analogies of Wasserstein gradient flows in the neural mapping metric space (Θ, G_{W}). Shortly, we apply them to define the neural mapping dynamics and compare them with their counterparts in L^2 mapping metric space and L^2 Wasserstein metric probability space. From now on, we assume that f is smooth w.r.t. parameter θ. This is not true for the ReLU activation function, which will be studied in detail in later sections.

The next proposition provides gradient operators of a function F 2 $C^2(\Theta; \mathsf{R})$ in the neural mapping metric space (Θ, G_{W}).

Proposition 1 (Neural mapping gradient operators)**.** *The gradient operator of F in (Θ, G_{W}), $\mathrm{grad}_{\mathrm{W}} F(\theta) = (\mathrm{grad}_{\mathrm{W}} F(\theta)_k)_{k=1}^{D}$, is given by*

$$\mathrm{grad}_{\mathrm{W}} F(\theta)_k = \sum_{i=1}^{D} G_{\mathrm{W}}^{\ 1}(\theta)_{ki}\partial_{\ i} F(\theta).$$

Proof. We briefly derive the gradient operator of F in (Θ, G_{W}) below. Suppose $\theta(t) = \theta_t$ is a smooth curve passing through the point $\theta(0) = \theta$. Consider a Taylor expansion of $F(\theta_t)$ at $t = 0$ by

$$\begin{aligned} F(\theta_t) =& F(\theta) + t \quad \frac{d}{dt} F(\theta_t) \jmath_{t=0} + o(t) \\ =& F(\theta) + t \quad (G_{\mathrm{W}}(\theta) \quad \mathrm{grad}_{\mathrm{W}} F(\theta), \dot{\theta}) + o(t), \end{aligned} \tag{9}$$

where we denote $\frac{d}{dt}\theta_t \jmath_{t=0} = \dot{\theta}$. Comparing linear terms of t in (9), we have

$$\begin{aligned} (G_{\mathrm{W}}(\theta) \quad \mathrm{grad}_{\mathrm{W}} F(\theta), \dot{\theta}) =& \frac{d}{dt} F(\theta_t) \jmath_{t=0} \\ =& (\mathsf{r} \quad F(\theta), \dot{\theta}), \end{aligned}$$

for any $\dot{\theta} \in T_\theta \Theta = \mathsf{R}^d$. Thus

$$\mathrm{grad}_{\mathrm{W}} F(\theta) = G_{\mathrm{W}}^{-1}(\theta) \, \mathsf{r} \quad F(\theta).$$

We are ready to present the neural mapping gradient flow, which will be used for our first-order algorithm in neural mapping optimization problems.

Proposition 2 (Neural mapping gradient flows)**.** *Consider an energy functional* $\mathsf{F} : \mathsf{P}(\Omega) \to \mathsf{R}$*. Then the gradient flow of function* $F(\theta) = \mathsf{F}(f_{\#} p_{\mathrm{r}})$ *in* (Θ, G_{W}) *is given by*

$$\frac{d\theta}{dt} = - \mathrm{grad}_{\mathrm{W}} F(\theta). \tag{10}$$

In particular,

$$\begin{aligned} \frac{d\theta_i}{dt} = - \sum_{j=1}^{D} \sum_{m=1}^{d} & \left(\mathsf{E}_{z \sim p_{\mathrm{r}}} \Big[\mathsf{r} \quad f(\theta, z) \, \mathsf{r} \quad f(\theta, z)^{\mathsf{T}} \Big] \right)_{ij}^{-1} \\ & \mathsf{E}_{\tilde{z} \sim p_{\mathrm{r}}} \Big[\mathsf{r}_{x_m} \frac{\delta}{\delta p} \mathsf{F}(p)(f(\theta, \tilde{z})) \quad \partial_{\theta_j} f_m(\theta, \tilde{z}) \Big], \end{aligned}$$

where $\frac{\delta}{\delta p(x)}$ *is the* L^2*-first variation w.r.t. variable* $p(x)$*,* $x = f(\theta, z)$*.*

Proof. As the neural mapping metric is given in definition 4, it suffices to calculate the formula for the Euclidean gradient $\partial_{\theta_j} F(\theta)$ as follows:

$$\begin{aligned} \partial_{\theta_j} F(\theta) =& \int_\Omega \partial_{\theta_j} \rho_\theta(x) \frac{\delta}{\delta p} \mathsf{F}(\rho_\theta)(x) dx \\ =& \int_\Omega \quad \mathsf{r}_x \quad \left[\rho_\theta(x) \partial_{\theta_j} f(\theta, f(\theta, \cdot)^{-1}(x)) \right] \frac{\delta}{\delta p} \mathsf{F}(\rho_\theta)(x) dx \\ =& \int_\Omega \partial_{\theta_j} f(\theta, f(\theta, \cdot)^{-1}(x)) \quad \mathsf{r}_x \left(\frac{\delta}{\delta p} \mathsf{F}(\rho_\theta) \right)(x) \rho_\theta(x) dx \\ =& \mathsf{E}_{z \sim p_{\mathrm{r}}} \Big[\partial_{\theta_j} f(\theta, z) \quad \mathsf{r}_x \left(\frac{\delta}{\delta p} \mathsf{F}(p) \right)(f(\theta, z)) \Big]. \end{aligned}$$

Here we denote $\rho_\theta = f_{\#} p_{\mathrm{r}}$.

3.5. **Neural projected Wasserstein flows.** The dynamics in parameter space can be formulated in terms of mappings and probability densities. For simplicity of discussion, we demonstrate that the neural mapping gradient flow is a projected Wasserstein gradient flow. Here the projection is from the full mapping space into a neural parameterized mapping space. Concretely, we present the following reformulations of equation (10), which are in terms of mapping functions and probability density functions. The proof is based on the gradient flow equation in proposition 2 and the application of the chain rule.

Proposition 3 (Neural projected Wasserstein gradient flows)**.** *Dynamic* (10) *in term of mapping functions* $f(\theta,z)=(f_m(\theta,z))_{m=1}^d$ *leads to*

$$\frac{\partial}{\partial t}f_m(\theta(t),z)= \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{n=1}^{d}\partial_{\ i}f_m(\theta,z)\Big(\mathsf{E}_{\tilde{z}\ \ p_{\mathrm{r}}}\Big[r\ \ f(\theta,\tilde{z})\,r\ \ f(\theta,\tilde{z})^{\mathsf{T}}\Big]\Big)^{-1}_{ij} \mathsf{E}_{\tilde{z}\ \ p_{\mathrm{r}}}\Big[r_{\ x_n}\frac{\delta}{\delta p(x)}\mathsf{F}(p)(f(\theta,\tilde{z}))\ \ \partial_{\ j}f_n(\theta,\tilde{z})\Big].$$

We present several examples of neural mapping Wasserstein gradient flows from proposition 2.

Example 4 (Neural projected linear transport equation)**.** *Consider a linear energy given by*

$$\mathsf{F}(p)=\int_\Omega V(x)p(x)dx.$$

In this case, the neural projected gradient flow satisfies

$$\frac{d\theta}{dt}= G_{\mathrm{W}}^{-1}(\theta)\ \ \mathsf{E}_{\tilde{z}\ \ p_{\mathrm{r}}}\Big[r\ \ V(f(\theta,\tilde{z}))\Big]. \tag{11}$$

In details,

$$\frac{d\theta_i}{dt}= \sum_{j=1}^{D}\Big(\mathsf{E}_{z\ \ p_{\mathrm{r}}}\Big[r\ \ f(\theta,z)\,r\ \ f(\theta,z)^{\mathsf{T}}\Big]\Big)^{-1}_{ij}\ \ \mathsf{E}_{\tilde{z}\ \ p_{\mathrm{r}}}\Big[r_{\ x}V(f(\theta,\tilde{z}))\ \ \partial_{\ j}f(\theta,\tilde{z})\Big].$$

Example 5 (Neural projected interaction transport equation)**.** *Consider an interaction energy given by*

$$\mathsf{F}(p)=\frac{1}{2}\int_\Omega\int_\Omega W(x_1,x_2)p(x_1)p(x_2)dx_1dx_2.$$

In this case, the neural mapping gradient flow satisfies

$$\frac{d\theta}{dt}= \frac{1}{2}G_{\mathrm{W}}^{-1}(\theta)\ \ \mathsf{E}_{(z_1;z_2)\ \ p_{\mathrm{r}}\ \ p_{\mathrm{r}}}\Big[r\ \ W(f(\theta,z_1),f(\theta,z_2))\Big]. \tag{12}$$

In details,

$$\frac{d\theta_i}{dt}= \sum_{j=1}^{D}\Big(\mathsf{E}_{z\ \ p_{\mathrm{r}}}\Big[r\ \ f(\theta,z)\,r\ \ f(\theta,z)^{\mathsf{T}}\Big]\Big)^{-1}_{ij} \mathsf{E}_{(z_1;z_2)\ \ p_{\mathrm{r}}\ \ p_{\mathrm{r}}}\Big[r_{\ x_1}W(f(\theta,z_1),f(\theta,z_2))\ \ \partial_{\ j}f(\theta,z_1)\Big].$$

Example 6 (Neural projected negative entropy)**.** *Consider a negative entropy functional given by*

$$\mathsf{F}(p) = \int_{\Omega} U(p(x))dx.$$

In this case, the neural mapping gradient flow satisfies

$$\frac{d\theta}{dt} = -G_{\mathrm{W}}^{-1}(\theta)\ \nabla_{\theta}\mathsf{E}_{z\sim p_{\mathrm{r}}}\Big[\nabla_{\theta}\hat{U}(\frac{p_{\mathrm{r}}(z)}{\det(D_z f(\theta,z))})\Big], \tag{13}$$

where $\hat{U}(p) = U(p)/p$*. This is because:*

$$\begin{aligned}\mathsf{F}(f_{\theta\#}p_{\mathrm{r}}) &= \int_{\Omega} U(p(f(\theta,z)))df(\theta,z)\\ &= \int_{Z} U(\frac{p_{\mathrm{r}}(z)}{\det(D_z f(\theta,z))})\frac{\det(D_z f(\theta,z))}{p_{\mathrm{r}}(z)}p_{\mathrm{r}}(z)dz\\ &= \mathsf{E}_{z\sim p_{\mathrm{r}}}\Big[\hat{U}(\frac{p_{\mathrm{r}}(z)}{\det(D_z f(\theta,z))})\Big].\end{aligned}$$

The choice $U(p) = p\log(p)$ *and* $\hat{U}(p) = \log(p)$ *corresponds to the negative entropy. This belongs to the family of internal energy. In details,*

$$\begin{aligned}\frac{d\theta_i}{dt} = -\sum_{j=1}^{D}\Big(\mathsf{E}_{z\sim p_{\mathrm{r}}}\Big[\nabla_{\theta}f(\theta,z)\,\nabla_{\theta}f(\theta,z)^{\mathsf{T}}\Big]\Big)^{-1}_{ij}\\ \mathsf{E}_{z\sim p_{\mathrm{r}}}\Big[\ \mathrm{tr}\Big(D_z f(\theta,z)^{-1}\colon \partial_{\theta_j}D_z f(\theta,z)\Big)\hat{U}'(\frac{p_{\mathrm{r}}(z)}{\det(D_z f(\theta,z))})\frac{p_{\mathrm{r}}(z)}{\det(D_z f(\theta,z))}\Big].\end{aligned}$$

Here we denote $\mathrm{tr}(A\colon B) = \mathrm{tr}(AB)$*, for matrices* $A,\ B \in \mathsf{R}^{d\ \ d}$*.*

The above examples are projected Wasserstein gradient flows in neural mapping metric space. In particular, Examples 4, 5, 6 correspond to the following classical PDEs, respectively.

$$\partial_t p(t,x) = \nabla_x\cdot\Big(p(t,x)\,\nabla_x V(x)\Big), \tag{14}$$

$$\partial_t p(t,x) = \nabla_x\cdot\Big(p(t,x)\int_{\Omega}\nabla_x W(x,y)p(t,y)dy\Big), \tag{15}$$

$$\partial_t p(t,x) = \nabla_x\cdot\Big(p(t,x)\,\nabla_x U'(p(t,x))\Big). \tag{16}$$

The above dynamics include potential transport, interaction transport, and porous medium equations. The Fokker-Planck equation is a combination of the above first and third equations.

3.6. **Algorithm.** In this section, we discuss the implementations of gradient flows projected onto the parameter space. We apply the forward Euler discretization of the natural gradient flow (10). Let $h > 0$ be the step size. Then the update is given by

$$\theta^{k+1} = \theta^{k} - h\Big(\tilde{G}_{\mathrm{W}}(\theta^{k})\Big)^{-1}\nabla_{\theta}\tilde{F}(\theta^{k}), \tag{17}$$

where $\tilde{G}_{\mathrm{W}}(\theta) = (\tilde{G}_{\mathrm{W}}(\theta)_{ij})_{1 \ i;j \ D} \ 2 \ \mathsf{R}^{D \ D}$, $r \ \tilde{F}(\theta)$ are empirical estimates of the matrix G_{W} and the gradient $r F(\theta) = f\partial_{\ j} F(\theta) g_{j=1}^{D}$, respectively. In details, if $(z_l)_{l=1}^{M} \quad p_{\mathrm{r}}$, where M is the number of empirical samples, then

$$\tilde{G}_{\mathrm{W}}(\theta)_{ij} = \frac{1}{M} \sum_{l=1}^{M} \sum_{m=1}^{d} \partial_{\ i} f_m(z_l, \theta) \partial_{\ j} f_m(z_l, \theta) \, .$$

In practice, the condition number of $\tilde{G}_{\mathrm{W}}(\theta)$ could be very large and it is more stable to use instead the pseudoinverse of $\tilde{G}_{\mathrm{W}}(\theta)$ in (17). Therefore, the update is

$$\theta^{k+1} = \theta^{k} \quad h \tilde{G}_{\mathrm{W}}(\theta)^{y} r \ \tilde{F}(\theta^{k}) \, .$$

When the reference measure is a one-dimensional standard Gaussian distribution, $G_{\mathrm{W}}(\theta)$ can be explicitly computed for our choice of neural network. In this case, we have

$$\theta^{k+1} = \theta^{k} \quad h G_{\mathrm{W}}(\theta)^{y} r \ \tilde{F}(\theta^{k}) \, .$$

We summarize the above explicitly update formulas below.

Algorithm 1 Projected Wasserstein gradient flows

Input: Initial parameters $\theta \ 2 \ \mathsf{R}^{D}$; stepsize $h > 0$, total number of steps L, samples $f z_i g_{i=1}^{M} \quad p_{\mathrm{r}}$ for estimating $\tilde{G}_{\mathrm{W}}(\theta)$ and $r \ \tilde{F}(\theta)$.

for $k = 1, 2, \ldots, L$ **do**

$$\theta^{k+1} = \theta^{k} \quad h \tilde{G}_{\mathrm{W}}(\theta)^{y} r \ \tilde{F}(\theta^{k}); \quad \text{(when } G_{\mathrm{W}}(\theta) \text{ is unknown)}$$

or

$$\theta^{k+1} = \theta^{k} \quad h G_{\mathrm{W}}(\theta)^{y} r \ \tilde{F}(\theta^{k}); \quad \text{(when } G_{\mathrm{W}}(\theta) \text{ is known)}$$

end for

4. Numerical analysis on neural network projected gradient flows

In this section, we establish theoretical guarantees for the performance of the neural projected dynamics. We start by deriving an analytic formula for the inverse of the neural mapping metric of a special ReLU family in section 4.1. Based on the closed-form projected dynamics equations, we can establish the truncated error analysis for the projected dynamics in section 4.2. The analysis of truncated error for general dynamics is presented in section 4.3.

