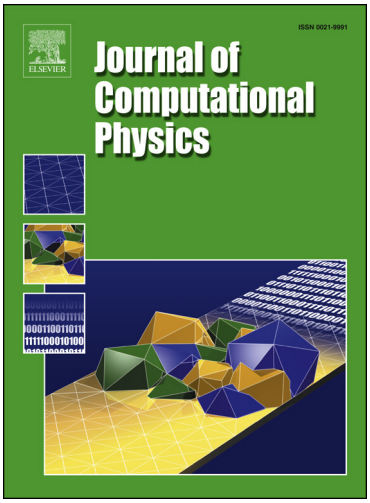
Journal Pre-proof



Deep JKO: time-implicit particle methods for general nonlinear gradient flows

Wonjun Lee, Li Wang and Wuchen Li

PII: S0021-9991(24)00436-4
DOI: https://doi.org/10.1016/j.jcp.2024.113187
Reference: YJCPH 113187

To appear in: *Journal of Computational Physics*

Received date: 16 November 2023
Revised date: 14 May 2024
Accepted date: 5 June 2024

Please cite this article as: W. Lee, L. Wang and W. Li, Deep JKO: time-implicit particle methods for general nonlinear gradient flows, *Journal of Computational Physics*, 113187, doi: https://doi.org/10.1016/j.jcp.2024.113187.

Highlights

- A neural network empowered implicit particle method is proposed for computing high dimensional nonlinear gradient flows.
- Our method benefits from the structure preservation properties inherent in the JKO formulation and can handle high-dimensional problems due to its mesh-free discretization.
- We consider general nonlinear functionals and nonlinear mobility, which are currently beyond the scope of large-scale gradient flow solvers.
- Concepts from continuous normalizing flow are employed for efficient density computation, and an explicit recurrence relation for derivative computation is utilized to significantly streamline the backpropagation process.
- We have showcased the effectiveness and versatility of our method in both high dimensions and diverse equations, including one arising from an inverse problem—the Kalman-Wasserstein gradient flow.

Deep JKO: time-implicit particle methods for general nonlinear gradient flows

Wonjun Lee[a], Li Wang[b], Wuchen Li[c]

[a]*Institute for Mathematics and Its Applications, University of Minnesota, U.S.*
[b]*School of Mathematics, University of Minnesota, U.S.*
[c]*Department of Mathematics, University of South Carolina, U.S.*

Abstract

We develop novel neural network-based implicit particle methods to compute high-dimensional Wasserstein-type gradient flows with linear and nonlinear mobility functions. The main idea is to use the Lagrangian formulation in the Jordan–Kinderlehrer–Otto (JKO) framework, where the velocity field is approximated using a neural network. We leverage the formulations from the neural ordinary differential equation (neural ODE) in the context of continuous normalizing flow for efficient density computation. Additionally, we make use of an explicit recurrence relation for computing derivatives, which greatly streamlines the backpropagation process. Our methodology demonstrates versatility in handling a wide range of gradient flows, accommodating various potential functions and nonlinear mobility scenarios. Extensive experiments demonstrate the efficacy of our approach, including an illustrative example from Bayesian inverse problems. This underscores that our scheme provides a viable alternative solver for the Kalman-Wasserstein gradient flow.

Keywords: Nonlinear gradient flows, JKO scheme, Neural ODE, Particle methods, Density estimation, Nonlinear mobility.

1. Introduction

Deep learning has revolutionized many areas, fundamentally reshaping fields such as natural language processing, visual recognition, and beyond [27]. It has also emerged as the cornerstone of contemporary scientific computing. By harnessing the power of deep neural networks and their ability to approximate complex functions, it has become an invaluable tool for approximating solutions to partial differential equations (PDEs), particularly those of high dimensions. These equations often defy conventional computational techniques due to their complexity and dimensionality.

Several methodologies have employed neural network approximations for solving PDEs. One approach is the deep Ritz method, crafted upon the variational formulation of elliptic-type equations [44]. Another avenue is the deep

backward stochastic differential equation (BSDE) method, based on the probabilistic and control-based formulation of parabolic equations [23]. A third technique involves the deep Galerkin method [38] and the physics-informed neural network (PINN) [35]. These methodologies both hinge on the minimization of the (L^2) residual of the equation. Importantly, they possess the capability to be employed across a wide spectrum of equation types. In parallel, this paper aims to develop a versatile solver for a substantial class of nonlinear PDEs characterized by a gradient flow structure.

Gradient flows describe the evolution of a function or a probability measure over time, with the rate of change at any point being proportional to the negative gradient of the energy with respect to the function's value. They were originally used to study physical or biological systems that naturally obey some form of least action law. However, they have since emerged as powerful tools in various domains, including sampling, variational inference, the theory of neural networks, and more.

To be more specific, we consider the continuity equation of the form:

$$\partial_t \rho = -\nabla_{\boldsymbol{x}} \cdot (\rho \boldsymbol{v}) := \nabla_{\boldsymbol{x}} \cdot [\rho \nabla_{\boldsymbol{x}} (U'(\rho) + V + W * \rho)], \quad \rho(0, \cdot) = \rho_0 \,. \tag{1}$$

Here, $\rho(t, \boldsymbol{x})$, with $\boldsymbol{x} \in \mathbb{R}^d$, represents the particle density function, ρ_0 is the initial density function, $U(\rho) \in \mathbb{R}$ represents the internal potential, $V(\boldsymbol{x}) \in \mathbb{R}$ is a drift potential, $W(\boldsymbol{x}, \mathbf{y}) = W(\mathbf{y}, \boldsymbol{x}) \in \mathbb{R}$ is an interaction potential involving $\boldsymbol{x}$ and $\mathbf{y}$ in $\mathbb{R}^d$, and $*$ denotes the convolution operator. Without loss of generality, we assume that U, V and W are sufficient regular in the sense that $V, W \in C(\mathbb{R})$, and $U \in C^1((0, \infty))$ with $\lim_{r \to \infty} \frac{U(r)}{r} = +\infty$ and $U(0) = 0$. See [10, Section 3.1] for more discussion.

This equation can be viewed as the gradient flow of the energy functional [40, 2]

$$\mathcal{E}(\rho) = \int_{\mathbb{R}^d} [U(\rho(\boldsymbol{x})) + V(\boldsymbol{x})\rho(\boldsymbol{x})] \, \mathrm{d}\boldsymbol{x} + \frac{1}{2} \int_{\mathbb{R}^d \times \mathbb{R}^d} W(\boldsymbol{x} - \boldsymbol{y}) \rho(\boldsymbol{x}) \rho(\boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \tag{2}$$

in Wasserstein-2 space, which is the space of probability measures equipped with Wasserstein-2 distance, denoted as $\mathcal{W}_2$.

Among all the recent endeavors in devising structure-preserving methodologies for (1), our focus in this paper will be exclusively on the Jordan-Kindelenr-Otto (JKO) scheme [26]. This scheme centers on the task of determining $\rho^{k+1}(\cdot) \approx \rho(t^{k+1}, \cdot)$ with $t^{k+1} = (k+1)\Delta t$, given the approximation $\rho^k(\cdot) \approx \rho(t^k, \cdot)$:

$$\rho^{k+1} \in \arg\min_{\rho} \{ \mathcal{W}_2(\rho, \rho^k)^2 + 2\Delta t \mathcal{E}(\rho) \} \,. \tag{3}$$

Here, $\mathcal{W}_2(\rho, \rho^k)$ denotes the Wasserstein-2 distance between ρ and ρ^k. The scheme in (3) can be conceptualized as a time-implicit version of the fundamental gradient descent, with the expression on the right-hand side of (3) being recognized as the Wasserstein proximal operator associated with energy functional $\mathcal{E}$.

In general terms, there exist two methods for expressing the $\mathcal{W}_2$ distance, each leading to distinct implementations of (3). The first method is the Eulerian representation, which establishes a connection between two densities through a continuity equation involving a velocity field that minimizes kinetic energy during the transport. In contrast, the second method is grounded in the Lagrangian formulation [11]. This approach explores the diffeomorphism that maps one density onto another, a technique that has also gained prominence in modern machine learning applications.

In this paper, we adopt the Lagrangian viewpoint by employing the particle method, coupled with principles drawn from neural ordinary differential equations (neural ODEs). To elaborate, starting from a set of particles $\{\boldsymbol{x}_j^0\}$ that are i.i.d. samples from ρ^0, we seek a sequence of transformations governing the update of all particles' positions as follows:

$$\{\boldsymbol{x}_j^0\} \xrightarrow{\boldsymbol{T}^1} \{\boldsymbol{x}_j^1\} \xrightarrow{\boldsymbol{T}^2} \{\boldsymbol{x}_j^2\} \cdots .$$

Here, $\boldsymbol{x}_j^k = \boldsymbol{T}^k(\boldsymbol{x}_j^{k-1})$. As a consequence, the density evolves in accordance with $\rho^k = \boldsymbol{T}^k \sharp \rho^{k-1}$ for $k = 1, 2, \ldots$, where $\sharp$ denotes the pushforward operator. We then approximate each map $\boldsymbol{T}^k$ by parameterizing the corresponding vector field using a deep neural network. Thanks to the Jacobi's formula, this results in a simple update in estimating the density ρ^k; see (4c) below.