4.1. Analytic formula for the inverse of neural mapping metric.

In this section, we consider the following special case of the ReLU model in 1D. We first rewrite the neural network mapping function into the following form:

$$f(\theta, z) = \frac{1}{N} \sum_{i=1}^{N} a_i \sigma(z \quad b_i), \quad \sigma(z) = \begin{cases} 0, & z < 0, \\ z, & z \quad 0. \end{cases} \tag{18}$$

In particular, we combine a_i, b_i into one parameter in the 1D case. Under this reparameterization, a_is represent the slopes of each ReLU component and b_is are the intercepts. To make the last assumption on this ReLU network mapping function which facilitates the analytic formula of the neural mapping metric, we require all the slope parameters to stay non-negative, i.e. $a_i \quad 0$. Although this is an artificial assumption to enforce analyticity, it is natural in the sense that positive slope parameters induce monotone mapping function. Meanwhile, solutions of the Monge problems in 1D are known to be monotone. In section 4.1, we plot a typical ReLU mapping function.

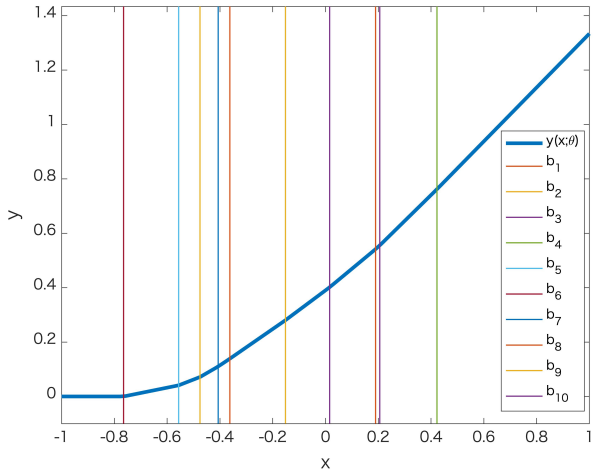


FIGURE 1. ReLU network mapping function considered in this section. The figure plots a typical monotone map parameterized by the ReLU network where the parameter a_i is required to be positive.

We start with the analytic formula for the neural mapping metric, assuming the reference measure is given by $p_{\mathrm{r}}(\ )$ with associated cumulative distribution function $\mathsf{F}_0(\ )$.

Proposition 4 (Neural mapping metric of two-layer ReLU network)**.** *The neural mapping metric of the two-layer ReLU network with reference measure $p_{\mathrm{r}}(\ )$ is given as*

$$
\begin{aligned}
G_{\mathrm{W}} &= \frac{1}{N^2}\begin{pmatrix} G_{\mathrm{W}}^{bb} & G_{\mathrm{W}}^{bw} \\ (G_{\mathrm{W}}^{bw})^T & G_{\mathrm{W}}^{ww} \end{pmatrix}, \\
G_{\mathrm{W}}^{bb} &= \begin{pmatrix} a_1^2(1 \quad \mathsf{F}_0(b_1)) & a_1a_2(1 \quad \mathsf{F}_0(b_2)) & \cdots & a_1a_N(1 \quad \mathsf{F}_0(b_N)) \\ a_1a_2(1 \quad \mathsf{F}_0(b_2)) & a_2^2(1 \quad \mathsf{F}_0(b_2)) & \cdots & a_2a_N(1 \quad \mathsf{F}_0(b_N)) \\ \vdots & \vdots & \ddots & \vdots \\ a_1a_{N\ \ 1}(1 \quad \mathsf{F}_0(b_{N\ \ 1})) & a_2a_{N\ \ 1}(1 \quad \mathsf{F}_0(b_{N\ \ 1})) & \cdots & a_Na_{N\ \ 1}(1 \quad \mathsf{F}_0(b_N)) \\ a_1a_N(1 \quad \mathsf{F}_0(b_N)) & a_2a_N(1 \quad \mathsf{F}_0(b_N)) & \cdots & a_N^2(1 \quad \mathsf{F}_0(b_N)) \end{pmatrix}, \\
G_{\mathrm{W}}^{ba} &= \begin{pmatrix} a_1\int_{b_1}^{1}(z \quad b_1)\,p_{\mathrm{r}}(z)dz & a_1\int_{b_2}^{1}(z \quad b_2)\,p_{\mathrm{r}}(z)dz & \cdots & a_1\int_{b_N}^{1}(z \quad b_N)\,p_{\mathrm{r}}(z)dz \\ a_2\int_{b_2}^{1}(z \quad b_1)\,p_{\mathrm{r}}(z)dz & a_2\int_{b_2}^{1}(z \quad b_2)\,p_{\mathrm{r}}(z)dz & \cdots & a_2\int_{b_N}^{1}(z \quad b_N)\,p_{\mathrm{r}}(z)dz \\ \vdots & \vdots & \ddots & \vdots \\ a_N\int_{b_N}^{1}(z \quad b_1)\,p_{\mathrm{r}}(z)dz & a_N\int_{b_N}^{1}(z \quad b_2)\,p_{\mathrm{r}}(z)dz & \cdots & a_N\int_{b_N}^{1}(z \quad b_N)\,p_{\mathrm{r}}(z)dz \end{pmatrix}, \\
(G_{\mathrm{W}}^{aa})_{ij} &= \int_{\max\{b_i;b_j\}}^{1}(z \quad b_j)\,(z \quad b_i)\,p_{\mathrm{r}}(z)dz.
\end{aligned}
\tag{19}
$$

Proof. We first calculate the derivative of the neural network map $f(\theta, z)$ w.r.t. network parameters θ

$$
\partial_{b_i} f(\theta,z) = \begin{cases} 0, & z < b_i, \\ \dfrac{a_i}{N}, & z > b_i, \end{cases} \qquad \partial_{a_i} f(\theta,z) = \frac{1}{N}\sigma(z \quad b_i), \tag{20}
$$

while the value at the singular point b_i does not exist and can be omitted from the measure-theoretical perspective. According to definition 4, one can evaluate different blocks of the metric tensor as the following integral

$$
\begin{aligned}
\left(G_{\mathrm{W}}^{bb}\right)_{ij} &= \int_{\mathsf{R}} \partial_{b_i} f(\theta,z)\partial_{b_j} f(\theta,z)p_{\mathrm{r}}(z)dz = \frac{a_ia_j}{N^2}(1 \quad \mathsf{F}_0(\max\{b_i,b_j\})), \\
\left(G_{\mathrm{W}}^{ba}\right)_{ij} &= \int_{\mathsf{R}} \partial_{b_i} f(\theta,z)\partial_{a_j} f(\theta,z)p_{\mathrm{r}}(z)dz = \frac{a_i}{N^2}\int_{\max\{b_i;b_j\}}^{1}(z \quad b_j)\,p_{\mathrm{r}}(z)dz, \\
(G_{\mathrm{W}}^{aa})_{ij} &= \int_{\mathsf{R}} \partial_{a_i} f(\theta,z)\partial_{a_j} f(\theta,z)p_{\mathrm{r}}(z)dz = \frac{1}{N^2}\int_{\max\{b_i;b_j\}}^{1}(z \quad b_j)\,(z \quad b_i)\,p_{\mathrm{r}}(z)dz.
\end{aligned}
$$

For general reference measure $p_{\mathrm{r}}(\ )$, the matrix elements of the $G_{\mathrm{W}}^{ba}, G_{\mathrm{W}}^{aa}$ relate to the first and second moments of the measure which may not have an analytic formula. Here, we consider a special neural mapping model with a Gaussian reference measure, thus rendering the metric with analytic elements.

Corollary 1. *With the same setting as proposition 4 and Gaussian reference measure, the matrix element of the neural mapping metric can be written analytically as*

$$\begin{aligned}\left(G_{\mathrm{W}}^{ba}\right)_{ij} &= p_{\mathrm{r}}(b_i) \quad b_j(1 \quad \mathsf{F}_0(b_i)), \quad b_i > b_j.\\ (G_{\mathrm{W}}^{aa})_{ij} &= b_i b_j(1 \quad \mathsf{F}_0(b_i)) \quad b_j p_{\mathrm{r}}(b_i) + (1 \quad \mathsf{F}_0(b_i)), \quad b_i > b_j.\end{aligned} \tag{21}$$

The other half of the elements can be obtained via switching b_i, b_j.

Proof. The proof is obtained by elementary integration calculation

$$\begin{aligned}&\int_{b_i}^{1} (z \quad b_j)\, p_{\mathrm{r}}(z)dz\\ &= p_{\mathrm{r}}(z)\Big|_{1}^{b_i} \quad b_j(1 \quad \mathsf{F}_0(b_i)) = p_{\mathrm{r}}(b_i) \quad b_j(1 \quad \mathsf{F}_0(b_i)),\\ &\int_{b_i}^{1} (z \quad b_j)\,(z \quad b_i)\, p_{\mathrm{r}}(z)dz\\ &= b_i b_j(1 \quad \mathsf{F}_0(b_i)) \quad (b_i + b_j)p_{\mathrm{r}}(b_i) + b_i p_{\mathrm{r}}(b_i) + (1 \quad \mathsf{F}_0(b_i))\\ &= b_i b_j(1 \quad \mathsf{F}_0(b_i)) \quad b_j p_{\mathrm{r}}(b_i) + (1 \quad \mathsf{F}_0(b_i)).\end{aligned} \tag{22}$$

Now, we focus on the upper right corner G_{W}^{bb} of the neural mapping metric. We will establish an analytical formula for the inverse of this matrix.

Theorem 2 (Analytic inverse of the neural mapping metric)**.** *The inverse matrix of the G_{W}^{bb} block in proposition 4 can be written analytically as*

$$\frac{1}{N^2}\left(G_{\mathrm{W}}^{\ 1}(\mathbf{b})\right)_{ij} = \begin{cases} \frac{1}{a_i^2}\left(\frac{1}{\mathsf{F}_0(b_i) \quad \mathsf{F}_0(b_{i\ 1})} + \frac{1}{\mathsf{F}_0(b_{i+1}) \quad \mathsf{F}_0(b_i)}\right), & i = j \ 6 = 1, N,\\ \frac{1}{a_i^2}\left(\frac{1}{\mathsf{F}_0(b_N) \quad \mathsf{F}_0(b_{N\ 1})} + \frac{1}{1 \quad \mathsf{F}_0(b_N)}\right), & i = j = N,\\ \frac{1}{a_i^2}\frac{1}{\mathsf{F}_0(b_2) \quad \mathsf{F}_0(b_1)}, & i = j = 1,\\ \frac{1}{a_i a_{i\ 1}}\frac{1}{\mathsf{F}_0(b_i) \quad \mathsf{F}_0(b_{i\ 1})}, & j = i \quad 1,\\ \frac{1}{a_i a_{i+1}}\frac{1}{\mathsf{F}_0(b_{i+1}) \quad \mathsf{F}_0(b_i)}, & j = i + 1,\\ 0, & o.w. \end{cases} \tag{23}$$

Proof. First, we decompose the neural mapping metric into the following matrix product

$$G_{\mathrm{W}} = \frac{1}{N^2} D \begin{pmatrix} 1 & \mathsf{F}_0(b_1) & 1 & \mathsf{F}_0(b_2) & & 1 & \mathsf{F}_0(b_N)\\ 1 & \mathsf{F}_0(b_2) & 1 & \mathsf{F}_0(b_2) & & 1 & \mathsf{F}_0(b_N)\\ & \vdots & & \vdots & \ddots & & \vdots\\ 1 & \mathsf{F}_0(b_N) & 1 & \mathsf{F}_0(b_N) & & 1 & \mathsf{F}_0(b_N) \end{pmatrix} D, \tag{24}$$

where $D = \mathrm{diag}(a_1, a_2, \quad , a_N)$ is a diagonal matrix. Then, it is direct to check that the middle matrix has the following tri-diagonal analytic inverse below:

$$\begin{pmatrix} \frac{1}{\mathsf{F}_0(b_2) \quad \mathsf{F}_0(b_1)} & \frac{1}{\mathsf{F}_0(b_2) \quad \mathsf{F}_0(b_1)} & 0 & & 0 \\ \frac{1}{\mathsf{F}_0(b_2) \quad \mathsf{F}_0(b_1)} & \frac{1}{\mathsf{F}_0(b_2) \quad \mathsf{F}_0(b_1)} + \frac{1}{\mathsf{F}_0(b_3) \quad \mathsf{F}_0(b_2)} & \frac{1}{\mathsf{F}_0(b_3) \quad F(b_2)} & & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & & \frac{1}{F(b_N) \quad F(b_{N \ 1})} + \frac{1}{1 \quad F(b_N)} \end{pmatrix}. \tag{25}$$

Multiplying this matrix with the inverse of the diagonal matrix D on both sides concludes this proof.

This analytic form of the inverse metric will be used intensively in the next subsection to prove the consistency of the numerical scheme based on the ReLU neural network.

4.2. **Truncated error analysis of the neural projected Wasserstein gradient flows based on analytic formula.** In this section, we perform the numerical analysis of the neural mapping projected Wasserstein flows introduced in section 3.5 based on the analytic formula in section 4.1. Because of the analytic inverse of the neural mapping metric, the right-hand side of the Wasserstein projected gradient flow can be calculated explicitly, and one can thus talk about its consistency and order of accuracy following the same spirit as classical numerical analysis. We perform this derivation for the Wasserstein projected gradient flows of the potential functional explicitly.

Let us first recall that the formula for neural projected Wasserstein gradient flow is given by

$$\frac{d\theta}{dt} = \quad G_{\mathrm{W}}^{1}(\theta) \quad \mathsf{E}_{\tilde{z} \quad p_{\mathrm{r}}} \Big[r \quad V(f(\theta, \tilde{z})) \Big]. \tag{26}$$

We have the following analytic formula for the projected gradient flow in the ReLU network model that we introduced in section 4.1.