More precisely, our primary formulation is as follows. To transit from time t^k to t^{k+1}, we solve the following minimization problem:

$$\begin{aligned}\theta^* = \\ \arg\min_{\theta\in\Theta} \ \mathbb{E}_{\boldsymbol{x}\sim\rho}\Big[\int_0^1 |\boldsymbol{v}_\theta(\tau, \boldsymbol{T}(\tau,\boldsymbol{x}))|^2 \mathrm{d}\tau + 2\Delta t\Big(\frac{U(\rho(\boldsymbol{T}(1,\boldsymbol{x})))}{\rho(\boldsymbol{T}(1,\boldsymbol{x}))} + V(\boldsymbol{T}(1,\boldsymbol{x}))\Big)\Big] \\ + \Delta t \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\rho\otimes\rho}\Big[W(\boldsymbol{T}(1,\boldsymbol{x}), \boldsymbol{T}(1,\boldsymbol{y}))\Big],\end{aligned} \tag{4a}$$

such that $\boldsymbol{T}(\tau, x)$ satisfies the dynamical constraint:

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{T}(\tau,\boldsymbol{x}) = \boldsymbol{v}_\theta(\tau, \boldsymbol{T}(\tau,\boldsymbol{x})), \qquad \boldsymbol{T}(0,\boldsymbol{x}) = \boldsymbol{x}, \tag{4b}$$

and the density $\rho(\boldsymbol{T}(1,\boldsymbol{x}))$ is computed as first evolving:

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\log|\det\nabla_{\boldsymbol{x}}\boldsymbol{T}(\tau,\boldsymbol{x})| = \mathrm{div}(\boldsymbol{v}_\theta)(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))\,, \quad \rho(\boldsymbol{T}(0,\boldsymbol{x})) = \rho^k(\boldsymbol{x})\,, \tag{4c}$$

followed by the pushforward relation:

$$\rho(\boldsymbol{T}(1,\boldsymbol{x})) = \rho^k(\boldsymbol{x})/|\det\nabla_{\boldsymbol{x}}\boldsymbol{T}(1,\boldsymbol{x})|\,. \tag{4d}$$

In this problem, we employ Benamou-Brenier's dynamic formulation [1] to calculate the Wasserstein distance and reformulate the continuity equation using the flow map $\boldsymbol{T}(\tau,\boldsymbol{x})$, with τ serving as an artificial time. The associated

velocity field is denoted as $\boldsymbol{v}_\theta$, where $\theta \in \Theta$ represents the parameters of a neural network. The notation $\mathbb{E}_{\boldsymbol{x}\sim\rho}$ denotes the expectation value for $\boldsymbol{x}$ distributed according to the probability density ρ, and $\mathbb{E}(\boldsymbol{x},\boldsymbol{y}) \sim \rho \otimes \rho$ represents the expectation over pairs of independent variables $(\boldsymbol{x},\boldsymbol{y})$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent and satisfy the joint density $\rho \otimes \rho$. Equation (4c) is a significant outcome from the perspective of neural ODEs, providing a straightforward means to compute $|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau,\boldsymbol{x})|$. Consequently, this facilitates an efficient computation of density as outlined in (4d).

It's crucial to highlight that the proposed method goes beyond the current scope of merely approximating Wasserstein gradient flows. Similar approaches, as outlined in (4), can be employed for general gradient flows, including nonlinear mobility Wasserstein and Kalman-Wasserstein gradient flows. In such cases, the identity (4c), utilized in neural ODE density estimations, takes on even greater significance. This is due to the fact that the objective function in (4) becomes dependent on the density, introducing additional nonlinearities into the optimization objective (loss function), see (24) in the subsequent context.

The exploration of computing (1) using the JKO scheme has experienced rapid growth. Previous endeavors employed finite difference or finite volume schemes [2, 10, 29, 41, 14, 19, 7], as well as Lagrangian methods [11, 13, 12]. More recently, efforts have been directed towards developing machine learning-based methods, particularly in tackling high-dimensional challenges. In this context, neural networks have been utilized to approximate maps or density functions [31, 18, 33, 25, 37, 4]. Furthermore, a variety of methods have arisen to approximate different formulations of one-step JKO schemes. These range from leveraging generative adversarial networks [30] to employing neural ODEs [36]. These approaches primarily target at sampling of high-dimensional distributions in machine learning applications [39, 42], manifesting as time-implicit updates of the projected dynamics within neural network spaces, specifically the natural gradient dynamics [30].

Contrasting previous methodologies, our approach employs neural ODE techniques to compute the time-implicit update of generalized Wasserstein-type gradient flows. This innovative strategy facilitates the streamlined computation of high-dimensional densities in Lagrangian coordinates. It is worth highlighting the similarities and disparities between our approach and its closest counterparts. One such sibling is the blob method [9]. While both approaches rely on particles and intrinsically exhibit energy dissipation, the dissipation in the blob method hinges on the chosen time step, whereas ours remains unconditionally stable. Furthermore, our method eliminates the need for kernel density estimation, a significant computational bottleneck present in the blob method. Another approach involves the direct use of a neural network to approximate the mapping between ρ^k and ρ in (3) [33, 25]. This approach can be seen as a special case of our method when the inner time step τ is set to 1. A critical distinction, however, lies in our utilization of the density evolution equation (4c) from the perspective of neural ODEs and the explicit formula (28) for computing derivatives, which significantly enhances the efficiency of density estimation. Furthermore, our approach can be readily extended to accommodate the non-

linear mobility case. A third comparable technique is the self-consistent velocity matching approach [4, 37]. In this method, neural networks are also employed to approximate the velocity field. However, instead of sequentially learning it within the JKO formulation as we do, this approach adopts an end-to-end learning strategy. While this method might lead to faster computation, it doesn't inherently preserve energy decay in general.

This paper is organized as follows. In section 2, we review the general formulation of Wasserstein gradient flows in Lagrangian coordinates and introduce the Lagrangian Wasserstein proximal operator and its particle version for time-implicit (i.e., JKO) updates of gradient flows. We expand our framework to include gradient flows characterized by metrics reliant on general nonlinear mobility functions. Section 3 outlines our proposition to approximate the push-forward map using the trajectory of neural ODEs. This approach empowers us to employ neural networks with diverse depths while simultaneously sidestepping the computationally intensive task of back-propagation. To illustrate the effectiveness and versatility of our approach, we present a series of numerical examples in Section 4, including Fokker-Planck equations, porous media equations, Kalman-Wasserstein gradient flow of Kullback–Leibler (KL) divergence, and nonlinear mobility gradient flows. Finally, the paper concludes in Section 5 with an overview of our findings and insights into future research directions.

2. Lagrangian JKO formulation

In this section, we present the main formulation of this paper. We first review the Lagrangian coordinates of Wasserstein gradient flows and then introduce our variational time-implicit schemes. One major component of our formulation is to use instantaneous change of variable formula (see (13)) in the continuous normalizing flow to achieve efficient density computation. We also consider the generalized nonlinear gradient flows, in which the Wasserstein-type metric is associated with nonlinear mobility functions.

2.1. Dynamic JKO schemes in Lagrangian coordinates

Recall the dynamic JKO formulation to (1) as follows. Given $\rho^n(\boldsymbol{x})$, then one can obtain $\rho^{n+1}(\boldsymbol{x})$ as $\rho(1,\boldsymbol{x})$ with $\rho(\tau,\boldsymbol{x})$ solving

$$\begin{cases} (\rho,\boldsymbol{v}) = \arg\inf\limits_{\rho,\boldsymbol{v}} \int_0^1 \int_{\mathbb{R}^d} \rho|\boldsymbol{v}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}\tau + 2\Delta t\mathcal{E}(\rho(1,\cdot)) \\ \text{s.t.} \quad \partial_\tau\rho + \nabla\cdot(\rho\boldsymbol{v}) = 0,\ \rho(0,\boldsymbol{x}) = \rho^n(\boldsymbol{x})\,. \end{cases} \tag{5}$$

In the above formulation, the minimization is over the probability density function $\rho(\tau,\boldsymbol{x})$, $\tau\in(0,1)$, terminal time density $\rho(1,\boldsymbol{x})$, and vector fields $\boldsymbol{v}(\tau,\boldsymbol{x})$, such that the continuity equation holds.