Proposition 5 (Wasserstein gradient flow of potential functionals in ReLU network)**.** *The projected potential ow in the ReLU network model eq.* (18) *has the following form:*

$$\begin{aligned} \dot{b}_i &= \frac{N}{a_i} \left[\frac{\mathsf{E}_{z \quad p_{\mathrm{r}}}[V^{\prime 0}(f(b,z)) \mathbf{1}_{[b_i; b_{i+1}]}]}{\mathsf{F}_0(b_{i+1}) \quad \mathsf{F}_0(b_i)} \quad \frac{\mathsf{E}_{z \quad p_{\mathrm{r}}}[V^{\prime 0}(f(b,z)) \mathbf{1}_{[b_{i \ 1}; b_i]}]}{\mathsf{F}_0(b_i) \quad \mathsf{F}_0(b_{i \ 1})} \right], \quad i \neq 1, N, \\ \dot{b}_N &= \frac{N}{a_N} \left[\frac{\mathsf{E}_{z \quad p_{\mathrm{r}}}[V^{\prime 0}(f(b,z)) \mathbf{1}_{[b_N; 1)}]}{1 \quad \mathsf{F}_0(b_N)} \quad \frac{\mathsf{E}_{z \quad p_{\mathrm{r}}}[V^{\prime 0}(f(b,z)) \mathbf{1}_{[b_{N \ 1}; b_N]}]}{\mathsf{F}_0(b_N) \quad \mathsf{F}_0(b_{N \ 1})} \right], \\ \dot{b}_1 &= \frac{N}{a_1} \frac{\mathsf{E}_{z \quad p_{\mathrm{r}}}[V^{\prime 0}(f(b,z)) \mathbf{1}_{[b_1; b_2]}]}{\mathsf{F}_0(b_2) \quad \mathsf{F}_0(b_1)}. \end{aligned} \tag{27}$$

Using the trapezoid rule to calculate the integration gives the following spatial discretization, which can be used to simulate the projected gradient ow:

$$\dot{b}_i = \frac{N}{2a_i} \left(V^{\prime 0}(f(b, b_{i+1})) \quad V^{\prime 0}(f(b, b_{i \ 1})) \right). \tag{28}$$

Proof. It suffices to calculate the gradient of the linear potential functional in this model. Let us start with the calculation of the functional form of the potential energy in the ReLU

network mapping model as follows

$$\mathsf{E}_{x \sim f_{b\#}p_{\mathrm{r}}}[V(x)] = \mathsf{E}_{z \sim p_{\mathrm{r}}}[V(f(b,z))], \tag{29}$$

where we use the change of the integration variable above. Therefore, the gradient of this functional w.r.t. b can be simplified to

$$\partial_{b_i}\mathsf{E}_{z \sim p_{\mathrm{r}}}[V(f(b,z))] = \mathsf{E}_{z \sim p_{\mathrm{r}}}[\partial_{b_i}V(f(b,z))] = \frac{a_i}{N}\mathsf{E}_{z \sim p_{\mathrm{r}}}[V^{\prime}(f(b,z))\mathbf{1}_{[b_i,\,1)}(z)], \tag{30}$$

where we use $\mathbf{1}_A$ to denote the characteristic function on the interval A. Now, plugging this result into the projected gradient flow eq. (26) with the analytical formula for the inverse matrix G_{W}^{-1} in theorem 2, we obtain

$$\begin{aligned}
\dot{b_i} = &\frac{N^2}{a_i^2}\left(\frac{1}{\mathsf{F}_0(b_i) - \mathsf{F}_0(b_{i-1})} + \frac{1}{\mathsf{F}_0(b_{i+1}) - \mathsf{F}_0(b_i)}\right)\frac{a_i}{N}\mathsf{E}_{z \sim p_{\mathrm{r}}}[V^{\prime}(f(b,z))\mathbf{1}_{[b_i,\,1)}(z)] \\
&\frac{N^2}{a_i a_{i-1}}\frac{1}{\mathsf{F}_0(b_i) - \mathsf{F}_0(b_{i-1})}\frac{a_{i-1}}{N}\mathsf{E}_{z \sim p_{\mathrm{r}}}[V^{\prime}(f(b,z))\mathbf{1}_{[b_{i-1},\,1)}(z)] \\
&\frac{N^2}{a_i a_{i+1}}\frac{1}{\mathsf{F}_0(b_{i+1}) - \mathsf{F}_0(b_i)}\frac{a_{i+1}}{N}\mathsf{E}_{z \sim p_{\mathrm{r}}}[V^{\prime}(f(b,z))\mathbf{1}_{[b_{i+1},\,1)}(z)] \\
= &\frac{N}{a_i}\left[\frac{\mathsf{E}_{z \sim p_{\mathrm{r}}}[V^{\prime}(f(b,z))\mathbf{1}_{[b_i,b_{i+1}]}(z)]}{\mathsf{F}_0(b_{i+1}) - \mathsf{F}_0(b_i)} \quad \frac{\mathsf{E}_{z \sim p_{\mathrm{r}}}[V^{\prime}(f(b,z))\mathbf{1}_{[b_i,b_{i+1}]}(z)]}{\mathsf{F}_0(b_i) - \mathsf{F}_0(b_{i-1})}\right].
\end{aligned} \tag{31}$$

Taking a close look at the terms inside the brackets, one finds that they are calculating the average value of $V^{\prime}$ inside the intervals $[b_{i-1}, b_i], [b_i, b_{i+1}]$ weighted by the base distribution $p_{\mathrm{r}}(\,)$. Lastly, in order to complete the spatial discretization, one needs to choose a quadrature rule to calculate the integration in the above formula. One example is the trapezoid rule:

$$\mathsf{E}_{z \sim p_{\mathrm{r}}}[V^{\prime}(f(b,z))\mathbf{1}_{[b_i,b_{i+1}]}(z)] \quad (\mathsf{F}_0(b_{i+1}) - \mathsf{F}_0(b_i))\frac{V^{\prime}(f(b,b_i)) + V^{\prime}(f(b,b_{i+1}))}{2},$$

which provides the desired discretization. Special attention should be paid to the boundary node b_1, b_N to obtain their corresponding evolution equation and discretization.

Given this spatial discretization, we can analyze the order of consistency of it, which is treated in the following proposition.

Proposition 6 (Consistency of the projected gradient flow)**.** *Assume potential functional satisfies $\|V^{\prime\prime}\|_1 < 1$. The spatial discretization eq.* (28) *is of first-order accuracy both in the mapping and the density coordinates.*

Proof. We prove this statement from two directions, i.e. consistency in the space of mapping distribution and consistency in the space of mapping function. We have

$$\begin{aligned}
\partial_t f(b(t),z) &= \dot{b}^T \partial_b f(b,z) \\
&= \sum_{i=1}^{N} \frac{N}{2a_i}\left(V^0(f(b,b_{i+1})) \quad V^0(f(b,b_{i-1}))\right)\frac{a_i}{N}\mathbf{1}_{[b_i,b_{i+1})}(z) \\
&= \sum_{i=1}^{N} \frac{V^0(f(b,b_{i+1})) \quad V^0(f(b,b_{i-1}))}{2}\mathbf{1}_{[b_i,b_{i+1})}(z) \\
&= \frac{V^0(f(b,b_{i+1})) + V^0(f(b,b_i))}{2}, \quad z \in [b_i,b_{i+1}].
\end{aligned} \tag{32}$$

In the above derivation, we slightly cheat in the derivation so we can use the consistent formula for the evolution equations for all the nodes b_i. It is easy to conclude that our discretization corresponds to the evolution of the mapping function f of constant speed $\frac{V^0(f(b;b_{i+1}))+V^0(f(b;b_i))}{2}$ on each interval $[b_i, b_{i+1}]$. Now, recall that in mapping space, the Wasserstein gradient flow of the potential function $V(x)$ corresponds to the velocity field $V^0(x)$. Therefore, given that the length of each interval is of order Δb, we conclude that our spatial discretization is first order consistent on the mapping space.

Next, we prove the statement for the mapping distribution. To do this, we need to derive the evolution equation for the mapping distribution according to eq. (28). We have for $x \in [f(b,b_i), f(b,b_{i+1})]$

$$\begin{aligned}
&\partial_t p(t,x) = \dot{b}^T \partial_b p(t,x) \\
&= \sum_{i=1}^{N} \frac{N}{2a_i}\left(V^0(f(b,b_{i+1})) \quad V^0(f(b,b_{i-1}))\right)\frac{a_i}{N}\left(\partial_x p(t,x)\mathbf{1}_{[f(b;b_i),f(b;b_{i+1}))}(x) + p(t,x)\delta_{f(b;b_i)}(x)\right) \\
&= \partial_x p(t,x)\sum_{i=1}^{N} \frac{V^0(f(b,b_{i+1})) \quad V^0(f(b,b_{i-1}))}{2}\mathbf{1}_{[f(b;b_i),f(b;b_{i+1}))}(x) \\
&\quad + p(t,x)\sum_{i=1}^{N} \frac{V^0(f(b,b_{i+1})) \quad V^0(f(b,b_{i-1}))}{2}\delta_{f(b;b_i)}(x) \\
&= \partial_x p(t,x)\frac{V^0(f(b,b_{i+1})) + V^0(f(b,b_i))}{2} \quad p(t,x)\frac{V^0(f(b,b_{i+1})) + V^0(f(b,b_{i-1}))}{2}\delta_{f(b;b_i)}(x).
\end{aligned} \tag{33}$$

A quick method to derive the formula of $\partial_b p(t,x)$ is to view it as a probability flow corresponds to the cotangent vector $\partial_b f$ and then use the Wasserstein metric to calculate via a continuity equation as in proposition 2. Recall that the potential gradient flow in the density manifold is given by

$$\partial_t p(t,x) = \nabla \cdot (p(t,x)\nabla V(x)) = \partial_x p(t,x)\partial_x V(x) + \partial_{xx} p(t,x)V(x). \tag{34}$$

Comparing eq. (33) and eq. (34), it is not difficult to recognize that the first term in eq. (33) approximates the continuous counterpart in eq. (34) in the first order. The remaining parts correspond to each other: the approximation is first order not in the strong sense, but

rather in the weak sense as there is Dirac measure in eq. (33). Combining the above two parts, we finish the proof.

4.2.1. *Projected dynamics of Negative entropy gradient flow.* The potential functional can be viewed as a linear functional whose projected gradient flow has a rather simple expression. The corresponding formula has a more complex expression for general nonlinear internal energy, such as entropy. We begin with calculating the negative entropy functional of a neural mapping measure $f_{\#}p_{\mathrm{r}}$:

$$\begin{aligned} H\left(f_{\#}p_{\mathrm{r}}\right) &= \mathsf{E}_{x\ \ cont\left(f_{\#}p_{\mathrm{r}}\right)}\left[\log f_{\#}p_{\mathrm{r}}\left(x\right)\right] + \mathsf{F}_0\left(b_1\right)\log \mathsf{F}_0\left(b_1\right) \\ &= \mathsf{E}_{z\ \ cont(p_{\mathrm{r}})}\left[\log f_{\#}p_{\mathrm{r}}\left(f\left(z\right)\right)\right] + \mathsf{F}_0\left(b_1\right)\log \mathsf{F}_0\left(b_1\right) \\ &= \mathsf{E}_{z\ \ cont(p_{\mathrm{r}})}\left[\log \frac{p_{\mathrm{r}}\left(z\right)}{f^{\emptyset}\left(z\right)}\right] + \mathsf{F}_0\left(b_1\right)\log \mathsf{F}_0\left(b_1\right) \\ &= \mathsf{E}_{z\ \ cont(p_{\mathrm{r}})}\left[\log p_{\mathrm{r}}\left(z\right)\right] \quad \mathsf{E}_{z\ \ cont(p_{\mathrm{r}})}\left[\log f^{\emptyset}\left(z\right)\right] + \mathsf{F}_0\left(b_1\right)\log \mathsf{F}_0\left(b_1\right), \end{aligned} \tag{35}$$

where we use the Monge-Ampère equation $f_{\#}p_{\mathrm{r}}\left(f\left(z\right)\right) = \frac{p_{\mathrm{r}}(z)}{f^{\emptyset}(z)}$ in one dimension. Moreover, notice that the last term corresponds to the entropy of the discrete part of distribution $f_{\#}p_{\mathrm{r}}$ as the ReLU mapping function maps $(\quad 1, b_1]$ to 0 and $cont(\,)$ refers to the continuous part of a distribution. Similarly, the relative entropy functional is given by

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}\left(f_{\#}p_{\mathrm{r}}\middle\|\nu\right) = \quad & \mathsf{E}_{z\ \ cont(p_{\mathrm{r}})}\left[\log p_{\mathrm{r}}\left(z\right)\right] \quad \mathsf{E}_{z\ \ cont(p_{\mathrm{r}})}\left[\log f^{\emptyset}\left(z\right)\right] \\ & \mathsf{E}_{z\ \ cont(p_{\mathrm{r}})}\left[\log \nu\left(f\left(z\right)\right)\right] + \mathsf{F}_0\left(b_1\right)\left(\log \mathsf{F}_0\left(b_1\right)\right). \end{aligned}$$

Moreover, the gradient flow of the KL-divergence differs from that of negative entropy only by a term that appears in the derivation in the potential functional gradient flow. This This also manifests in calculus on the density manifold between the heat and Fokker-Planck equations. Now, one calculates the derivative of continuous parts w.r.t. parameter b_i

$$\begin{aligned} \partial_{b_i}\mathsf{E}_{x\ \ p_{\mathrm{r}}}\left[\log f^{\emptyset}\left(x\right)\right] &= \begin{cases} \log \dfrac{\sum_{j=1}^{i\ \ 1} a_j}{\sum_{j=1}^{i} a_j} p_{\mathrm{r}}\left(b_i\right), & i \neq 1, \\ p_{\mathrm{r}}\left(b_1\right)\dfrac{\log a_1}{N}, & i = 1. \end{cases} \\ \partial_{b_i}\mathsf{E}_{x\ \ p_{\mathrm{r}}}\left[\log \nu\left(f\left(x\right)\right)\right] &= \mathsf{E}_{x\ \ p_{\mathrm{r}}}\left[\frac{\nu^{\emptyset}\left(y\left(x\right)\right)\partial_{b_i}y\left(x\right)}{\nu\left(y\left(x\right)\right)}\right]. \end{aligned}$$

The first derivation is based on the observation that the function $\log f^{\emptyset}\left(x\right)$ is a step function which changes its value at b_i. It takes value $\log \sum_{j=1}^{i\ \ 1} a_j$ at interval $[b_{i\ \ 1}, b_i]$. Hence the desired conclusion follows, where $p_{\mathrm{r}}\left(b_i\right)$ comes in since this is the expectation w.r.t. distribution $p_{\mathrm{r}}\left(x\right)$. Therefore, the derivative of the entropy and relative entropy functional

reads as follows

$$\partial_{b_i} H\left(f_{\#} p_{\mathrm{r}}\right)=\begin{cases}\log \dfrac{\sum_{j=1}^{i} a_j}{\sum_{j=1}^{i-1} a_j} p_{\mathrm{r}}\left(b_i\right), & i \geq 1, \\ p_{\mathrm{r}}\left(b_1\right)\left(\log \mathsf{F}_0\left(b_1\right)+1+\log \dfrac{a_1}{N}\right), & i=1.\end{cases} \tag{36}$$

$$\partial_{b_i} \mathrm{D}_{\mathrm{KL}}\left(f_{\#} p_{\mathrm{r}} \| \nu\right)=\mathsf{E}_{x \sim p}\left[\frac{\nu^{\prime}\left(f\left(x\right)\right) \partial_{b_i} f\left(x\right)}{\nu\left(f\left(x\right)\right)}\right] \quad \log \frac{\sum_{j=1}^{i} a_j}{\sum_{j=1}^{i-1} a_j} p_{\mathrm{r}}\left(b_i\right).$$

With all these preparations, we can write out the gradient flow equation of the entropy functional:

$$\begin{aligned}
\dot{b_i} = & \frac{1}{\mathsf{F}_0\left(b_i\right)-\mathsf{F}_0\left(b_{i-1}\right)}\left(\frac{\log \frac{\sum_{j=1}^{i-1} a_j}{\sum_{j=1}^{i} a_j} p_{\mathrm{r}}\left(b_i\right)}{a_i^2} \quad \frac{\log \frac{\sum_{j=1}^{i-2} a_j}{\sum_{j=1}^{i-1} a_j} p_{\mathrm{r}}\left(b_{i-1}\right)}{a_i a_{i-1}}\right) \\
& +\frac{1}{\mathsf{F}_0\left(b_{i+1}\right)-\mathsf{F}_0\left(b_i\right)}\left(\frac{\log \frac{\sum_{j=1}^{i-1} a_j}{\sum_{j=1}^{i} a_j} p_{\mathrm{r}}\left(b_i\right)}{a_i^2} \quad \frac{\log \frac{\sum_{j=1}^{i} a_j}{\sum_{j=1}^{i+1} a_j} p_{\mathrm{r}}\left(b_{i+1}\right)}{a_i a_{i+1}}\right), \quad i=2, \quad, N-1, \\
\dot{b_1} = & \frac{1}{a_1\left(\mathsf{F}_0\left(b_2\right)-\mathsf{F}_0\left(b_1\right)\right)}\left(\frac{p_{\mathrm{r}}\left(b_1\right)\left(\log \mathsf{F}_0\left(b_1\right)+1+\log \frac{a_1}{N}\right)}{a_1}+\frac{\log \frac{\sum_{j=1}^{1} a_j}{\sum_{j=1}^{2} a_j} p_{\mathrm{r}}\left(b_2\right)}{a_2}\right), \\
\dot{b_N} = & \frac{\log \frac{\sum_{j=1}^{N-1} a_j}{\sum_{j=1}^{N} a_j} p_{\mathrm{r}}\left(b_N\right)}{a_N^2\left(1-\mathsf{F}_0\left(b_N\right)\right)} \quad \frac{1}{\mathsf{F}_0\left(b_N\right)-\mathsf{F}_0\left(b_{N-1}\right)}\left(\frac{\log \frac{\sum_{j=1}^{N-2} a_j}{\sum_{j=1}^{N-1} a_j} p_{\mathrm{r}}\left(b_{N-1}\right)}{a_N a_{N-1}} \quad \frac{\log \frac{\sum_{j=1}^{N-1} a_j}{\sum_{j=1}^{N} a_j} p_{\mathrm{r}}\left(b_N\right)}{a_N^2}\right).
\end{aligned} \tag{37}$$

Similar to the proof in proposition 6, one can carefully expand the neural projected dynamics of the entropy functional and prove that it converges to the heat equation in the limit that number of neurons tends to infinity and the gap between neurons nodes tends to zero.

4.2.2. *Analysis of the long-time existence of the neural-projected heat flow.* In general, the projected Wasserstein gradient flow does not necessarily need to be a linear dynamics even though the original gradient flow is linear, e.g., the projected gradient flow corresponding to the heat equation is highly nonlinear. This poses great difficulties in analyzing and establishing the long-time existence of the projected dynamics, as mentioned in [25]. Specifically, we focus on the nonlinear projected gradient flow of the entropy, which corresponds to the Heat equation in the full space. If we view all nodes $b_i, i \in [N]$ as grid points and view the scheme as an example of the moving mesh method [17], then the mesh quality is an important quantity to observe during simulation. One does not want the mesh quality to decrease too much and even become degenerate during the simulations. Therefore, we consider the well-posedness of the non-linear ODE eq. (37).

Proposition 7. *The neural projected dynamics eq.* (37) *of the heat flow is well-posed, e.g. the solution extends to arbitrary time.*

Proof. We consider a special scenario when two adjacent nodes b_i, b_{i+1} become close to each other while maintaining a relatively large gap with all other nodes, i.e.

$$o(1) = b_{i+1} - b_i = o(b_p - b_q), \quad \forall q \in [N] \setminus \{i, i+1\}, \quad p = i, i+1. \tag{38}$$

WLOG, we assume $b_{i+1} = b_i + \Delta b > b_i$ and reduce the following term which appears both in their time derivative expression in eq. (37)

$$\begin{aligned}
&\frac{1}{\mathsf{F}_0(b_i) - \mathsf{F}_0(b_{i-1})} \begin{pmatrix} \dfrac{\log \frac{\sum_{j=1}^{i-1} a_j}{\sum_{j=1}^{i} a_j} p_{\mathrm{r}}(b_i)}{a_i^2} & \dfrac{\log \frac{\sum_{j=1}^{i-2} a_j}{\sum_{j=1}^{i-1} a_j} p_{\mathrm{r}}(b_{i-1})}{a_i a_{i-1}} \end{pmatrix} \\
= &\left(\frac{1}{p_{\mathrm{r}}(b_i)\Delta b} + O(1)\right) \left(\log \frac{i-1}{i} p_{\mathrm{r}}(b_i) \quad \log \frac{i-2}{i-1} p_{\mathrm{r}}(b_{i-1})\right) \\
= &\left(\frac{1}{p_{\mathrm{r}}(b_i)\Delta b} + O(1)\right) \left(\log \frac{i-1}{i} p_{\mathrm{r}}(b_i) \quad \log \frac{i-2}{i-1} \left(p_{\mathrm{r}}(b_i) + O(\Delta b)\right)\right) \\
= &\left(\frac{1}{p_{\mathrm{r}}(b_i)\Delta b} + O(1)\right) \left(\log \frac{i^2-2i+1}{i^2-2i} p_{\mathrm{r}}(b_i) + O(\Delta b)\right) \\
= &\frac{1}{\Delta b} \log \frac{i^2-2i+1}{i^2-2i} + O(1) \to +\infty, \quad \Delta b \to 0^+,
\end{aligned} \tag{39}$$

where we use the simplified model where all the weights a_i are set to 1 and Taylor expansion to conclude that $\mathsf{F}_0(b_i) - \mathsf{F}_0(b_{i-1}) = p_{\mathrm{r}}(b_i)\Delta b + O(\Delta b^2)$ and $p(b_{i-1})$ follows the same spirit. This term appears with positive sign in the RHS of $\dot{b}_i$ and negative sign in the RHS of $\dot{b}_{i-1}$, indicating that the left (right) node $b_{i-1}(b_i)$ will move fast towards left (right) respectively. This repulsion behavior guarantees that the Lagrangian coordinates will never collide with each other and the mesh degeneracy will not appear.

Next, we analyze our scheme using the time derivative of the Lagrangian coordinate. It is a well-known result that under the heat flow the mean of the distribution is fixed. Therefore, due to the diffusive nature of the heat equation, one can imagine that the position of the quantile greater than the mean should move right in the heat equation and vice versa. Suppose $x \in [b_i, b_{i+1}]$ is a quantile with b_i greater than the mean 0. As the base measure is a standard Gaussian distribution whose probability density function decreases over $[0, +\infty)$, we conclude that

$$0 < b_i < b_{i+1} \Longrightarrow p_{\mathrm{r}}(b_i) > p_{\mathrm{r}}(b_{i+1}) \Longrightarrow \log \frac{\sum_{j=1}^{i-1} a_j}{\sum_{j=1}^{i} a_j} p_{\mathrm{r}}(b_i) < \log \frac{\sum_{j=1}^{i} a_j}{\sum_{j=1}^{i+1} a_j} p_{\mathrm{r}}(b_{i+1}) < 0. \tag{40}$$

Consequently, the Lagrangian coordinate $f_b(z)$ is indeed moving towards right, which matchs the intuition from the heat equation.

The neural projected dynamics can be understood as a Lagrangian scheme [8, 24, 26] with neural network basis. Specifically, fixing basis as ReLU components in eq. (18), one can view a_i's and b_i's as the shape and location coefficients of the basis functions respectively. Updating a_i's is similar to classical finite-element method with fixed basis functions, while adding the degree of freedom of b_i's is similar to the moving mesh method. The Lagrangian schemes can handle the problem of the free boundary such as porous medium, e.g. in [24], they use finite element method to solve the mapping function of the porous medium equation with high accuracy. While most Lagrangian schemes are based on updating the a_i's parameters, our methods have more flexibility and expressivity as it takes more degree of freedom into account. The primal-dual structure of the Wasserstein gradient flow also leverages a lot of usage of Lagrangian schemes [8].

On the other hand, our numerical algorithm and the moving mesh method. The principal ingredients of the moving mesh method include the equidistribution principle, the moving mesh equation, and the method of lines approach [36]. The moving mesh equation is solved during the simulation to ensure the adaptivity such that the mesh can resolve to the detailed structure. In many classical moving mesh methods, the mesh equations are solved separately from the governing PDE itself to guarantee the adaptivity of the numerical methods. This implies that how the mesh change will not depend explicitly on the underlying PDE. There also exist moving mesh methods such that the mesh updates take into account of the governing PDE (e.g., the arbitrary Lagrangian-Eularian methods [5]). From this perspective, the projected dynamics provide a PDE-specific moving mesh equation, i.e. the mesh moved according to the PDE dynamics to simulate which is more adaptive and efficient. Moreover, through a detailed study of the simple case, we can establish a theoretical guarantee on the quality of our moving mesh method in proposition 7.

4.3. **Truncated error analysis for general neural projected Wasserstein gradient flow.** The proof of the consistency of the numerical scheme relies on the analytic formula derived before which is restrictive. In this section, we provide another methodology to prove the consistency of the numerical scheme we derived in this paper. Instead of calculating the evolution of the mapping explicitly, we calculate the deviation of the projected gradient w.r.t. the original gradient direction. Let us first state a geometric proposition where we attempt to be as general as possible. This result is also proved in [25] and we prove it here for completeness.

Let X be a manifold (possibly infinite-dimensional) with a Riemannian metric g_{X}, which provides an inner product on the tangent space $T_x\mathsf{X}$ (possibly infinite-dimensional Hilbert space) for each $x \in \mathsf{X}$. Let $\mathsf{Y} \subset \mathsf{X}$ be its submanifold with induced metric denoted by g_{Y}, i.e. $\forall y \in \mathsf{Y}$:

$$g_{\mathsf{Y}}(y) : T_y\mathsf{Y} \times T_y\mathsf{Y} \to \mathsf{R}, \quad g_{\mathsf{Y}}(y)(v, w) = g_{\mathsf{X}}(y)(v, w), \quad \forall v, w \in T_y\mathsf{Y}.$$

Furthermore, let $H : \mathsf{X} \to \mathsf{R}$ be a functional defined over X and we use $\widetilde{H} : \mathsf{Y} \to \mathsf{R}$ for its restriction on Y. We have the following proposition.

Proposition 8. *Let* $\nabla_{g_{\mathcal{X}}} H(y) \in T_y\mathcal{X}$ ($\nabla_{g_{\mathcal{Y}}} \widetilde{H}(y) \in T_y\mathcal{Y}$) *denote the gradient of the functional* H *w.r.t. the metric* $g_{\mathcal{X}}$ ($g_{\mathcal{Y}}$) *at* $y \in \mathcal{X}$ ($y \in \mathcal{Y}$). *Then, we have*

$$\nabla_{g_{\mathcal{Y}}} \widetilde{H}(y) = \Pi(y) \nabla_{g_{\mathcal{X}}} H(y), \tag{41}$$

where $\Pi(y)$ *is the orthogonal projection operator from* $T_y\mathcal{X}$ *to* $T_y\mathcal{Y}$.

Proof. As $\mathcal{Y}$ is a submanfold of $\mathcal{X}$, we have inclusion map $\mathrm{I}(y) : T_y\mathcal{Y} \to T_y\mathcal{X}$ and restriction map $\mathrm{I}^*(y) : T_y\mathcal{X} \to T_y\mathcal{Y}$ for each $y \in \mathcal{Y}$. Both mappings are linear and are adjoint to each other. Therefore, viewing the metric tensor $g_{\mathcal{Y}}(y)$ as a linear mapping between $T_y\mathcal{Y} \to T_y\mathcal{Y}$, we have

$$g_{\mathcal{Y}}(y) = \mathrm{I}^*(y) \circ g_{\mathcal{X}}(y) \circ \mathrm{I}(y), \quad \forall y \in \mathcal{Y}.$$

Moreover, the inner product $g_{\mathcal{X}}(y)$ on the Hilbert space $T_y\mathcal{X}$ induces an orthogonal decomposition:

$$T_y\mathcal{X} = T_y\mathcal{Y} \oplus T_y\mathcal{Y}^{\perp}, \quad \forall y \in \mathcal{Y},$$

along with an orthogonal projection operator $\Pi(y)$. Now, recall that the Riemannian gradient $\nabla_{g_{\mathcal{X}}} H(y)$ is defined as

$$g_{\mathcal{X}}(y) \nabla_{g_{\mathcal{X}}} H(y) = dH(y).$$

The differential of $H(\,)$ and $\widetilde{H}(\,)$ is related by

$$d\widetilde{H}(y) = \mathrm{I}^*(y) dH(y), \quad \forall y \in \mathcal{Y}.$$

Therefore, gathering all the ingredients, we have the following commutative diagram

$$\begin{array}{ccc} & T_y\mathcal{Y} \underset{p_{\mathrm{r}}(y)}{\overset{\mathrm{I}(y)}{\rightleftarrows}} T_y\mathcal{X} & \ni \nabla_{g_{\mathcal{X}}} H(y) \\ & {\scriptstyle g_{\mathcal{Y}}(y)}\downarrow \qquad\qquad \downarrow{\scriptstyle g_{\mathcal{X}}(y)} & \\ d\widetilde{H}(y) \in & T_y\mathcal{Y} \xleftarrow{\mathrm{I}^*(y)} T_y\mathcal{X} & \ni dH(y) \end{array}$$

As $\Pi(y)$ is the orthogonal projection, we conclude that

$$\nabla_{g_{\mathcal{Y}}} \widetilde{H}(y) = (\mathrm{I}^*(y) g_{\mathcal{X}}(y) \mathrm{I}(y))^{-1} \mathrm{I}^*(y) dH(y) = \Pi(y) \nabla_{g_{\mathcal{X}}} H(y).$$

We can prove the consistency of our numerical schemes over different PDEs with the Wasserstein gradient flow structures by leveraging this proposition in the case $\mathcal{X} = P_2^{\mathcal{M}}(\mathcal{R})$ is the density manifold and $g_{\mathcal{X}}$ is chosen to be the W_2 metric. To achieve this, we can rewrite eq. (41) as

$$\nabla_{g_{\mathcal{Y}}} \widetilde{H}(y) = \mathrm{argmin}_{v \in T_y\mathcal{Y}} \| \nabla_{g_{\mathcal{X}}} H(y) - v \|_{g_{\mathcal{X}}}. \tag{42}$$

Therefore, $\forall v \in T_y\mathcal{Y}$ will provide an upper bound for the truncated error of our approximation scheme. Moreover, if we assume that the submanifold $\mathcal{Y} \subset P_2^{\mathcal{M}}(\mathcal{R})$ is identical to a generative model via mapping function $f_{\#}$, i.e. $\mathcal{Y} = f_{\#} p_{\mathrm{r}}$ with $\theta \in \Theta$ and p_{r} the base measure. Then, the projected gradient direction can also be characterized using the metric over the mapping space, i.e.

$$\nabla_{\Theta} \widetilde{H}(\theta) = \mathrm{argmin}_{v \in T_\theta \Theta} \int (\nabla H(\theta)(x) - v(x))^2 f_{\#} p_{\mathrm{r}}(x) dx, \tag{43}$$

where θ is mapped to point y and we abuse the notion of $\nabla_\Theta H(\theta)$ to denote the gradient vector in the mapping coordinate. Moreover, we can perform truncated error analysis directly over the mapping space, which is more convenient and clear. Let us focus on the ReLU network mapping eq. (18). The tangent space in the mapping coordinate is spanned by the vectors in eq. (20). Meanwhile, the tangent space in the density coordinate is spanned by

$$\partial_{b_i} f_{\#}\rho_{\mathrm{r}}(x) = \frac{a_i}{N}\rho_{\mathrm{r}}^{(0)}(x)\mathbf{1}_{[f(\cdot;b_i);1)}, \quad \partial_{a_i} f_{\#}\rho_{\mathrm{r}}(x) = \frac{f^{-1}(x) - b_i}{N}\rho_{\mathrm{r}}^{(0)}(x)\mathbf{1}_{[f(\cdot;b_i);1)}, \tag{44}$$

where the notation $f(\cdot) = f(\theta, \cdot)$. Here we use the fact that the mapping f is linear with slope $\frac{\sum_{j=1}^{i} a_j}{N}$ over the interval $[b_i, b_{i+1}]$. If b_is are fixed, the projected dynamics belongs to projection-based model reduction [6] where the basis is fixed to be neurons. While changing b_is correspond to model reduction with adaptive basis.