Now we translate the variational problem (5) into the Lagrangian formulation. Assume the velocity field is sufficiently regular, then the solution $\rho(\tau,\boldsymbol{x})$ to the constrained continuity equation can be written as

$$\rho(\tau,\cdot) = \boldsymbol{T}(\tau,\cdot)\sharp\rho^n\,, \tag{6}$$

where $\boldsymbol{T}$ is the flow map that solves the following ODE:

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{T}(\tau,\boldsymbol{x}) = \boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x})), \qquad \boldsymbol{T}(0,\boldsymbol{x}) = \boldsymbol{x}\,. \tag{7}$$

Note that (6) implies that for any integrable test function ϕ, we have that

$$\int_{\mathbb{R}^d} \phi(\boldsymbol{x})\rho(\tau,\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \int_{\mathbb{R}^d} \phi(\boldsymbol{T}(\tau,\boldsymbol{x}))\rho^n(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\,.$$

Then variational problem (5) rewrites as

$$\begin{cases} \min\limits_{\boldsymbol{v}} \displaystyle\int_0^1 \int_{\mathbb{R}^d} \rho^n(\boldsymbol{x})|\boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))|^2\mathrm{d}\tau\mathrm{d}\boldsymbol{x} + 2\Delta t\mathcal{E}(\boldsymbol{T}(1,\cdot)\sharp\rho^n) \\ \text{s.t.} \quad \dfrac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{T}(\tau,\boldsymbol{x}) = \boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x})), \qquad \boldsymbol{T}(0,\boldsymbol{x}) = \boldsymbol{x}\,. \end{cases} \tag{8}$$

To proceed, it is necessary to derive an explicit formulation of $\mathcal{E}(\boldsymbol{T}(1,\cdot)\sharp\rho^n)$ that can be interpreted as an expectation of a functional, allowing for its representation using particles in the subsequent discussion. In the following, we present a comprehensive formulation for each component of equation (2). Specifically, the external potential and interaction potential have straightforward expectation formulations through the push forward relations, whereas the internal energy necessitates further reformulation.

External potential energy

$$\int_{\mathbb{R}^d} V(\boldsymbol{x})\rho(1,\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \int_{\mathbb{R}^d} V(\boldsymbol{T}(1,\boldsymbol{x}))\rho^n(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\,. \tag{9}$$

Interaction energy

$$\begin{aligned} &\int_{\mathbb{R}^d\times\mathbb{R}^d} W(\boldsymbol{x},\boldsymbol{y})\rho(1,\boldsymbol{x})\rho(1,\boldsymbol{y})\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{y} \\ &\quad = \int_{\mathbb{R}^d\times\mathbb{R}^d} W(\boldsymbol{T}(1,\boldsymbol{x}),\boldsymbol{T}(1,\boldsymbol{y}))\rho^n(\boldsymbol{x})\rho^n(\boldsymbol{y})\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{y}\,. \end{aligned} \tag{10}$$

Internal energy/entropy

$$\begin{aligned} &\int_{\mathbb{R}^d} U(\rho(1,\boldsymbol{x}))\mathrm{d}\boldsymbol{x} \\ &= \int_{\mathbb{R}^d} U(\rho(1,\boldsymbol{T}(1,\boldsymbol{x})))|\det\nabla_{\boldsymbol{x}}\boldsymbol{T}(1,\boldsymbol{x})|\mathrm{d}\boldsymbol{x} \\ &= \int_{\mathbb{R}^d} U(\rho(1,\boldsymbol{T}(1,\boldsymbol{x})))\frac{|\det\nabla_{\boldsymbol{x}}\boldsymbol{T}(1,\boldsymbol{x})|}{\rho^n(\boldsymbol{x})}\rho^n(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &= \int_{\mathbb{R}^d} U(\rho(1,\boldsymbol{T}(1,\boldsymbol{x})))\frac{1}{\rho(1,\boldsymbol{T}(1,\boldsymbol{x}))}\rho^n(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\,, \end{aligned} \tag{11}$$

where we have used the Monge-Amperè equation

$$\rho(1, \boldsymbol{T}(1, \boldsymbol{x}))|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(1, \boldsymbol{x})| = \rho^n(\boldsymbol{x}) \tag{12}$$

thanks to (6). In practice, computing the determinant of the Jacobian $\nabla_{\boldsymbol{x}} \boldsymbol{T}(1, \boldsymbol{x}))$ poses a significant bottleneck due to its cubic cost with respect to the dimension of $\boldsymbol{x}$. To address this issue, we draw inspiration from the concept of continuous normalizing flow and employ an efficient approach based on the instantaneous change of variable formula [15, 22]. The crucial element is the following formula.

Proposition 1.

$$\frac{\mathrm{d}}{\mathrm{d}\tau} \log|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x})| = \mathit{div}(\boldsymbol{v})(\tau, \boldsymbol{T}(\tau, \boldsymbol{x})). \tag{13}$$

Proof. Equation (13) is derived through a calculus that involves both ODE (7) and the Monge-Amperè equation (12). Assume that $\boldsymbol{T}(\tau, \boldsymbol{x})$ is invertible, i.e. $\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x}) \neq 0$, then

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\tau} \log|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x})| =& \frac{\mathit{sgn}(\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x}))}{|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x})|} \frac{\mathrm{d}}{\mathrm{d}\tau} (\det(\nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x}))) \\
=& \mathrm{tr}\Big((\nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x}))^{-1} \frac{\mathrm{d}}{\mathrm{d}\tau} \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x}) \Big) \\
=& \mathrm{tr}\Big((\nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x}))^{-1} \nabla_x \frac{\mathrm{d}}{\mathrm{d}\tau} \boldsymbol{T}(\tau, \boldsymbol{x}) \Big) \\
=& \mathrm{tr}\Big((\nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x}))^{-1} \nabla_x \boldsymbol{v}(\tau, \boldsymbol{T}(\tau, \boldsymbol{x})) \Big) \\
=& \mathrm{div}(\boldsymbol{v})(\tau, \boldsymbol{T}(\tau, \boldsymbol{x})),
\end{aligned}$$

where the second equality uses the Jacobi's formula, and the last equality uses the chain rule, and $\mathit{sgn}(x) = \frac{x}{|x|}$. □

As a result, the computational burden shifts towards calculating $\mathrm{div}(\boldsymbol{v})(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}))$, which only involves a trace calculation and can be accomplished much more efficiently. In sum, (12) and (13) play a crucial role in enabling efficient density estimation, as elaborated in the next section and Section 3.1.

2.2. A Particle method

By interpreting the integral against $\rho(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ as an expectation, (8) reveals a direct particle representation. More precisely, let $\{\boldsymbol{x}_j^n\}_{j=1}^N$ be particles sampled from $\rho^n(\boldsymbol{x})$, we discretize (8) as follows. Note here we omit the superscript n in $\boldsymbol{x}_j^n$.

Case 1 : (8) with (9) and/or (10)

$$\begin{cases} \min\limits_{\boldsymbol{v}} \dfrac{1}{N}\displaystyle\sum_{j=1}^{N}\Big[\int_0^1 |\boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}_j))|^2 \mathrm{d}\tau + 2\Delta t V(\boldsymbol{T}(1,\boldsymbol{x}_j)) \\ \qquad\qquad\qquad\qquad + \dfrac{2\Delta t}{N}\displaystyle\sum_{l=1}^{N} W(\boldsymbol{T}(1,\boldsymbol{x}_j),\boldsymbol{T}(1,\boldsymbol{x}_l))\Big] \\ \text{s.t.} \quad \dfrac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{T}(\tau,\boldsymbol{x}_j) = \boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}_j)), \qquad \boldsymbol{T}(0,\boldsymbol{x}_j) = \boldsymbol{x}_j\,. \end{cases} \tag{14}$$

Case 2 : (8) with (11)

$$\begin{cases} \min\limits_{\boldsymbol{v}} \dfrac{1}{N}\displaystyle\sum_{j=1}^{N}\Big[\int_0^1 |\boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}_j))|^2 \mathrm{d}\tau + 2\Delta t U\left(\dfrac{\rho^n(\boldsymbol{x}_j)}{|\det\nabla_{\boldsymbol{x}}\boldsymbol{T}(1,\boldsymbol{x}_j)|}\right)\dfrac{|\det\nabla_{\boldsymbol{x}}\boldsymbol{T}(1,\boldsymbol{x}_j)|}{\rho^n(\boldsymbol{x}_j)}\Big] \\ \text{s.t.} \quad \dfrac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{T}(\tau,\boldsymbol{x}_j) = \boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}_j)), \qquad \boldsymbol{T}(0,\boldsymbol{x}_j) = \boldsymbol{x}_j \\ \qquad \dfrac{\mathrm{d}}{\mathrm{d}\tau}\log|\det\nabla_{\boldsymbol{x}}\boldsymbol{T}(\tau,\boldsymbol{x}_j)| = \mathrm{div}(\boldsymbol{v})(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}_j))\,, \quad \log|\det\nabla_{\boldsymbol{x}}\boldsymbol{T}(0,\boldsymbol{x}_j)| = 0\,, \end{cases} \tag{15}$$

In cases where the energy solely comprises potential and interaction components, it suffices to monitor the particle positions. However, when internal energy is introduced, it becomes necessary also to track the density ρ, which is accomplished by monitoring the logarithmic determinant of the transport map.