Proposition 9. *The numerical scheme based on ReLU network mapping is consistent with order 2 using both a, b parameters and of order 1 with either a or b parameters.*

Proof. In view of eq. (20), we have that the approximation using only $\partial_{b_i} f$ is simply piecewise constant approximation. As each ingredient has the shape of a Heaviside function, it is consistent with order 1. While the approximation using both $\partial_{b_i} f$ and $\partial_{a_i} f$ is a piece-wise linear approximation, thereby consistent of order 2. This is because another set of ReLU-shape functions is added to the basis.

The connection between the ReLU neural network and the linear finite element space is systematically studied in [15]. They theoretically establish that at least two hidden layers are needed in a ReLU neural network to represent any linear finite element functions in $\Omega \subset \mathbb{R}^d$ when $d \geq 2$.

Based on this concrete understanding of the structure of the tangent space, we can calculate the local truncation error of the projected gradient flow.

Theorem 3. *Given a tangent vector $v(x) \in T_{f_{\theta\#}\rho_{\mathrm{r}}}\mathcal{P}(\mathbb{R})$ whose approximated tangent vector in projected dynamics is given by $-\nabla H(\theta)$, the local truncation error in the ReLU network mapping is given by*

$$\sum_{i=1}^{N} \int_{b_i}^{b_{i+1}} v^2(f(z))\rho_{\mathrm{r}}(z)dx \quad \frac{\left(\int_{b_i}^{b_{i+1}} v(f(z))(z - m_i)\rho_{\mathrm{r}}(z)dz\right)^2}{\int_{b_i}^{b_{i+1}}(z - m_i)^2\rho_{\mathrm{r}}(z)dz} \quad \frac{\left(\int_{b_i}^{b_{i+1}} v(f(z))\rho_{\mathrm{r}}(z)dz\right)^2}{F_0(b_{i+1}) - F_0(b_i)} \tag{45}$$

where m_i is the center of mass of $\rho_{\mathrm{r}}(z)$ in $[b_i, b_{i+1}]$ and b_{N+1} is understood as $+1$. Under the assumption that v has bounded second order derivative and $b_{i+1} - b_i < \Delta b, \forall i$.

$$\|v(x) - \nabla H(\theta)\|^2_{L^2(f_{\theta\#}\rho_{\mathrm{r}})} = \frac{1}{4}\left(\frac{\sum_{j=1}^{N} a_j}{N}\right)^2 \left\|v^{(0)}\right\|_1 O(\Delta b^4). \tag{46}$$

Proof. As mentioned in the above theorem, the approximation using ReLU network mapping is equivalent to piecewise linear approximation in the mapping coordinate. Moreover, at each node b_i, the slope and value of the The function does not need to be continuous,

which is exactly the same as the linear spline interpolation. The main difference is that the grid points b_i are not fixed since they can evolve over time. Therefore, we rewrite the optimization problem eq. (43) as

$$\arg\min_{c_i,d_i} \quad \sum_{i=1}^{N} \int_{f_\theta(b_i)}^{f_\theta(b_{i+1})} (v(x) \quad c_i x \quad d_i)^2 \, f \, {}_{\#}p_{\mathrm{r}}(x)dx, \tag{47}$$

which can be further reduced to $N \quad 1$ separated optimization problem of c_i, d_i over small interval $[f\ (b_i), f\ (b_{i+1})]$. For each subproblem, we have

$$\int_{f_\theta(b_i)}^{f_\theta(b_{i+1})} (v(x) \quad c_i x \quad d_i)^2 \, f \, {}_{\#}p_{\mathrm{r}}(x)dx = \int_{b_i}^{b_{i+1}} (v(f\ (z)) \quad c_i f\ (z) \quad d_i)^2 \, p_{\mathrm{r}}(z)dz.$$

This is a quadratic optimization problem of c_i, d_i with positive definite Hessian matrix. Taking derivative w.r.t. c_i, d_i, we obtain

$$\begin{aligned} \int_{b_i}^{b_{i+1}} (v(f\ (z)) \quad c_i f\ (z) \quad d_i)\, p_{\mathrm{r}}(z)dz = 0, \\ \int_{b_i}^{b_{i+1}} f\ (z)\,(v(f\ (z)) \quad c_i f\ (z) \quad d_i)\, p_{\mathrm{r}}(z)dz = 0. \end{aligned}$$

Now, using the fact that $f\ (z)$ is a linear function over the interval $[b_i, b_{i+1}]$, we have

$$c_i f\ (z) + d_i = \frac{\int_{b_i}^{b_{i+1}} v(f\ (z))(z \quad m_i)p_{\mathrm{r}}(z)dx}{\int_{b_i}^{b_{i+1}} (z \quad m_i)^2 p_{\mathrm{r}}(z)dz}(z \quad m_i) + \frac{\int_{b_i}^{b_{i+1}} v(f\ (z))p_{\mathrm{r}}(z)dx}{\mathsf{F}_0(b_{i+1}) \quad \mathsf{F}_0(b_i)}. \tag{48}$$

Plugging back, we obtain the approximation error as

$$\begin{aligned} &\int_{b_i}^{b_{i+1}} (v(f\ (z)) \quad c_i f\ (z) \quad d_i)^2 \, p_{\mathrm{r}}(z)dz \\ &= \int_{b_i}^{b_{i+1}} v(f\ (z))\,(v(f\ (z)) \quad c_i f\ (z) \quad d_i)\, p_{\mathrm{r}}(z)dz \\ &= \int_{b_i}^{b_{i+1}} v^2(f\ (z))p_{\mathrm{r}}(z)dz \quad \frac{\left(\int_{b_i}^{b_{i+1}} v(f\ (z))(z \quad m_i)p_{\mathrm{r}}(z)dz\right)^2}{\int_{b_i}^{b_{i+1}} (z \quad m_i)^2 p_{\mathrm{r}}(z)dz} \quad \frac{\left(\int_{b_i}^{b_{i+1}} v(f\ (z))p_{\mathrm{r}}(z)dz\right)^2}{\mathsf{F}_0(b_{i+1}) \quad \mathsf{F}_0(b_i)}. \end{aligned} \tag{49}$$

Next, we assume all the intervals $[b_i, b_{i+1}]$ are short (of scale $O(\Delta)$) and consider expanding the v as Taylor series around m_i, i.e.

$$\begin{aligned} v(f\ (z)) = v(f\ (m_i)) + \frac{\sum_{j=1}^{i} a_j}{N} v^{\prime}(f\ (m_i))(z \quad m_i) \\ + \frac{1}{2}\left(\frac{\sum_{j=1}^{i} a_j}{N}\right)^2 v^{\prime\prime}(f\ (m_i))(z \quad m_i)^2 + O(\Delta^3). \end{aligned} \tag{50}$$

Here, we use the fact that $f\ (z)$ is a linear function with slope $\frac{\sum_{j=1}^{i} a_j}{N}$ over the interval $[b_i, b_{i+1}]$. Plugging into eq. (48), we obtain

$$\begin{aligned} c_i f\ (z) \quad d_i = & \frac{\sum_{j=1}^{i} a_j}{N} v^{\prime}(f\ (m_i))(z \quad m_i) + v(f\ (m_i)) \\ & + \frac{1}{2}\left(\frac{\sum_{j=1}^{i} a_j}{N}\right)^2 v^{\prime\prime}(f\ (m_i)) \frac{\int_{b_i}^{b_{i+1}} (z \quad m_i)^2 p_{\mathrm{r}}(z) dz}{\mathsf{F}_0(b_{i+1}) \quad \mathsf{F}_0(b_i)} \\ & + \frac{1}{2}\left(\frac{\sum_{j=1}^{i} a_j}{N}\right)^2 v^{\prime\prime}(f\ (m_i)) \frac{\int_{b_i}^{b_{i+1}} (z \quad m_i)^3 p_{\mathrm{r}}(z) dz}{\int_{b_i}^{b_{i+1}} (z \quad m_i)^2 p_{\mathrm{r}}(z) dz} (z \quad m_i) + O(\Delta^3). \end{aligned} \tag{51}$$

Notice that the first two terms are exactly the zero-th and first order term of the $v(f\ (z))$ function which is similar to classical linear function approximation by discarding all the higher order term. The appearance of residue terms is due to approximation in $L^2(p)$ sense. To calculate the L^2-approximation error, we have

$$\begin{aligned} & \left[\frac{1}{2}\left(\frac{\sum_{j=1}^{i} a_j}{N}\right)^2 v^{\prime\prime}(f\ (m_i))\right]^2 \\ & \int_{b_i}^{b_{i+1}} \left((z \quad m_i)^2 \quad \frac{\int_{b_i}^{b_{i+1}} (z \quad m_i)^2 p_{\mathrm{r}}(z) dz}{\mathsf{F}_0(b_{i+1}) \quad \mathsf{F}_0(b_i)} \quad \frac{\int_{b_i}^{b_{i+1}} (z \quad m_i)^3 p_{\mathrm{r}}(z) dz}{\int_{b_i}^{b_{i+1}} (z \quad m_i)^2 p_{\mathrm{r}}(z) dz} (z \quad m_i) + O(\Delta^3)\right)^2 p_{\mathrm{r}}(z) dz \\ = & \left[\frac{1}{2}\left(\frac{\sum_{j=1}^{i} a_j}{N}\right)^2 v^{\prime\prime}(f\ (m_i))\right]^2 O((b_{i+1} \quad b_i)^5) p_{\mathrm{r}}(b_i) + O((b_{i+1} \quad b_i)^6) p_{\mathrm{r}}(b_i). \end{aligned} \tag{52}$$

In summary, the L^2 approximation error consists of the sum over all the interval $[b_i, b_{i+1}]$, with each term depends on a_i through the factor $\frac{\sum_{j=1}^{i} a_j}{N}$, on b_i through $(b_{i+1} \quad b_i)^5$ and the term $v^{\prime\prime}(f\ (m_i))$, which also contains a_i, b_i.

Let us calculate a special case of the Fokker-Planck equation

$$\partial_t p(t,x) \quad r \quad (p(t,x)\, r\, V(x)) \quad \gamma \Delta p(t,x) = 0.$$

Under the Wasserstein metric, the tangent vector in the mapping space is given by

$$v(x) = \quad V^{\prime}(x) \quad \gamma \frac{p^{\prime}(t,x)}{p(t,x)}.$$

In this case, we have that

$$v^{\prime\prime}(x) = \quad V^{(3)}(x) \quad \gamma \frac{p^{(3)}(t,x) p(t,x)^2 + 2 p^{\prime}(t,x)^3 \quad 3 p(t,x) p^{\prime}(t,x) p^{\prime\prime}(t,x)}{p(t,x)^3}.$$

The above function will determine the approximation quality of the projected dynamics.

Remark 1. The high-order neural mapping function class and associated high-order projected dynamics can also be derived following a similar procedure. For example, we can

add a quadratic term of the ReLU function into the network mapping function as

$$f(\theta, z) = \frac{1}{N} \sum_{i=1}^{N} a_i \sigma (z \quad b_i) + c_i \sigma^2 (z \quad b_i) . \tag{53}$$

Notice that adding high order ReLU term is different from increasing the layers in the ReLU neural network which corresponds to function composition. We leave the detailed analysis and numerical experiments on high-order methods in future work.

5. Numerical Examples

In this section, we provide several numerical experiments to test our algorithm and theories. We focus our attention on the linear transport equation, Fokker-Planck equation, porous medium equations, and Keller-Segel equation. They all correspond to some specific energy functionals in the probability space equipped with the Wasserstein-2 distance.

5.1. **Neural Network structure.**

We first describe the structure of our neural network for the experiment. We focus on two-layer neural network with ReLU as activation functions.

$$f(\theta, z) = \sum_{i=1}^{N} a_i \quad \sigma(z \quad b_i) + \sum_{i=N+1}^{2N} a_i \quad \sigma(b_i \quad z) . \tag{54}$$

Here $\theta \ 2 \ \mathsf{R}^{4N}$ represents the collection of weights $\mathsf{f}a_i\mathsf{g}_{i=1}^{2N}$ and bias $\mathsf{f}b_i\mathsf{g}_{i=1}^{2N}$. To simplify our notation, we have absorbed the $1/N$ factor into a_i's. At initialization, we set $a_i = 1/N$ for $i \ 2 \ \mathsf{f}1, \dots, N\mathsf{g}$ and $a_i = \quad 1/N$ for $i \ 2 \ \mathsf{f}N + 1, \dots, 2N\mathsf{g}$. To choose the b_i's, we first set $\mathbf{b} = \text{linspace}(\quad B, B, N)$ for some positive constant B (e.g. $B = 4$ or $B = 10$). We then set $b_i = \mathbf{b}[i]$ for $i = 1, \dots, N$ and $b_j = \mathbf{b}[j \quad N] + \varepsilon$ for $j = N + 1, \dots, 2N$. Here $\varepsilon = 5 \quad 10^{\ 6}$ is a small offset which will be explained later in Section 5.3. Our initialization is chosen such that $f(\theta, \ )$ approximates the identity map at initialization. In practice, we find it beneficial to perform a rescaling of the weights a_i's. We replace a_i with $\overline{a}_i/\beta$ for some fixed constant $\beta > 0$. And we initialize $\overline{a}_i = \beta/N$ for $i \ 2 \ \mathsf{f}1, \dots, N\mathsf{g}$ and $\overline{a}_i = \quad \beta/N$ for $i \ 2 \ \mathsf{f}N + 1, \dots, 2N\mathsf{g}$. This rescaling makes sure that $f(\theta, \ )$ still approximates the identity map at initialization. We provide a brief intuition for rescaling. Let us consider $g(a, b, z) = a \quad \sigma(b \quad z)$ for $b = \mathsf{O}(1)$, $z = \mathsf{O}(1)$, $a = \mathsf{O}(1/N)$. Then $\partial_a g = \sigma(z \quad b) = \mathsf{O}(1)$. On the other hand, $\partial_b g = a \quad \sigma^{\emptyset}(b \quad z) = \mathsf{O}(1/N)$. This simple calculation shows that the partial gradient of (54) with respect to weights and bias are of different scales. Therefore, to make them the same scale, a natural choice is choosing $\beta = \mathsf{O}(N)$.

Remark 2. The choice of neural network (54) is slightly more complicated than the one studied in Section 4. This symmetric structure is used in numerical experiments to overcome ReLU's drawback such that only the positive input is activated. Moreover, (54) allows us to construct an approximation to the identity map over R easily. However, the results of Proposition 9 still hold for (54). And Theorem 3 can be generalized to neural network of the form given in (54) in a straightforward manner. The metric tensor G_{W} is now a $4N \quad 4N$ matrix. The calculations of the individual components of G_{W} follow the same procedure presented in proposition 4.

Remark 3. We remind our readers that our algorithm takes the form of

$$\theta^{k+1} = \theta^k \quad hG_{\mathrm{W}}(\theta)^{y}\, r \;\; \tilde{F}(\theta^k)\,.$$

During implementation, $r \;\; \tilde{F}(\theta)$ can be obtained by backpropagating $\tilde{F}(\theta)$ in the case of Example 4 and Example 5. However, we need to pay special attention to $\partial_{b_i} F(\theta)$ when dealing with Example 6. This will be elaborated further in Section 5.3 and Section 5.4.

5.2. **Linear transport PDE.** We investigate the linear transport PDE given by Eq. (14) with several choices of potential $V(x)$, corresponding to the gradient flow of them under the Wasserstein metric. For a simple potential function, this example can serve as a sanity check of the projected dynamics formulation. The trajectories of the particles for Eq. (14) (i.e. Lagrangian formulation) follows the following ODE

$$\dot{x}(t) = \quad r\, V(x)\,. \tag{55}$$

Let us denote by $T(t, z_0)$ the solution to Eq. (55) with initial condition $x(0) = z_0$. In other words, $T(t, z_0)$ is the transport map at time t starting from position z_0. We define the error at time t by

$$\begin{aligned}\text{error} = \int_{1}^{1} & \big| f(\theta_t, z_0) \quad T(t, z_0)\big| p_0(z_0)\, dz_0 \\ & \frac{1}{N_1}\sum_{j=1}^{N_1} \big| f(\theta_t, z_j) \quad T(t, z_j)\big| p_0(z_j)\,,\end{aligned} \tag{56}$$

where we discretize the integration domain by N_1 equally spaced points to approximate the integral. And $p_0(z_0)$ denotes the initial distribution of z_0. Below we test our projected dynamics under three choices of potential functions and investigate the convergence behavior of two projected dynamics: (i) fixing the bias terms b_i and only updating the weights a_i and (ii) updating both bias b_i and weights a_i. Note that when the bias terms b_i's are fixed, we have that $G_{\mathrm{W}}\ 2\ \mathsf{R}^{2N\ \ 2N}$. Recall that we are essentially simulating the gradient flow on parameter θ_t given by Eq. (11). We use $M = 5 \quad 10^5$ particles sampled from a standard Gaussian distribution for approximating $\mathsf{E}_{\tilde{z}\ \ p_{\mathrm{r}}}\Big[V(f(\theta, \tilde{z}))\Big]$. Once we have the empirical loss function

$$\mathsf{E}_{\tilde{z}\ \ p_{\mathrm{r}}}\Big[V(f(\theta, \tilde{z}))\Big] \quad \frac{1}{M}\sum_{i=1}^{M} V(f(\theta, z_i))\,,$$

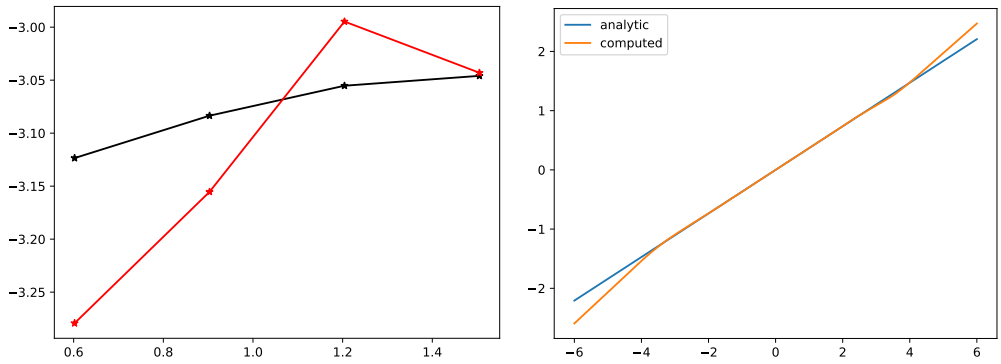
we can backpropagate this loss to obtain

$$\mathsf{E}_{\tilde{z}\ \ p_{\mathrm{r}}}\Big[\, r\ \ V(f(\theta, \tilde{z}))\Big] \quad \frac{1}{M}\sum_{i=1}^{M} r\ \ V(f(\theta, z_i))\,,$$

which will be used in the update of θ_t given by Eq. (11).

5.2.1. *Quadratic potential.* As the first example for linear transport PDE, we consider the quadratic potential $V(x) = \frac{1}{2}(x \quad \mu_0)^2$ as a sanity check. The stationary distribution will be the delta mass supported at μ_0. Using the method of characteristics, one can show that the solution at time $t > 0$ is given by

$$p(t, x) = p_0\big((x \quad \mu_0)e^t + \mu_0\big)e^t\,, \tag{57}$$



(A) Error (B) Mapping comparison

FIGURE 2. Left: log-log plot of linear transport PDE with a quadratic potential. The y-axis represents $\log_{10}$ error defined by (56). x-axis represents $\log_{10}(N)$. The bias terms b_i are initialized based on Section 5.1 with $B = 4$. Red line represents results when only the weights terms are updated. Black line represents results when both weights and bias are updated. Right: Mapping comparison between $T(t, z)$ given by Eq. (58) and our computed solution $f(\theta_t, z)$.

where $p_0(x) = p(0, x)$ is the initial distribution. In Lagrangian coordinates, the transport map of a point z_0 at time t is given by

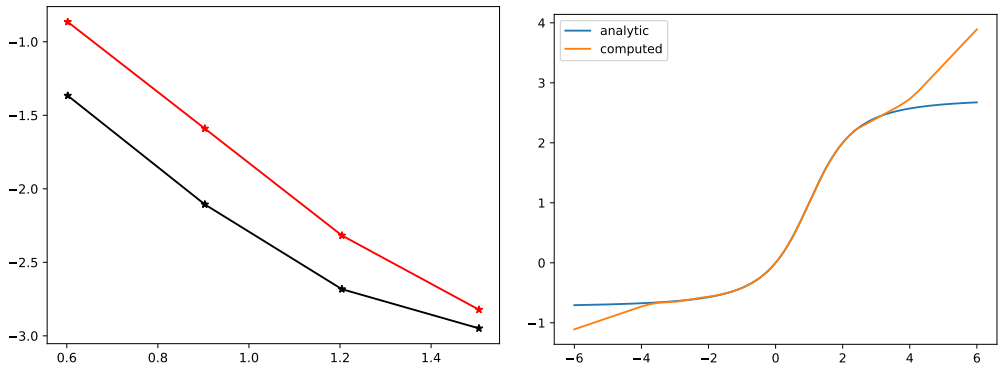
$$T(t, z_0) = \mu_0 + e^{\ t}(z_0 \quad \mu_0)\,. \tag{58}$$

One can check that $T(z_0, 0) = z_0$ and $T(t, z_0) \ ! \ \mu_0$ as $t \ ! \ 1$. It is worthwhile mentioning that at each $t > 0$, the Lagrangian map $x_t(z_0) : z_0 \ 7! \ T(t, z_0)$ is a *linear map*. For simplicity, we take $\delta_0 = 0$. We choose $dt = 10^{\ 3}$ and run for 1000 steps. We compare our numerical results with Eq. (58). The result is demonstrated in Fig. 2. In Fig. 2b, we have provided a visualization of the analytic solution to the linear transport PDE in Lagrangian coordinates at $t = 1$ and our computed solution. As shown in Fig. 2b, the analytic transport map is linear while the neural mapping function is piecewise linear. Increasing N does not necessarily give a smaller approximation error. In fact, we see in Fig. 2a that larger N usually gives a larger error, commonly known as overfitting in machine learning.

5.2.2. *Quartic potential.* Let us consider $V(x) = (x \quad 1)^4/4 \quad (x \quad 1)^2/2$. The analytic solution of the transport map is given by

$$T(t, z_0) = \begin{cases} \operatorname{sgn}(z_0 \quad 1)\dfrac{e^t}{\sqrt{(z_0 \quad 1)^{\ 2} + e^{2t} \quad 1}} + 1, & z_0 \ 6= 1\,, \\ 1, & z_0 = 1\,. \end{cases} \tag{59}$$

Basic settings are the same as the previous case. We choose $dt = 2 \quad 10^{\ 4}$ and run for 1000 steps. We compare our numerical results with Eq. (59). We present our results in Fig. 3. In Fig. 3a, we observe a clear decrease in error as the number of neurons becomes larger. In Fig. 3b, we visualize the analytic solution to the linear transport PDE in Lagrangian



(A) Error (B) Mapping comparison

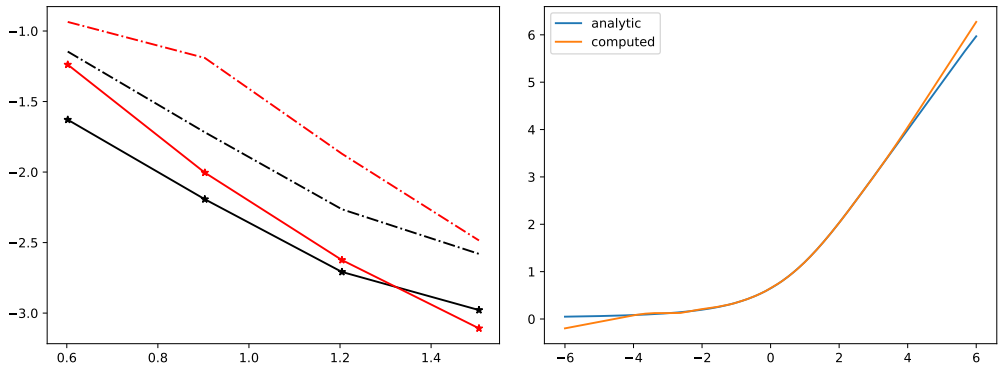
FIGURE 3. Left: log-log plot of linear transport PDE with quartic polynomial potential. The y-axis represents $\log_{10}$ error defined by (56). x-axis represents $\log_{10}(N)$. The bias terms b_i are initialized based on Section 5.1 with $B = 4$. Red line represents results when only the weights terms are updated. Black line represents results when both weights and bias are updated. Right: Mapping comparison between $T(t, z)$ given by Eq. (59) and our computed solution $f(\theta_t, z)$.

coordinates at $t = 0.2$ and our computed solution. We can see that even when the optimal transport map is nonlinear, our computed solution still matches the analytic solution very accurately.

5.2.3. *Sixth order polynomial potential.* Let us consider $V(x) = (x \quad 4)^6/6$. The analytic solution of the transport map is given by

$$T(t, z_0) = \begin{cases} 4 + \operatorname{sgn}(z_0 \quad 4)\dfrac{1}{\sqrt{2\sqrt{\frac{1}{4(z_0 \quad 4)^4} + t}}}, & z_0 \neq 4\,, \\ 4, & z_0 = 4\,. \end{cases} \tag{60}$$

We choose $dt = 10^{\ 6}$ and run for 1000 steps. The reason to choose such a small step size is that the ODE (55) is stiff when $V(x)$ is a sixth order polynomial. This can be readily seen by considering the forward Euler scheme for solving (55), which results in the popular gradient descent algorithm. The step size that can guarantee convergence in gradient descent is at most $2/L$ where L is the Lipschitz constant of the gradient function. In our case, the gradient function $rV(x)$ is not globally Lipschitz. Even if we consider a fixed interval $(\ l, l)$, the Lipschitz constant is $L = 5(l+4)^4$. If we take $l = 10$, then we get $L = O(10^{\ 6})$. We compare our numerical results with Eq. (60). We have chosen $fz_j g_{j=1}^{N_1}$ to be a uniform mesh of size $N_1 = 4 \quad 10^6$ on $[\ 6, 6]$ in Eq. (56) and p_0 is the standard Gaussian distribution. Note that N_1 is chosen to be much larger than the number of neurons N in the network mapping function as it is used to evaluate the accuracy of our algorithm. We present our results in Fig. 4. We can see a clear decrease in error when N increases from Fig. 4a. It is also clear from Fig. 4a that updating both weights and



(A) Error (B) Mapping comparison

FIGURE 4. Left: log-log plot of linear transport PDE with sixth order polynomial potential. The y-axis represents $\log_{10}$ error defined by (56). x-axis represents $\log_{10}(N)$. The bias terms b_i are initialized based on Section 5.1 with $B = 10$ for dashed line and $B = 4$ for solid line. Red lines represent results when only the weights terms are updated. Black lines represent results when both weights and bias are updated. Right: Mapping comparison between $T(t, z)$ given by Eq. (60) and our computed solution $f(\theta_t, z)$.

bias tends to have a smaller error than just updating the weights, although the difference becomes smaller when N increases and more mesh points become available. Comparing dashed and solid lines in Fig. 4a, we find that the initialization of b_i also plays a role in the overall performance of our solution. The error is smaller when the initial mesh points (i.e., the b_i's) are more concentrated near the center of the reference measure. In our case, the reference measure is a standard Gaussian, whose measure is "almost" supported on $[\ 4, 4]$. Hence we see that the solid lines show a smaller error than the dashed lines in Fig. 4a. In Fig. 4b, we have given a visualization of the analytic solution to the linear transport PDE in Lagrangian coordinates at $t = 10^{\ 3}$ and our computed solution. It is worth noting from Fig. 4b that our learned Lagrangian map approximates the analytic Lagrangian map well near the center of the reference distribution, which is concentrated near the origin. Even though the error of the learned Lagrangian map is larger outside of $[\ 4, 4]$, the overall error from Eq. (56) is still small since the reference measure (standard Gaussian measure) on $\mathsf{R}\,n\,[\ 4, 4]$ is exponentially small.

Remark 4. According to Proposition 9, updating both a_i and b_i is a second order method. This can be seen from Fig. 4a when N is small. When N is large, the numerical advantage of updating both a_i and b_i is less significant compared with updating only a_i. This is partially explained by the condition number of the $G_{\mathrm{W}}(\theta)$ grows too large when θ contains all of a_i and b_i. This phenomenon is also observed in our other experiments. Using the implicit scheme or proximal scheme (without solving the linear system that involves $G_{\mathrm{W}}(\theta)$ directly) might help with this difficulty, which we leave as a future study.

5.3. **Fokker-Planck Equation.** We consider Fokker-Planck equations. In general, there is no closed-form solution for either the Eulerian or Lagrangian coordinate except for some special forms of potential V (e.g. quadratic). We can still have an approximation of the analytic transport map by realizing that the optimal transport map of a point z_0 at time t is given by

$$T(t, z_0) = \mathsf{F}_t\big(\mathsf{F}_0{}^{1}(z_0)\big) \tag{61}$$

where F_t is the cumulative distribution function (CDF) of $p(t, x)$. F_0 has a closed form expression when we choose our reference measure to be a standard Gaussian. But we still need to know $p(t, x)$. Therefore, to investigate the performance of our algorithm, we need to use a numerical solver to solve for $p(t, x)$. We choose a center difference in space, implicit in time discretization as our choice of numerical solver with vanishing boundary condition. Recall that we are essentially simulating the gradient flow on parameter θ_t given by Eq. (11) and Eq. (13). To calculate the derivative of the energy functionals, we used $M = 10^6$ particles sampled from a standard Gaussian distribution for approximating $\mathsf{E}_{z\ \ p_{\mathrm{r}}}\Big[V(f(\theta, z)) + \hat{U}(\frac{p_{\mathrm{r}}(z)}{D_z f(\ ;z)})\Big]$. Approximating $\mathsf{E}_{z\ \ p_{\mathrm{r}}}\Big[\ r\ \ V(f(\theta, z))\Big]$ is straightforward and has been explained in detail in Section 5.2. On the other hand, some care needs to be taken when approximating $\mathsf{E}_{z\ \ p_{\mathrm{r}}}\Big[\ r\ \ \hat{U}(\frac{p_{\mathrm{r}}(z)}{D_z f(\ ;z)})\Big]$ as explained in Section 4.2.1. Suppose that all of the $\mathsf{f}b_k\mathsf{g}_{k=1}^{2N}$ are different. Take $2 \quad j \quad N$. Let us also assume that the b_k's are ordered so that $b_1 \quad b_2 \qquad b_N$.