More precisely, denote the minimizer of either problem $\boldsymbol{T}^{n+1}$, then

$$\boldsymbol{T}^{n+1}(1,\boldsymbol{x}_j^n) =: \boldsymbol{x}_j^{n+1}\,,$$

which can be viewed as samples from $\rho^{n+1}(\boldsymbol{x})$. Therefore, starting from $\{\boldsymbol{x}_j^0\}$, we have a sequence of update

$$\{\boldsymbol{x}_j^0\} \xrightarrow{\boldsymbol{T}^1} \{\boldsymbol{x}_j^1\} \xrightarrow{\boldsymbol{T}^2} \{\boldsymbol{x}_j^2\} \xrightarrow{\boldsymbol{T}^3} \{\boldsymbol{x}_j^3\}\cdots, \tag{16}$$

where $\boldsymbol{x}_j^n = \boldsymbol{T}^n(1,\boldsymbol{x}_j^{n-1})$. Likewise, the density evolves as

$$\rho^0 \xrightarrow{\boldsymbol{T}^1} \rho^1 \xrightarrow{\boldsymbol{T}^2} \rho^2 \xrightarrow{\boldsymbol{T}^3} \rho^3\cdots, \tag{17}$$

where $\rho^n = \boldsymbol{T}^n(1,\cdot)\sharp\rho^{n-1}$.

In practice, since we only evolve particles, a major difficulty in (17) lies in the *density estimation* of ρ^n, which can be very expensive and inaccurate in high dimensions. To resolve this issue, a key observation is that we don't need the full information of the density, but only the density evaluated along the trajectory of particles!

Suppose we are given a set of samples $\boldsymbol{x}_j^0$ drawn from the known analytical expression of ρ_0. In the first JKO step, we can compute the updated density as follows:

$$\rho^1(\underbrace{\boldsymbol{T}^1(1,\boldsymbol{x}_j^0)}_{\boldsymbol{x}_j^1}) = \frac{\rho^0(\boldsymbol{x}_j^0)}{|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}^1(1,\boldsymbol{x}_j^0)|}\,, \tag{18}$$

where $\boldsymbol{T}^1$ represents the map obtained in the first JKO step. Here, $\boldsymbol{x}_j^1$ corresponds to the transformed sample after applying $\boldsymbol{T}^1$ to $\boldsymbol{x}_j^0$. Following a similar procedure, we obtain the density in the subsequent JKO steps:

$$\rho^2(\underbrace{\boldsymbol{T}^2(1,\boldsymbol{x}_j^1)}_{\boldsymbol{x}_j^2}) = \frac{\rho^1(\boldsymbol{x}_j^1)}{|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}^2(1,\boldsymbol{x}_j^1)|}\,, \tag{19}$$

and this iterative process can be continued for any subsequent JKO step.

Remark 1. *We emphasize that this observation holds true for general nonlinear internal potential functions, denoted as U. In the existing literature, U is commonly assumed to be the negative Boltzmann-Shannon entropy, given by $U(z) = z\log z$. In this particular case, the aforementioned difficulty can be circumvented. Indeed, since*

$$U\left(\frac{\rho^n(\boldsymbol{x}_j)}{|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(1,\boldsymbol{x}_j)|}\right)\frac{|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(1,\boldsymbol{x}_j)|}{\rho^n(\boldsymbol{x}_j)} = \log \rho^n(\boldsymbol{x}_j) - \log|\det \nabla_x \boldsymbol{T}(1,\boldsymbol{x}_j)|,$$

we can simplify the computation by avoiding the need to directly calculate $\rho^n(\boldsymbol{x}_j)$. This is because $\log \rho^n(\boldsymbol{x}_j)$ is separated from $\log|\det(\nabla_{\boldsymbol{x}} \boldsymbol{T}(1,\boldsymbol{x}_j))|$ in equation (15)*, enabling us to exclude it from the calculation. However, for general potential functions, the computation of $\rho^n(\boldsymbol{x}_j)$ cannot be circumvented.*

2.3. Nonlinear mobility

In this section, we expand upon the previous method to encompass the generalized gradient flow, as delineated by the following equation:

$$\partial_t \rho = \nabla_{\boldsymbol{x}} \cdot \left(M(\rho)\nabla_{\boldsymbol{x}} \frac{\delta \mathcal{E}(\rho)}{\delta \rho}\right). \tag{20}$$

Here, the function $M(\rho) \geq 0$ represents a nonlinear mobility function. This type of equation appears in various contexts, such as thin films [3], phase separation in binary alloys described by the Cahn-Hilliard equation [6], and the transport phenomena of biological systems with overcrowding prevention [5], among others.

For general mobility function $M(\rho)$, equation (20) can also can also be interpreted as a gradient flow, especially when $M(\rho)$ is concave and satisfies

some other properties in [16, 8]. The generalized metric also admits a dynamic formulation, leading to the following extension of the dynamic JKO scheme (5):

$$\begin{cases} (\rho, \tilde{\boldsymbol{v}}) = \arg\inf\limits_{\rho, \tilde{\boldsymbol{v}}} \int_0^1 \int_{\mathbb{R}^d} M(\rho)|\tilde{\boldsymbol{v}}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}\tau + 2\Delta t \mathcal{E}(\rho(1,\cdot)) \\ \text{s.t.} \quad \partial_\tau \rho + \nabla \cdot (M(\rho)\tilde{\boldsymbol{v}}) = 0, \ \rho(0,\boldsymbol{x}) = \rho^n(\boldsymbol{x}) \,. \end{cases} \tag{21}$$

In the above formulation, the minimization is over density function $\rho(\tau, \boldsymbol{x})$ for $\tau \in (0,1)$, terminal time density $\rho(1,\boldsymbol{x})$, and vector fields $\tilde{\boldsymbol{v}}(\tau,\boldsymbol{x})$, such that the nonlinear mobility induced continuity equation holds.

Let $\boldsymbol{v} = \frac{M(\rho)}{\rho}\tilde{\boldsymbol{v}}$, equation (21) can be rewritten into

$$\begin{cases} (\rho, \boldsymbol{v}) = \arg\inf\limits_{\rho, \boldsymbol{v}} \int_0^1 \int_{\mathbb{R}^d} \frac{\rho^2}{M(\rho)}|\boldsymbol{v}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}\tau + 2\Delta t \mathcal{E}(\rho(1,\cdot)) \,, \\ \text{s.t.} \quad \partial_\tau \rho + \nabla \cdot (\rho\boldsymbol{v}) = 0, \ \rho(0,\boldsymbol{x}) = \rho^n(\boldsymbol{x}) \,. \end{cases} \tag{22}$$

This reformulation states that the minimization is constrained by the classical continuity equation. It is worth noting that such reformulation does not change the optimizer of the problem. Indeed, let ϕ be the Lagrangian multiplier, then one can check that the critical point system of variational problems (21) and (22) lead to the same Hamilton-Jacobi equation for ϕ:

$$\partial_\tau \phi + \frac{1}{4} M'(\rho)|\nabla \phi|^2 = 0 \,.$$

We note that variational problem (22) now admits a similar Lagrangian representation as in (8), with the only distinction being the first term, which now becomes:

$$\int_0^1 \int_{\mathbb{R}^d} \frac{\rho^2(\tau,\boldsymbol{x})}{M(\rho(\tau,\boldsymbol{x}))}|\boldsymbol{v}(\tau,\boldsymbol{x})|^2 \mathrm{d}\tau\mathrm{d}\boldsymbol{x} \,.$$

By employing the same technique as utilized in deriving (11), the integral with respect to $\boldsymbol{x}$ in the given expression rewrites as

$$\begin{aligned} &\int_{\mathbb{R}^d} \frac{\rho^2(\tau,\boldsymbol{x})}{M(\rho(\tau,\boldsymbol{x}))}|\boldsymbol{v}(\tau,\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} \\ =& \int_{\mathbb{R}^d} \frac{\rho^2(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))}{M(\rho(\tau,\boldsymbol{T}(\tau,\boldsymbol{x})))}|\boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))|^2 |\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau,\boldsymbol{x})| \mathrm{d}\boldsymbol{x} \\ =& \int_{\mathbb{R}^d} \frac{\rho^2(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))}{M(\rho(\tau,\boldsymbol{T}(\tau,\boldsymbol{x})))}|\boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))|^2 \frac{|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau,\boldsymbol{x})|}{\rho^n(\boldsymbol{x})} \rho^n(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ =& \int_{\mathbb{R}^d} \frac{\rho^2(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))}{M(\rho(\tau,\boldsymbol{T}(\tau,\boldsymbol{x})))} \frac{|\boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))|^2}{\rho(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))} \rho^n(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ =& \int_{\mathbb{R}^d} \frac{\rho(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))}{M(\rho(\tau,\boldsymbol{T}(\tau,\boldsymbol{x})))}|\boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{x}))|^2 \rho^n(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \,. \end{aligned} \tag{23}$$