$$\begin{aligned}\mathsf{E}_{z\ \ p_{\mathrm{r}}}\partial_{b_j}\log(D_z f(\theta, z)) &= \mathsf{E}_{z\ \ p_{\mathrm{r}}}\partial_{b_j}\log\left(\sum_{i=1}^{N} a_i \mathbf{1}_{[b_i;\ 1\ )}(z) \quad \sum_{i=N+1}^{2N} a_i \mathbf{1}_{(\ \ 1\ ;b_i]}(z)\right) \\ &= p_{\mathrm{r}}(b_j)\log\left(\frac{\sum_{i=1}^{j}{}^{1} a_i \quad \sum_{i=N+1}^{2N} a_i \mathbf{1}_{(\ \ 1\ ;b_i]}(b_j)}{\sum_{i=1}^{j} a_i \quad \sum_{i=N+1}^{2N} a_i \mathbf{1}_{(\ \ 1\ ;b_i]}(b_j)}\right).\end{aligned} \tag{62}$$

And

$$\mathsf{E}_{z\ \ p_{\mathrm{r}}}\partial_{b_1}\log(D_z f(\theta, z)) = p_{\mathrm{r}}(b_1)\log\left(\frac{\sum_{i=N+1}^{2N} a_i \mathbf{1}_{(\ \ 1\ ;b_i]}(b_1)}{a_1 \quad \sum_{i=N+1}^{2N} a_i \mathbf{1}_{(\ \ 1\ ;b_i]}(b_1)}\right). \tag{63}$$

Similarly, if we assume that $b_{N+1} \quad b_{N+2} \qquad b_{2N}$ and let $N + 2 \quad j \quad 2N$, we have

$$\begin{aligned}\mathsf{E}_{z\ \ p_{\mathrm{r}}}\partial_{b_j}\log(D_z f(\theta, z)) &= \mathsf{E}_{z\ \ p_{\mathrm{r}}}\partial_{b_j}\log\left(\sum_{i=1}^{N} a_i \mathbf{1}_{[b_i;\ 1\ )}(z) \quad \sum_{i=N+1}^{2N} a_i \mathbf{1}_{(\ \ 1\ ;b_i]}(z)\right) \\ &= p_{\mathrm{r}}(b_j)\log\left(\frac{\sum_{i=1}^{N} a_i \mathbf{1}_{[b_i;\ 1\ )}(b_j) \quad \sum_{i=N+1}^{j} a_i}{\sum_{i=1}^{N} a_i \mathbf{1}_{[b_i;\ 1\ )}(b_j) \quad \sum_{i=N+1}^{j}{}^{1} a_i}\right).\end{aligned} \tag{64}$$

And

$$\mathsf{E}_{z\ \ p_{\mathrm{r}}}\partial_{b_{N+1}}\log(D_z f(\theta, z)) = p_{\mathrm{r}}(b_{N+1})\log\left(\frac{\sum_{i=1}^{N} a_i \mathbf{1}_{[b_i;\ 1\ )}(b_{N+1}) \quad a_{N+1}}{\sum_{i=1}^{N} a_i \mathbf{1}_{[b_i;\ 1\ )}(b_{N+1})}\right). \tag{65}$$

Note that during implementation, we do not have to order the b_j's in order to evaluate the above partial derivatives. Let $0 < \delta \quad \frac{1}{2}\min_{i\neq j} \mathsf{j}b_i \quad b_j\mathsf{j}$. Then by a straightforward

calculation, we have

$$\mathsf{E}_{z\ \ p_{\mathrm{r}}}\partial_{b_j}\log(D_z f(\theta,z)) = \begin{cases} p_{\mathrm{r}}(b_j)\log\left(\frac{D_z f(\ ;b_j\ \ )}{D_z f(\ ;b_j)}\right), & 1\quad j\quad N\,. \\ p_{\mathrm{r}}(b_j)\log\left(\frac{D_z f(\ ;b_j)}{D_z f(\ ;b_{j+}\ )}\right), & N+1\quad j\quad 2N\,. \end{cases} \tag{66}$$

In our experiment, we set $\delta = \varepsilon/2$ where ε is the small offset we introduced in Section 5.1 during initialization.

5.3.1. *Quadratic potential.* As a first example for the Fokker-Planck equation, we use the quadratic potential as a sanity check. Here $V(x)$ is chosen to be a quadratic function. This is one of the rare cases where the Fokker-Planck equation has a closed-form analytic solution. In Lagrangian coordinates, the trajectories of the particles follow the following SDE, commonly known as the Ornstein-Uhlenbeck process:

$$dX_t = \quad \gamma_0(X_t \quad \mu_0)dt + \sigma_0 dW_t\,. \tag{67}$$

The corresponding Langevin equation for the density $p(t,x)$ is given by

$$\frac{\partial p}{\partial t} = \gamma_0\frac{\partial}{\partial x}\big((x \quad \mu_0)p\big) + D\frac{\partial^2 p}{\partial x^2}\,, \tag{68}$$

where $D = \sigma_0^2/2$. It can be shown that the solution to (68) is given by

$$p(t,x) = \sqrt{\frac{\gamma_0}{2\pi D(1 \quad \mathrm{e}^{\ 2\ 0t})}}\int_{\ 1}^{1}\exp\Big(\quad \frac{\gamma_0}{2D}\frac{(x \quad \mu_0 \quad x^{0}\mathrm{e}^{\ \ 0t})^2}{1 \quad \mathrm{e}^{\ 2\ 0t}}\Big)p_0(x^{0})\,\mathrm{d}x^{0}\,, \tag{69}$$

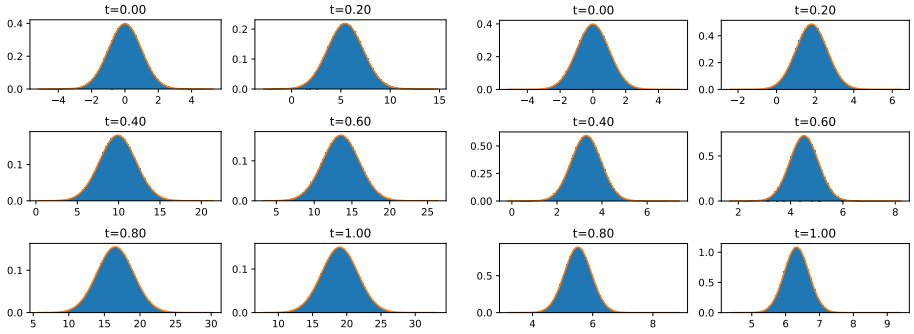
where $p_0(x) = p(0,x)$ is the initial distribution. In our experiment, $p_0(x)$ is a standard Gaussian. Then (69) implies that $p(t,x)$ is also Gaussian with mean $\mu_0(1 \quad e^{\ \ 0t})$ and variance $e^{\ 2\ 0t} + \frac{D(1\ \ e^{\ 2\gamma_0 t})}{0}$. Then the transport map is given by the optimal transport map between two Gaussians, which has a closed form expression. In this example, the transport map is

$$T(t,z) = \mu_0(1 \quad e^{\ \ 0t}) + z\sqrt{e^{\ 2\ 0t} + D(1 \quad e^{\ 2\ 0t})/\gamma_0}\,, \tag{70}$$

which is always a linear map, no matter the choice of μ_0, γ_0 and D. We use $M = 10^6$ particles sampled from a standard Gaussian distribution for approximating $\mathsf{E}_{z\ \ p_{\mathrm{r}}}\Big[\ r\ \ V(f(\theta,z)) + r\ \ \hat{U}(\frac{p_{\mathrm{r}}(z)}{D_z f(\ ;z)})\Big]$. We choose $dt = 10^{\ \ 3}$ and run for 1000 steps. We used a neural network with $m = 32$ and $B = 4$ following the setup in Section 5.1. We have the following two choices of parameters corresponding to different dynamics.

> *Moving and widening Gaussian.* We choose $\gamma_0 = 1$, $\mu_0 = 30$, $\sigma_0 = 4$. Under this setting, the solution at time t is a Gaussian distribution with mean $30(1 \quad e^{\ \ t})$ and variance $e^{\ \ 2t} + 8(1 \quad e^{\ \ 2t})$. This evolution is shown on the left panel of Fig. 5.
> *Moving and shrinking Gaussian.* We choose $\gamma_0 = 1$, $\mu_0 = 10$, $\sigma_0 = 0.01$. Under this setting, the solution at time t is a Gaussian distribution with mean $10(1 \quad e^{\ \ t})$ and variance $e^{\ \ 2t} + 5 \quad 10^{\ \ 5}(1 \quad e^{\ \ 2t})$. This evolution is shown on the right panel of Fig. 5.

Our results are demonstrated in Fig. 5. As shown in Fig. 5, the computed density closely follows the analytic density of the Fokker-Planck equation from $t = 0$ to $t = 1$.



(A) Moving, widening Gaussian (B) Moving, shrinking Gaussian

FIGURE 5. Density evolution of Eq. (68). Orange curve represents the solution given by Eq. (69). Blue rectangles represent the histogram using 10^6 particles in 100 bins from $t = 0$ to $t = 1$. Left panel: a Gaussian distribution shifting to the right with increasing variance. Right panel: a Gaussian distribution shifting to the right with decreasing variance.

5.3.2. *Quartic potential.* We consider $V(x) = (x \quad 1)^4/4 \quad (x \quad 1)^2/2$. We choose $dt = 2 \quad 10^{\ 4}$ and run for 1000 steps. We compare our numerical results with the transport map computed from Eq. (61). The results are shown in Fig. 6. In Fig. 6a, we observe a clear decrease in error when the number of neurons increases. In Fig. 6b, we plot a comparison between our computed Lagrangian map $f(\theta, z)$ vs the transport map computed from Eq. (61) using a numerical solver. The evolution of the density is demonstrated in Fig. 6c from $t = 0$ to $t = 0.2$.

5.3.3. *Sixth order polynomial potential.* We consider $V(x) = (x \quad 4)^6/6$. We choose $dt = 10^{\ 6}$ and run for 1000 steps. We compare our numerical results with the transport map computed from Eq. (61). The results are shown in Fig. 7. We have observed similar behavior as in the case of linear transport PDE: the error becomes smaller when N increases. Moreover, comparing dashed and solid lines in Fig. 7a we see that as the initial mesh points (i.e. the b_i's) concentrate nearer the center of our reference measure, the errors are smaller. In Fig. 7b we show a comparison between Lagrangian maps computed by our method and the numerical solver. We have also plotted the evolution of the density in Fig. 7c from $t = 0$ to $t = 10^{\ 3}$.

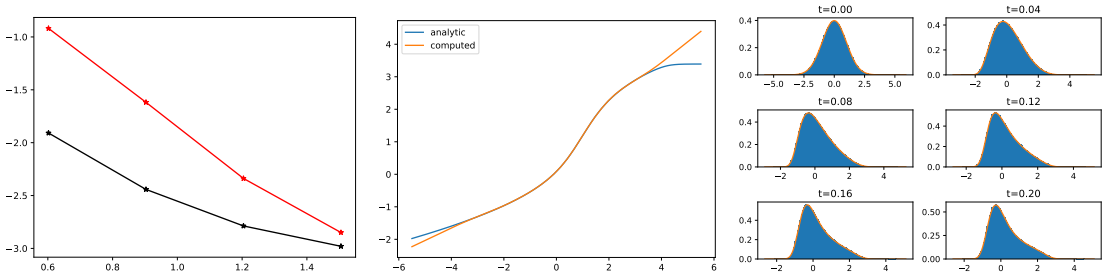
5.4. **Porous Medium Equation.** We consider Example 6 with the functional $U(p(x)) = \frac{1}{m \ 1} p(x)^m$, $m > 1$. This choice of U yields the porous medium equation

$$\partial_t p(t,x) = \Delta p(t,x)^m \,. \tag{71}$$

It is known that Eq. (71) admits solutions given by the Barenblatt profile

$$p(t,x) = (t_0 + t) \quad \left(C \quad \beta k x k^2 (t_0 + t)^{\ 2 \ =d}\right)_+^{\frac{1}{m \ 1}}, \tag{72}$$

where $x \ 2 \ \mathsf{R}^d$, $\alpha = \frac{d}{d(m \ 1)+2}$, $\beta = \frac{(m \ 1)}{2dm}$, $t_0 > 0$ and C is a normalization constant so that Eq. (72) integrates to 1 for all $t \quad 0$. In this example, we consider the case when



(A) Error (B) Mapping comparison (C) Density evolution

FIGURE 6. Left: log-log plot of Fokker-Planck equation with a quartic polynomial potential. The y-axis represents $\log_{10}$ error defined by (56). x-axis represents $\log_{10}(N)$. The bias terms b_i are initialized based on Section 5.1 with $B = 4$. Red lines represent results when only the weights terms are updated. Black lines represent results when both weights and bias are updated. Middle: mapping comparison between $T(t, z)$ (using Eq. (61)) and our computed solution $f(\theta_t, z)$. Right: density evolution of the Fokker-Planck equation with a quartic polynomial potential. Orange curve represents the density $p(t, x)$ computed by a numerical solver. Blue rectangles represent the histogram of 10^6 particles in 100 bins from $t = 0$ to $t = 0.2$.

$m = 2$. Then $\alpha = \frac{1}{3}$, $\beta = \frac{1}{12}$ and $C = \frac{3^{1/3}}{4}$. Eq. (72) also suggests that the support of the reference measure $p_0(x) = p(x, 0)$ is bounded in $[\ \ 3^{2=3}t_0^{1=3}, 3^{2=3}t_0^{1=3}]$, which could help us with initializing the bias. More precisely, we cound initialize our b_i's following Section 5.1 with $B = 3^{2=3}t_0^{1=3}$. In our experiment, we set $t_0 = 1$. We use $dt = 10^{\ 3}$ and run for 1000 steps. We use $M = 10^6$ particles sampled from $p(x, 0)$ defined in Eq. (72) using importance sampling to approximate $\mathsf{E}_{z\ \ p_{\mathrm{r}}}\left[r\ \ \hat{U}(\frac{p_{\mathrm{r}}(z)}{D_z f(\ ;z)})\right]$, where $\hat{U}(p) = p$. Similar to the case of Fokker-Planck equation, special care needs to be taken when evaluating $\partial b_i \mathsf{E}_{z\ \ p_{\mathrm{r}}}\left[\hat{U}(\frac{p_{\mathrm{r}}(z)}{D_z f(\ ;z)})\right]$. Using similar analysis from Section 5.3, we have that

$$\partial b_i \mathsf{E}_{z\ \ p_{\mathrm{r}}}\left[\hat{U}(\frac{p_{\mathrm{r}}(z)}{D_z f(\theta, z)})\right] = \begin{cases} \frac{p_{\mathrm{r}}(b_i)^2}{D_z f(\ ;b_i\ \ )}\ \ \frac{p_{\mathrm{r}}(b_i)^2}{D_z f(\ ;b_i)}, & 1 \quad i \quad N\,. \\ \frac{p_{\mathrm{r}}(b_i)^2}{D_z f(\ ;b_i)}\ \ \frac{p_{\mathrm{r}}(b_i)^2}{D_z f(\ ;b_i+\ )}, & N+1 \quad i \quad 2N\,. \end{cases} \tag{73}$$

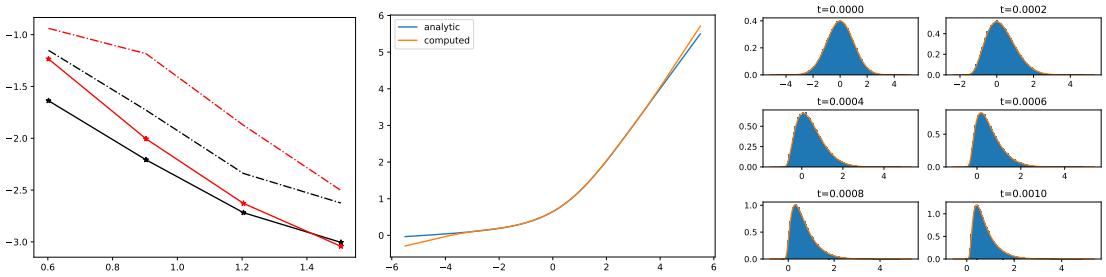
Our results are demonstrated in Fig. 8. In Fig. 8b, 8c we have $N = 32$; both the bias and weights are updated.