As in (18), the density $\rho(\tau, \mathbf{T}(\tau, \boldsymbol{x}))$ is determined by the Monge-Amperè equation:

$$\rho(\tau, \boldsymbol{T}(\tau, \boldsymbol{x})) = \frac{\rho^n(\boldsymbol{x})}{|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x})|}.$$

Similar to the JKO scheme presented in (14) or (15), given $\{\boldsymbol{x}_j^n\}_j$ and $\{\rho^n(\boldsymbol{x}_j^n)\}_j$ (we omit the superscript n in $\boldsymbol{x}_j^n$ in the following formula), the updated formulation can now be expressed as follows:

$$\begin{cases} \min\limits_{\boldsymbol{T}} \dfrac{1}{N}\sum\limits_{j=1}^{N}\left[\displaystyle\int_0^1 \frac{\rho(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}_j))}{M(\rho(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}_j)))}|\boldsymbol{v}(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}_j))|^2 \mathrm{d}\tau + 2\Delta t \frac{U(\rho(1, \boldsymbol{T}(1, \boldsymbol{x}_j)))}{\rho(1, \boldsymbol{T}(1, \boldsymbol{x}_j))}\right] \\ \text{s.t.} \quad \dfrac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{T}(\tau, \boldsymbol{x}_j) = \boldsymbol{v}(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}_j)), \qquad \boldsymbol{T}(0, \boldsymbol{x}_j) = \boldsymbol{x}_j \\ \qquad \dfrac{\mathrm{d}}{\mathrm{d}\tau}\log|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x}_j)| = \mathrm{div}(\boldsymbol{v})(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}_j)), \quad \log|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(0, \boldsymbol{x}_j)| = 0 \\ \qquad \rho(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}_j)) = \dfrac{\rho^n(\boldsymbol{x}_j)}{|\det \nabla_{\boldsymbol{x}} \boldsymbol{T}(\tau, \boldsymbol{x}_j)|}. \end{cases} \tag{24}$$

It is obvious that when $M(\rho) = \rho$, (24) reduces to (15).

3. Neural ODE empowered JKO schemes

In this section, we leverage neural network functions to approximate the vector field $\boldsymbol{v}$ as described in equations (14), (15), or, more broadly, equation (24). As a result, the optimization procedure aims at refining the parameters of the neural network. Building upon the efficiency gains achieved through the use of continuous normalizing flow for density estimation, as previously discussed, we incorporate a recursive relation formula proposed in a prior work [34] to compute derivatives, significantly improving the efficiency of backpropagation. Lastly, we conclude this section by presenting a concise summary of the algorithm.

3.1. Learning the potential function

In view of (14) and (15), the primary objective is to determine the optimal transport map, denoted as $\boldsymbol{T}$. However, this task can be computationally demanding, especially in high-dimensional scenarios. Fortunately, by applying the principles of optimal transport [20, 40], we can express the velocity field $\boldsymbol{v}(t, \boldsymbol{x})$ as the gradient of a potential function $\phi(t, \boldsymbol{x})$ such that $\boldsymbol{v}(t, \boldsymbol{x}) = -\nabla_{\boldsymbol{x}}\phi(t, \boldsymbol{x})$. Consequently, our focus shifts to finding the potential function ϕ. To achieve this, we utilize a neural network to approximate $\phi(t, \boldsymbol{x})$, denoted as $\phi_\theta(t, \boldsymbol{x})$. As a result, we obtain $\boldsymbol{v}_\theta(t, \boldsymbol{x}) = -\nabla_{\boldsymbol{x}}\phi_\theta(t, \boldsymbol{x})$ as the corresponding approximation of the velocity field.

To be more specific, let us consider solving equation (14). By substituting $\boldsymbol{v}$ with $\boldsymbol{v}_\theta = -\nabla_{\boldsymbol{x}}\phi_\theta$, the constraint in (14) becomes:

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{T}(\tau, \boldsymbol{x}_j) = -\nabla_{\boldsymbol{x}}\phi_\theta(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}_j)), \qquad \boldsymbol{T}(0, \boldsymbol{x}_j) = \boldsymbol{x}_j.$$

Similarly, in the case of the more complex equation (15), computing $\rho(\boldsymbol{T}(1, \boldsymbol{x}_j))$ calls for the following update:

$$\frac{\mathrm{d}}{\mathrm{d}\tau} \log \left|\det(\nabla^2_{\boldsymbol{x}} \phi_\theta(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}_j)))\right| = \mathrm{div}(\boldsymbol{v}_\theta)(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}_j)) = -\mathrm{tr}(\nabla^2_{\boldsymbol{x}} \phi_\theta(\tau, \boldsymbol{T}(\tau, \boldsymbol{x}_j))).$$

Detailed computations of $\nabla^2 \phi_\theta$ will be given in the next subsection.

3.2. Deep JKO Algorithms

We hereby provide comprehensive details for computing the minimizer of (14), (15) or (24), when $\boldsymbol{v}$ is replaced by $\boldsymbol{v}_\theta$. Let N_τ denote the number of time discretization such that $[0, 1]$ is discretized into $0, \frac{1}{N_\tau}, \frac{2}{N_\tau}, \cdots, , \frac{N_\tau - 1}{N_\tau}$, and θ is a vector of parameters for neural network functions. Given the density ρ^n at the n-th JKO step and sampled points $\{\boldsymbol{x}_j\}_{j=1}^N$, the loss function to compute the density at the $n+1$-th JKO step is defined as follows:

$$L(\theta, \{\boldsymbol{x}_j\}_{j=1}^N) = \frac{1}{N} \sum_{j=1}^{N} \Bigg[d\tau \sum_{k=1}^{N_\tau} \left|\nabla_{\boldsymbol{x}} \phi_\theta(kd\tau, \mathbf{z}_j^k)\right|^2 + 2\Delta t \left(\frac{U(\rho(\mathbf{z}_j^{N_\tau}))}{\rho(\mathbf{z}_j^{N_\tau})} + V(\mathbf{z}_j^{N_\tau}) + \frac{1}{N} \sum_{l=1}^{N} W(\mathbf{z}_j^{N_\tau}, \mathbf{z}_l^{N_\tau}) \right) \Bigg], \tag{25}$$

where the inner time step is denoted as $d\tau = 1/N_\tau$ and the update equations are given by:

$$\begin{aligned} \mathbf{z}_j^{k+1} &= \mathbf{z}_j^k - d\tau \, \nabla_{\boldsymbol{x}} \phi_\theta(kd\tau, \mathbf{z}_j^k), \quad \mathbf{z}_j^0 = \boldsymbol{x}_j, \\ l_j^{k+1} &= l_j^k - d\tau \, \mathrm{tr}(\nabla^2_{\boldsymbol{x}} \phi_\theta(kd\tau, \mathbf{z}_j^k)), \quad l_j^0 = 0, \end{aligned}$$

for $k = 0, \cdots, N_\tau - 1$ and the terminal density ρ at the location $\boldsymbol{z}_j^{N_\tau}$ is defined as

$$\rho(1, \boldsymbol{z}_j^{N_\tau}) = \frac{\rho^n(\boldsymbol{x}_j)}{\exp(l_j^{N_\tau})}. \tag{26}$$

The above equations for $\mathbf{z}_j^{k+1}$ and $\log \rho(\mathbf{z}_j^{k+1})$ utilize the forward Euler method to solve the ordinary differential equations (ODEs) in the constraints of (15). However, alternative ODE solvers such as Runge-Kutta can also be employed. For our numerical experiment, we utilize the RK4 as the ODE solver.

The loss function, denoted as $L(\theta, \{\boldsymbol{x}_j\}_{j=1}^N)$, takes two inputs: the neural network parameter θ and a set of sampled points $\{\boldsymbol{x}_j\}_{j=1}^N$. The objective is to minimize this loss function with respect to θ. This θ plays a vital role in approximating the potential function $\phi_\theta(\boldsymbol{x}, \tau)$, which relies on two variables: a state vector $\boldsymbol{x} \in \mathbb{R}^d$ and a time value $t \in [0, 1]$. The velocity is computed as:

$$\boldsymbol{v}_\theta(\tau, \boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \phi_\theta(\tau, \boldsymbol{x}),$$

and its divergence is obtained as the trace of the Hessian:

$$\mathrm{div}(\boldsymbol{v}_\theta)(\tau, \boldsymbol{x}) = -\mathrm{tr}(\nabla^2_{\boldsymbol{x}} \phi_\theta(\tau, \boldsymbol{x})).$$

To simplify the calculation of gradients and Hessians in the formulas above, we utilize the explicit expression introduced in [34]. This approach capitalizes on the smoothness of the activation function within the neural network. Consequently, our optimization process experiences significant acceleration, as we no longer rely on automatic differentiation (AD) for backpropagation to compute gradients and Hessians in each iteration.