5.5. **Keller-Segel equation.** We consider the one-dimensional modified Keller-Segel equation, which is a combination of interaction energy in Example 5 and potential energy in Example 6:

$$\partial_t p(t, x) = r \quad \big(p(t, x)\, r\, (U^0(p) + W \quad p)\big), \tag{74}$$

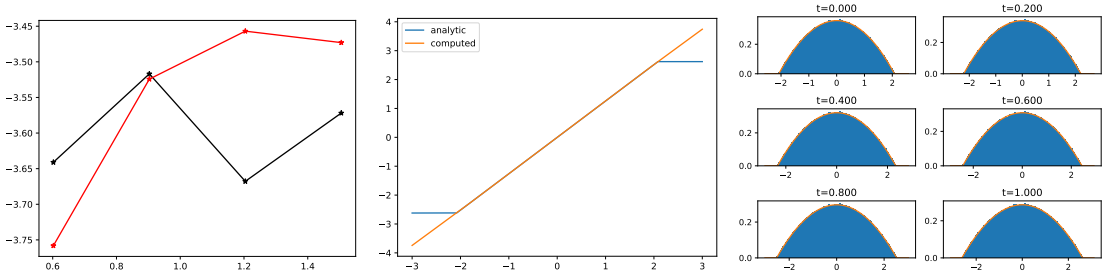
where $U(p) = p \log p$ and $W(x) = 2\chi \log jxj$, $\chi > 0$ is a constant. The second moment of $p(t, x)$ has an analytic form given by

$$\mathsf{E}_{z\ \ p(\ ;t)}[z^2] = 2(1 \quad \chi)t\, \mathsf{E}_{z\ \ p(\ ;0)}[z^2]\,. \tag{75}$$



(A) Error (B) Mapping comparison (C) Density evolution

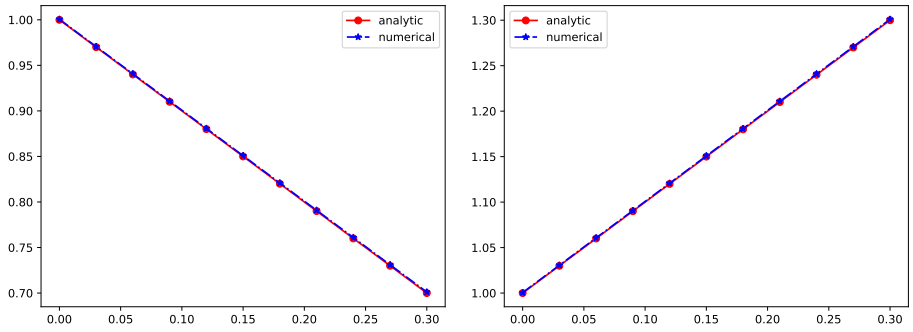
FIGURE 7. Left: log-log plot of Fokker-Planck equation with a sixth order polynomial potential. The y-axis represents $\log_{10}$ error defined by (56). x-axis represents $\log_{10}(N)$. The bias terms b_i are initialized based on Section 5.1 with $B = 10$ for the dashed line and $B = 4$ for the solid line. Red lines represent results when only the weights terms are updated. Black lines represent results when both weights and bias are updated. Middle: mapping comparison between $T(t, z)$ (using Eq. (61)) and our computed solution $f(\theta_t, z)$. Right: density evolution of the Fokker-Planck equation with a sixth order polynomial potential. Orange curve represents the density $p(t, x)$ computed by a numerical solver. Blue rectangles represents the histogram of 10^6 particles in 100 bins from $t = 0$ to $t = 10$ [3].



(A) Error (B) Mapping comparison (C) Density evolution

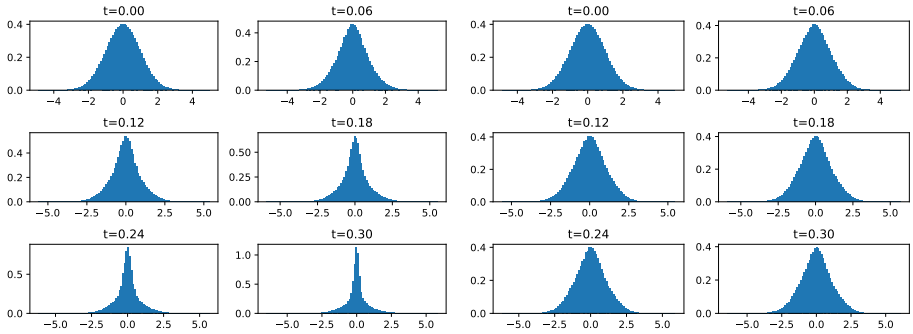
FIGURE 8. Left: log-log plot of porous medium equation. The y-axis represents $\log_{10}$ error defined by (56). x-axis represents $\log_{10}(N)$. The bias terms b_i are initialized based on Section 5.1 with $B = 3^{2=3}$. Red lines represent results when only the weights terms are updated. Black lines represent results when both weights and bias are updated. Middle: mapping comparison between $T(t, z)$ (using Eq. (61)) and our computed solution $f(\theta_t, z)$. Right: density evolution of the porous medium equation. Orange curve represents the density $p(t, x)$ given by Eq. (71). Blue rectangles represent the histogram of 10^6 particles in 100 bins from $t = 0$ to $t = 1$.

It is clear from Eq. (75) that $\chi = 1$ is a critical value. When $\chi > 1$, the solution blows up as $t \; ! \; 1$. So we consider two cases: $\chi = 1.5$, and $\chi = 0.5$. We present our results in Fig. 9, 10 and 11. We used 2000 particles with a standard Gaussian initial distribution.



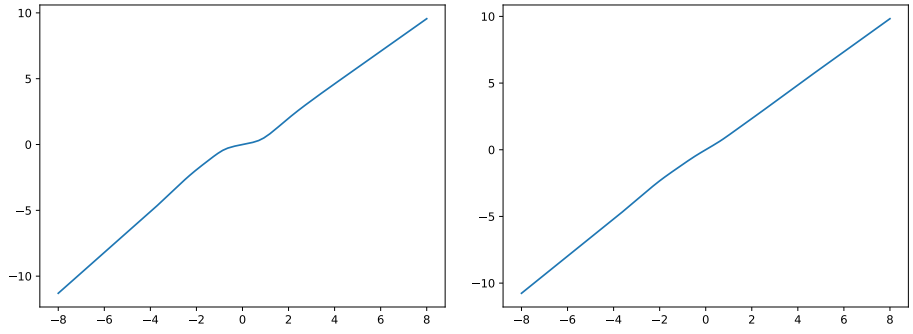
(A) $\chi = 1.5$ (B) $\chi = 0.5$

FIGURE 9. Second moment comparison between our numerical solution and analytic solution (75). x-axis represents time.



(A) $\chi = 1.5$ (B) $\chi = 0.5$

FIGURE 10. Density evolution of Keller-Segel equation with different χ. Blue rectangles represent the histogram of 10^6 particles in 100 bins from $t = 0$ to $t = 0.3$.



(A) $\chi = 1.5$ (B) $\chi = 0.5$

FIGURE 11. Lagrangian mapping of Keller-Segel equation with different χ at $t = 0.3$.

We set $dt = 3 \quad 10^{\ 4}$ and run for 1000 steps. The interaction term $W \quad p$ is evaluated using the 2000 particles with self-interaction excluded. We used a neural network with $N = 32$ and $B = 4$ following the setup and initialization in Section 5.1. We update both the bias and weights terms in our experiment.

6. Discussion

This paper analyzes the neural network projected dynamics for one-dimensional Wasserstein gradient flows of general energy functionals. For two-layer neural network functions with ReLU activations, we analyze the convergence and stability issues for the proposed numerical schemes from location parameter b and scale parameter a. In numerical experiments, we demonstrate the second-order spatial domain accuracy as discussed in the numerical analysis.

In future work, we shall study neural projected dynamics as a computational framework for building theoretical guaranteed machine learning numerical schemes. Various topics in this direction remain to be studied. First, we shall design neural network approximations to approximate the initial value of high-dimensional PDEs, which traditional PDE solvers cannot efficiently solve due to the curse of dimensionality. In particular, how can we understand the numerical accuracy of deep neural network functions in high dimensions when approximating PDEs? Next, we shall generalize the neural projected dynamics to dynamical systems for conservative-dissipative equations in statistical physics. The equation includes Hamiltonian structures induced from the conservative system and the related mean-field control problems. Furthermore, considering the closed relationship between the Wasserstein density manifold and sampling algorithms, we shall investigate sampling using the projected dynamics on neural parameter spaces and study their theoretical and statistical properties. We also consider the time-implicit (proximal-type) computations of the proposed algorithm [19, 23], which could improve the performance and stability of the scheme.

References

[1] Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.
[2] Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.
[3] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2005.
[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.
[5] Andrew J Barlow, Pierre-Henri Maire, William J Rider, Robert N Rieben, and Mikhail J Shashkov. Arbitrary Lagrangian–Eulerian methods for modeling high-speed compressible multimaterial flows. *Journal of Computational Physics*, 322:603–665, 2016.
[6] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.
[7] Joan Bruna, Benjamin Peherstorfer, and Eric Vanden-Eijnden. Neural Galerkin schemes with active learning for high-dimensional evolution equations. *Journal of Computational Physics*, 496:112588, 2024.
[8] Jose A Carrillo, Daniel Matthes, and Marie-Therese Wolfram. Lagrangian schemes for Wasserstein gradient flows. *Handbook of Numerical Analysis*, 22:271–311, 2021.

[9] JS Chang and G Cooper. A practical difference scheme for Fokker–Planck equations. *Journal of Computational Physics*, 6(1):1–16, 1970.
[10] Yifan Chen and Wuchen Li. Optimal transport natural gradient for statistical manifolds with continuous sample space. *Information Geometry*, 3(1):1–32, 2020.
[11] Casey Chu, Kentaro Minami, and Kenji Fukumizu. The equivalence between Stein variational gradient descent and black-box variational inference. *arXiv preprint arXiv:2004.01822*, 2020.
[12] Yifan Du and Tamer A Zaki. Evolutional deep neural network. *Physical Review E*, 104(4):045303, 2021.
[13] Jiaojiao Fan, Qinsheng Zhang, Amirhossein Taghvaei, and Yongxin Chen. Variational Wasserstein gradient flow. *arXiv preprint arXiv:2112.02424*, 2021.
[14] Nathan Gaby, Xiaojing Ye, and Haomin Zhou. Neural control of parametric solutions for high-dimensional evolution PDEs. *arXiv preprint arXiv:2302.00045*, 2023.
[15] Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. ReLU deep neural networks and linear finite elements. *Journal of Computational Mathematics*, 38(3):502–527, June 2020.
[16] Ziqing Hu, Chun Liu, Yiwei Wang, and Zhiliang Xu. Energetic variational neural network discretizations to gradient flows. *arXiv preprint arXiv:2206.07303*, 2022.
[17] Weizhang Huang and Robert D Russell. *Adaptive moving mesh methods*, volume 174. Springer Science & Business Media, 2010.
[18] Wonjun Lee, Li Wang, and Wuchen Li. Deep JKO: time-implicit particle methods for general nonlinear gradient flows. *arXiv preprint arXiv:2311.06700*, 2023.
[19] Wuchen Li, Alex Tong Lin, and Guido Montúfar. Affine natural proximal learning. In *Geometric Science of Information: 4th International Conference, GSI 2019, Toulouse, France, August 27–29, 2019, Proceedings 4*, pages 705–714. Springer, 2019.
[20] Wuchen Li and Guido Montúfar. Natural gradient via optimal transport. *Information Geometry*, 1:181–214, 2018.
[21] Wuchen Li and Jiaxi Zhao. Scaling limits of the Wasserstein information matrix on Gaussian mixture models. *arXiv preprint arXiv:2309.12997*, 2023.
[22] Wuchen Li and Jiaxi Zhao. Wasserstein information matrix. *Information Geometry*, pages 1–53, 2023.
[23] Alex Tong Lin, Wuchen Li, Stanley Osher, and Guido Montúfar. Wasserstein proximal of gans. In *International Conference on Geometric Science of Information*, pages 524–533. Springer, 2021.
[24] Chun Liu and Yiwei Wang. On Lagrangian schemes for porous medium type generalized diffusion equations: A discrete energetic variational approach. *Journal of Computational Physics*, 417:109566, 2020.
[25] Shu Liu, Wuchen Li, Hongyuan Zha, and Haomin Zhou. Neural parametric Fokker–Planck equation. *SIAM Journal on Numerical Analysis*, 60(3):1385–1449, 2022.
[26] Pierre-Henri Maire, Rémi Abgrall, Jérôme Breil, and Jean Ovadia. A cell-centered Lagrangian scheme for two-dimensional compressible flow problems. *SIAM Journal on Scientific Computing*, 29(4):1781–1824, 2007.
[27] Petr Mokrov, Alexander Korotin, Lingxiao Li, Aude Genevay, Justin M Solomon, and Evgeny Burnaev. Large-scale Wasserstein gradient flows. *Advances in Neural Information Processing Systems*, 34:15243–15256, 2021.
[28] Kirill Neklyudov, Rob Brekelmans, Alexander Tong, Lazar Atanackovic, Qiang Liu, and Alireza Makhzani. A computational framework for solving Wasserstein Lagrangian flows. *arXiv preprint arXiv:2310.10649*, 2023.
[29] Levon Nurbekyan, Wanzhou Lei, and Yunan Yang. Efficient natural gradient descent methods for large-scale PDE-based optimization problems. *SIAM Journal on Scientific Computing*, 45(4):A1621–A1655, 2023.
[30] N Nüsken. On the geometry of Stein variational gradient descent. *Journal of Machine Learning Research*, 24:1–39, 2023.
[31] Yann Ollivier. Riemannian metrics for neural networks I: feedforward networks. *Information and Inference: A Journal of the IMA*, 4(2):108–153, 2015.
[32] Lars Onsager. Reciprocal relations in irreversible processes. I. *Phys. Rev.*, 37:405–426, Feb 1931.
[33] Felix Otto. The geometry of dissipative evolution equations the porous medium equation. *Communications in Partial Differential Equations*, 26(1-2):101–174, 2001.

[34] Lars Ruthotto, Stanley J Osher, Wuchen Li, Levon Nurbekyan, and Samy Wu Fung. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proceedings of the National Academy of Sciences*, 117(17):9183–9193, 2020.
[35] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
[36] Tao Tang. Moving mesh methods for computational fluid dynamics. *Contemporary mathematics*, 383(8):141–173, 2005.
[37] Cédric Villani. *Optimal Transport: Old and New*, volume 338. Springer, 2009.
[38] Hao Wu, Shu Liu, Xiaojing Ye, and Haomin Zhou. Parameterized Wasserstein Hamiltonian flow. *arXiv preprint arXiv:2306.00191*, 2023.

Email address: zxz@math.ucla.edu

Department of Mathematics, University of California, Los Angeles, CA, 90095.

Email address: jiaxi.zhao@u.nus.edu

Department of Mathematics, National University of Singapore.

Email address: shuliu@math.ucla.edu

Department of Mathematics, University of California, Los Angeles, CA, 90095.

Email address: sjo@math.ucla.edu

Department of Mathematics, University of California, Los Angeles, CA, 90095.

Email address: wuchen@mailbox.sc.edu

Department of Mathematics, University of South Carolina, Columbia, SC, 29208.