More precisely, we adopt a multi-layer ResNet that takes the input $\mathbf{s} = (\tau, \boldsymbol{x}) \in \mathbb{R}^{d+1}$, where $\boldsymbol{x} \in \mathbb{R}^d$ represents the initial state, and $\tau \in [0, 1]$ denotes time, yielding an output in $\mathbb{R}$. The neural network function consists of L hidden layers with recursive relations defined as follows:

$$\begin{aligned} \mathbf{u}_1 &= \sigma(W_0\mathbf{s} + \mathbf{b}_0), \\ \mathbf{u}_{l+1} &= \mathbf{u}_l + \sigma(W_l\mathbf{u}_l + \mathbf{b}_l), \quad l = 1, \cdots L-1, \\ NN(\mathbf{s};\theta) &= \boldsymbol{w}^\top \mathbf{u}_L. \end{aligned}$$

Here, $\theta = (W_0, W_l, b_l)$, with $W_0 \in \mathbb{R}^{m\times(d+1)}$ and $W_l \in \mathbb{R}^{m\times m}$ $(l = 1, \cdots, L)$ are dense matrices. Additionally, $b_k \in \mathbb{R}^m$ $(l = 0, \cdots, L)$ are biases, while m denotes the number of nodes in each hidden layer and $\boldsymbol{w} \in \mathbb{R}^m$ is a vector we perform dot product at the end of the ResNet. The element-wise activation function $\sigma(\boldsymbol{x}) = \log(\exp(\boldsymbol{x}) + \exp(-\boldsymbol{x}))$ is used, which is differentiable. The dual variable ϕ_θ is defined through

$$\phi_\theta(\tau, \boldsymbol{x}) = NN((\tau, \boldsymbol{x}); \theta). \tag{27}$$

Denote by $\nabla_{\boldsymbol{x}}$ a gradient operator with respect to the space variable $\boldsymbol{x}$ and $\nabla_{\mathbf{s}}$ be a gradient operator with respect to the state vector $\mathbf{s}$. Both the gradient $\nabla_{\boldsymbol{x}}\phi_\theta(\boldsymbol{x}, \tau)$ and the Hessian $\nabla^2_{\boldsymbol{x}}\phi_\theta(\boldsymbol{x}, \tau)$ can be effortlessly computed using the explicit formulas presented in [34]. Specifically, for the case where $L = 2$, these formulas are as follows:

$$\nabla_{\mathbf{s}}\phi_\theta(\tau, \boldsymbol{x}) = W_0^\top \operatorname{diag}(\sigma'(W_0\boldsymbol{s} + b_0))\boldsymbol{z}_1, \tag{28a}$$

$$\boldsymbol{z}_1 = \boldsymbol{w} + W_1^\top \operatorname{diag}(\sigma'(W_1\boldsymbol{u}_1 + b_1))\boldsymbol{w}, \tag{28b}$$

and

$$\nabla^2_{\boldsymbol{s}}\phi_\theta(\boldsymbol{x}, \tau) = t_0 + t_1, \tag{28c}$$

where

$$\begin{aligned} t_0 &= W_0^\top \operatorname{diag}(\sigma''(W_0\boldsymbol{s} + b_0) \odot \boldsymbol{z}_1)W_0, \\ t_1 &= J_0^\top W_1^\top \operatorname{diag}(\sigma''(W_1\boldsymbol{u}_1 + b_1) \odot \boldsymbol{w})W_1 J_0, \\ J_0 &= W_0^\top \operatorname{diag}(\sigma'(W_0\boldsymbol{s} + b_0))^\top. \end{aligned}$$

Here $\odot$ represents the element-wise multiplication. Then, $\nabla_{\boldsymbol{x}}\phi_\theta(\boldsymbol{x}, \tau)$ is the first d elements of $\nabla_{\mathbf{s}}\phi_\theta(\tau, \boldsymbol{x})$, and $\nabla^2_{\boldsymbol{x}}\phi_\theta(\tau, \boldsymbol{x}) = [\nabla^2_{\mathbf{s}}\phi_\theta(\tau, \boldsymbol{x})]_{1:d,1:d}$ where the subscript $1:d, 1:d$ denotes the submatrix of size $d \times d$. For comprehensive expressions of the gradient and Hessian operators for a general L, we recommend readers to see details in [34].

We now provide a summary of the proposed algorithms. Algorithm 2 serves as the primary algorithm, guiding the movement of particles along the gradient flow. Whenever sampling/resampling is necessary, Algorithm 1 will be invoked. Please note that the l update and density computation in Algorithm 1 are only required when working with general nonlinear internal energy and mobility. Additionally, in the algorithm, we utilize forward Euler, but it is adaptable for replacement with explicit Runge-Kutta type ODE solvers.

Algorithm 1 Generate samples and densities at n-th JKO step

Input:

- Outer JKO time step Δt.
- Inner dynamic formulation time step $d\tau$ and total number of inner steps N_τ.
- Initial distribution ρ^0 and initial samples $\{\boldsymbol{x}_j^0\}_{j=1}^N$ and densities $\{\rho(\boldsymbol{x}_j^0)\}_{j=1}^N$
- Potential functions from previous steps $(\phi_\theta^0, \cdots, \phi_\theta^{n-1})$.

Output:

- Sampled points $\{\boldsymbol{x}_j^n\}_{j=1}^N$ at n-th outer iteration.
- Density values $\{\rho(\boldsymbol{x}_j^n)\}_{j=1}^N$ at n-th outer iteration.

for $i = 0, \cdots, n-1$ **do**
  for $j = 1, \cdots, N$ **do**
    $\mathbf{z}_j^0 = \boldsymbol{x}_j^i$, $l_j^0 = 0$
    for $k = 0, \cdots, N_\tau - 1$ **do**
      $\mathbf{z}_j^{k+1} = \mathbf{z}_j^k - d\tau \, \nabla_{\boldsymbol{x}} \phi_\theta^i(kd\tau, \mathbf{z}_j^k)$
      $l_j^{k+1} = l_j^k - d\tau \, \mathrm{tr}(\nabla_{\boldsymbol{x}}^2 \phi_\theta^i(kd\tau, \mathbf{z}_j^k))$
    end for
    $\boldsymbol{x}_j^{i+1} = \mathbf{z}_j^{N_\tau}$, $\rho(\boldsymbol{x}_j^{i+1}) = \frac{\rho^n(\boldsymbol{x}_j)}{\exp(l_j^{N_\tau})}$
  end for
end for

Algorithm 2 Deep JKO scheme

Input:

- Outer JKO time step Δt
- Inner dynamic formulation time step $d\tau$
- Initial distribution ρ^0
- Learning rate α for the Adam optimizer
- Total number of outer iterations K
- Resampling frequency C

Output:

- Neural network parameter θ and corresponding potential function $\phi_\theta^n(t, \boldsymbol{x})$, $n = 1, 2, \cdots, K$
- Particle locations: $\{\boldsymbol{x}_j^n\}$, $n = 1, 2, \cdots, K$, $j = 1, 2, \cdots, N$.
- Density along particle trajectories: $\rho^n(\boldsymbol{x}_j^n)$, $n = 1, 2, \cdots, K$, $j = 1, 2, \cdots, N$.

Initialize θ from a standard normal distribution.

for $n = 1, \cdots, K$ **do**
  $c = 0$.
  while not converged **do**
    if $mod(c, C) = 0$ **then do**
      Sample $\{\boldsymbol{x}_j^0\}_{j=1}^N \overset{\text{i.i.d.}}{\sim} \rho^0$
      Compute $\{\boldsymbol{x}_j^n\}_{j=1}^N$ using Algorithm 1 and update the density value through (26).
    end if
    Update $\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta, \{\boldsymbol{x}_j^n\}_{j=1}^N)$
    where L is from (25) and the gradient is computed using automatic differentiation (AD).
    $c \leftarrow c + 1$
  end while
end for

4. Numerical examples

This section presents several numerical examples of the proposed deep JKO schemes for computing gradient flows with various choices of energy functionals and mobility functions.

4.1. Wasserstein gradient flow of the KL divergence

In this experiment, we present the computed solutions of (15) with the energy functional defined as

$$U(\rho) = \mathrm{D}_{\mathrm{KL}}(\rho\|q) = \int \rho(\boldsymbol{x}) \log \frac{\rho(\boldsymbol{x})}{q(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}, \tag{29}$$

where q represents a reference density. The initial density ρ^0 is characterized by two Gaussian distributions centered at $(\pm 1.2, 0)$ with a standard deviation of $\sigma = 0.5$. The reference density q is defined as four Gaussian distributions centered at $(\pm 2, \pm 2)$ each with a covariance matrix represented by a diagonal matrix with every diagonal entry equal to 0.25.

For the numerical experiment, the inner timestep is set as $N_\tau = 1$ and the JKO time stepsize is set as $\Delta t = 0.025$. The model architecture includes 3 layers with $m = 64$ nodes in the hidden layers. The training is performed using a batch size of 1000, and the learning rate is chosen to be 10^{-5}. The total number of iterations is set to $10,000$.

The two-dimensional results are depicted in Figure 1 and Figure 2. Figure 1 illustrates the evolution of the density from $t = 0$ to $t = 0.4$. In this figure, the color of the points represents the value of the density $\rho^n(\boldsymbol{x})$ at the corresponding location x. Brighter colors indicate higher density values. The computation involves a total of 16 JKO steps. Figure 2 displays the trajectories of each particle with respect to time t and the energy value computed from the algorithm with respect to the iterations. As the iterations progress, the energy decreases and eventually converges to the stationary value.

We further extended our experiments to a 10-dimensional space while retaining the same energy functional (29). The initial density ρ^0 was set as a Gaussian distribution centered at the origin, and the reference density q was constructed using a combination of four Gaussian distributions centered at $(2, 2.5)$, $(-2.5, 2)$, $(-2, -2.5)$, and $(2.5, -2)$ each with a covariance matrix represented by a diagonal matrix with every diagonal entry equal to 0.25. In this experimental setup, we use a time step size of $\Delta\tau = 0.2$, a learning rate of 10^{-5}, and executed a total of 10,000 iterations. The results are illustrated in Figure 3, highlighting the initial 2D cross sections of the density's evolution from $t = 0$ to $t = 3.4$. These visual representations present kernel density plots derived from the point clouds generated by the algorithm at each time instance t.

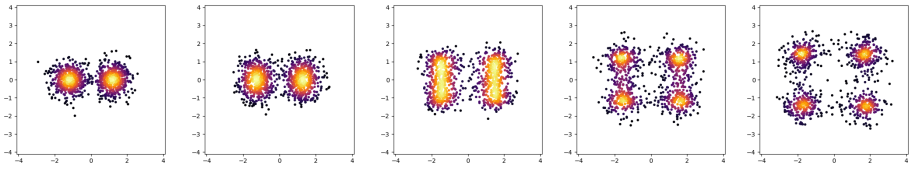
4.2. Porous medium equation

The second experiment concerns the porous medium equation represented by

$$\partial_t \rho(t, \boldsymbol{x}) = \Delta \rho(t, \boldsymbol{x})^m, \quad m > 1\,, \boldsymbol{x} \in \mathbb{R}^d\,, \tag{30}$$

which can be interpreted as the Wasserstein gradient flow with the internal energy functional U defined as

$$U(\rho) = \int_{\mathbb{R}^d} \frac{1}{m-1} \rho(\boldsymbol{x})^m d\boldsymbol{x}.$$



(a) $t = 0$ (b) $t = 0.025$ (c) $t = 0.1$ (d) $t = 0.2$ (e) $t = 0.4$

Figure 1: Computed solutions of 2D Fokker-Planck equation, i.e. the Wasserstein gradient flow of KL divergence (29). An initial density is a mixture of two Gaussians and the target density is a mixture of four Gaussians. The figures visually depict the evolution of these densities from $t = 0$ to $t = 0.4$.

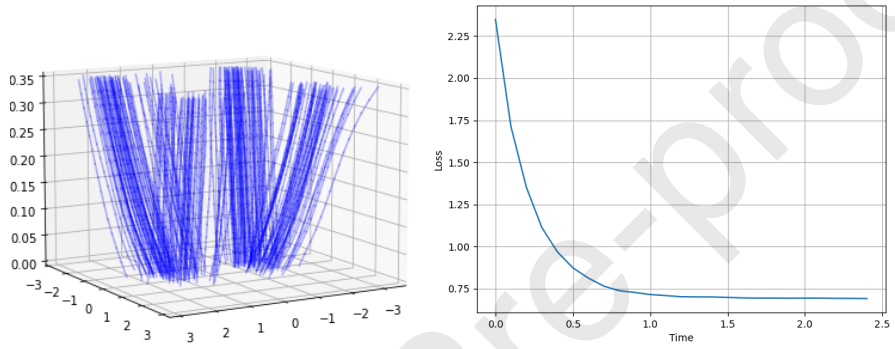
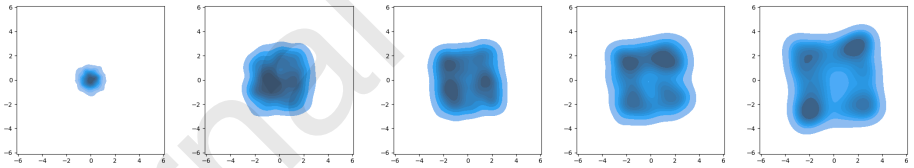


Figure 2: 3D plot (left) and the energy decay plot (right) from Figure 1. The left plot shows the trajectories of each particles where z-axis represents time from $t = 0$ to $t = 0.35$. The right plot illustrates the decay of the energy from $t = 0$ to $t = 2.4$.



(a) $t = 0$ (b) $t = 0.6$ (c) $t = 1$ (d) $t = 2$ (e) $t = 3.4$

Figure 3: Evolution of densities from 10D Fokker-Planck equation from $t = 0$ to $t = 3.4$. An initial density is a Gaussian distribution centered at the origin and the target density is a mixture of four Gaussians.

This equation admits a close-form solution given by the Barenblatt formula:

$$\rho(t, \boldsymbol{x}) = (t + t_0)^{-\alpha} \left(C - \beta \|\boldsymbol{x}\|^2 (t + t_0)^{-\frac{2\alpha}{d}} \right)_+^{\frac{1}{m-1}}, \tag{31}$$

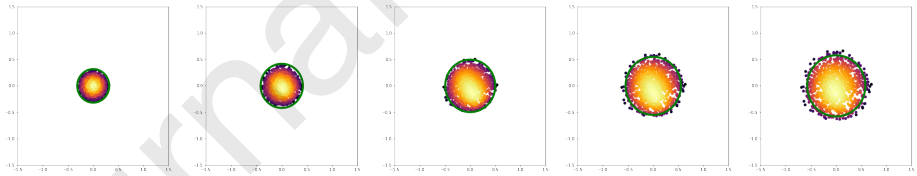
where $\alpha = \frac{d}{d(m-1)+2}$, $\beta = \frac{(m-1)\alpha}{2dm}$, C is a constant that makes $\int_{\mathbb{R}^d} \rho(t, \boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1$. We choose $t_0 = 10^{-3}$. Formula (31) serves as a reference solution to evaluate the performance of our algorithm.

In the numerical experiment, we employ an inner timestep of $N_\tau = 2$ and set the JKO time step size to $\Delta t = 0.001$. The model architecture consists of 3 layers with 128 nodes in each hidden layer. During training, a batch size of 1000 is used, and the learning rate is set at 10^{-5}. The total number of iterations for the experiment is configured to be $10,000$.

The experimental results are presented in Figure 4 and Figure 5. Figure 4 visualizes the density evolution from $t = 0$ to $t = 0.12$, where particles positions $\boldsymbol{x}$ and their corresponding density values, $\rho(t, \boldsymbol{x})$, are accurately computed at each time step. The color intensity at each particle's location represents its density, with brighter colors indicating higher density values. Notably, the computed solutions maintain radial symmetry, providing a precise representation. The green rings in the figures correspond to the support of the analytical solution (31) and align well with the computed solutions, affirming the algorithm's accuracy. Figure 5 displays particle trajectories from $t = 0$ to $t = 0.01$, exhibiting linear trajectories as expected from the analytical solution of the PDE.

We then utilize the algorithm to compute solutions for the porous medium equation in high-dimensional spaces, specifically in dimensions 6 and 10. For both cases, we choose an outer time step size of $\Delta t = 0.0005$, and set the inner timestep to $N_\tau = 2$. The model architecture comprises three layers with 128 nodes in the hidden layers. Training is carried out using a batch size of 1000, and the learning rate is fixed at 10^{-6}. The total number of iterations is set to be $20,000$.

Figure 6 illustrates the evolution of densities in both 6D and 10D. To visualize the results and demonstrate the accuracy of the algorithm, 2D cross-section plots of the particles are presented, along with the support of the densities obtained from the analytical solution, represented as a green ring. The computed densities exhibit spherical evolution and closely match the analytical solutions.



(a) $t = 0$ (b) $t = 0.002$ (c) $t = 0.005$ (d) $t = 0.008$ (e) $t = 0.01$

Figure 4: Computed solutions of 2D porous medium equation. The figures show the evolution of densities from $t = 0$ to $t = 0.12$. The algorithm calculates the positions of particles $\boldsymbol{x} \in \mathbb{R}^2$ and their corresponding density values $\rho(t, \boldsymbol{x})$ at each time t. The color of each particle represents its density value, with brighter colors indicating higher density values. The green ring indicates the support of the density from the analytical solution at the given time t.

Here, we discuss the complexity and accuracy of our scheme. For 2D and 6D cases, we use 4-layer fully connected neural network with 512 nodes per layer, and for 10D case, we use 4-layer fully connected neural network with 1024 nodes per layer. For all cases, we set the number of iterations to be 50,000 during the first outer iteration and 5,000 iterations thereafter. The cost per iteration is

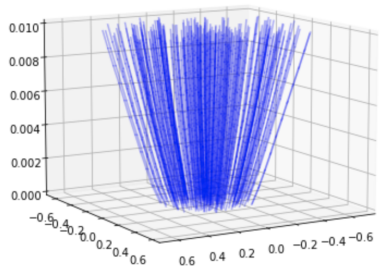
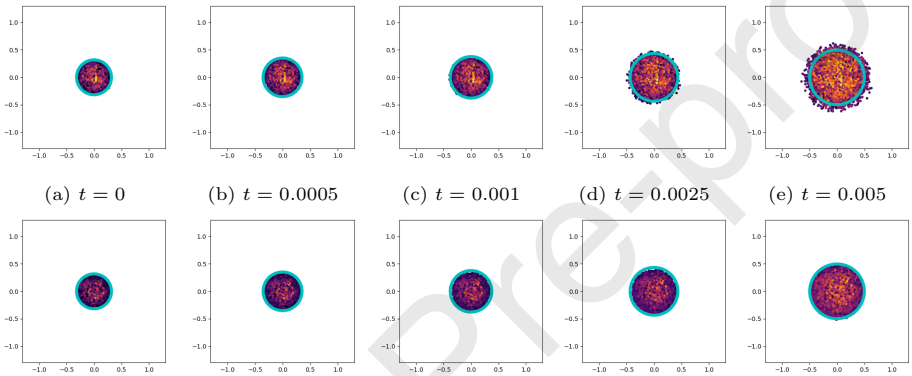


Figure 5: 3D plot from Figure 4. It shows the trajectories of each particles where z-axis represents time from $t = 0$ to $t = 0.01$.



(a) $t = 0$ (b) $t = 0.0005$ (c) $t = 0.001$ (d) $t = 0.0025$ (e) $t = 0.005$

(f) $t = 0$ (g) $t = 0.0005$ (h) $t = 0.001$ (i) $t = 0.0025$ (j) $t = 0.005$

Figure 6: Computed solutions of 6D (first row) and 10D (second row) porous medium equation. The figures show the evolution of densities from $t = 0$ to $t = 0.005$ for 6D and 10D porous medium flows. The green ring indicates the support of the density from the analytical solution at the given time t.

however the same acorss different dimensions, approximately 1 to 2 minutes per 1000 iterations on 2 GPUs. We evaluate the accuracy by comparing the computed density using the density update formula with the exact solution. Figure 7 illustrates this comparison for a dataset $\{\boldsymbol{x}_j\}_{j=1}^N$ with $N = 50,000$. The L^2 error is calculated as:

$$L^2 \text{ error} = \left(\frac{1}{N} \sum_{j=1}^{N} |\rho_{\text{computed}}(x_j^k) - \rho_{\text{exact}}(x_j^k)|^2 \right)^{1/2}.$$

4.3. Nonlocal transport equation with nonlinear mobility

In this section, we explore an example that involves nonlinear mobility, a phenomenon frequently encountered in the study of transport in biological systems, where it serves to prevent overcrowding [5]. Particularly, we examine a

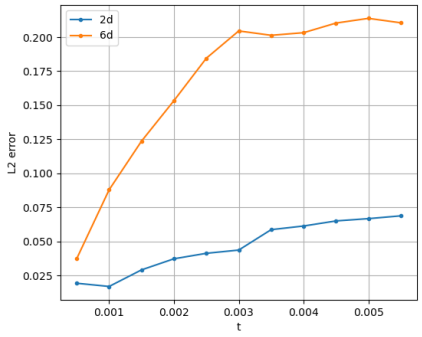


Figure 7: L^2 error from porous medium equation experiments for 2D and 6D results

specific mobility function, represented as $M(\rho) = \rho(1-\rho)$, and a particular energy functional described by the equation:

$$\mathcal{E}(\rho) = \frac{1}{2}\int_{\mathbb{R}^d\times\mathbb{R}^d} W(\boldsymbol{x}-\boldsymbol{y})\rho(\boldsymbol{x})\rho(\boldsymbol{y})\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{y}\,, \quad W(\boldsymbol{x}) = |\boldsymbol{x}|^2\,. \tag{32}$$

The corresponding equation can be expressed as:

$$\partial_t\rho(t,\boldsymbol{x}) = \nabla\cdot\Big(M(\rho(t,\boldsymbol{x}))\nabla\frac{\delta\mathcal{E}}{\delta\rho}(t,\boldsymbol{x})\Big)\,. \tag{33}$$

This model is based on [17]. Beginning with an initial condition $\rho_0(\boldsymbol{x}) = \frac{3}{4}(1-|\boldsymbol{x}|^2)_+$, it is anticipated that ρ_∞ will take the form of a characteristic function whose support is determined by the total mass. The experimental results are presented in Figure 8. The top row of figures illustrates particle evolution from a bird's-eye view, where each particle's position $\boldsymbol{x}$ is color-coded according to its associated density value, $\rho^n(\boldsymbol{x})$. The bottom row of figures presents a side view of the density values for each particle when their locations are projected onto the x-axis. These visualizations provide compelling evidence of the density converging toward a characteristic function.

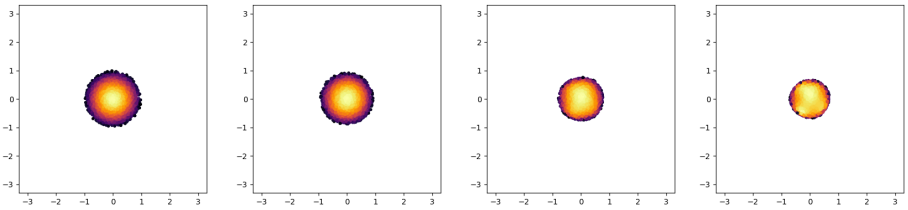
4.4. Kalman-Wasserstein gradient flow

In the last example, we consider the Kalman-Wasserstein gradient flow [21]:

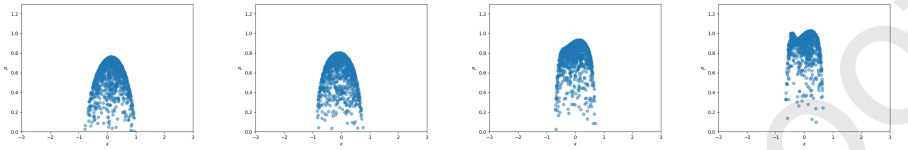
$$\partial_t\rho(t,\boldsymbol{u}) = \nabla_{\boldsymbol{u}}\cdot\Big(\rho(t,\boldsymbol{u})C(\rho)\nabla_{\boldsymbol{u}}\frac{\delta\mathcal{E}}{\delta\rho}(t,\boldsymbol{u})\Big), \tag{34}$$

where $\boldsymbol{u}\in\mathbb{R}^d$ is the parameter in the Bayesian inverse problem,

$$C(\rho) = \int_{\mathbb{R}^d}(\boldsymbol{u}-\boldsymbol{m}(\rho))\otimes(\boldsymbol{u}-\boldsymbol{m}(\rho))\rho(\boldsymbol{u})\mathrm{d}\boldsymbol{u}\,, \quad \boldsymbol{m}(\rho) = \int_{\mathbb{R}^d}\boldsymbol{u}\rho(\boldsymbol{u})\mathrm{d}\boldsymbol{u}\,, \tag{35}$$



(a) $t = 0$ (b) $t = 0.02$ (c) $t = 0.1$ (d) $t = 0.2$



(e) $t = 0$ (f) $t = 0.02$ (g) $t = 0.1$ (h) $t = 0.2$

Figure 8: Computed solutions of nonlocal transport equation with nonlinear mobility (33). The figures show the evolution of densities from $t = 0$ to $t = 0.2$. Top row: bird eye view with color coded density. Bottom row: particle density (on the y-axis) versus its projected location on the x-axis.

and

$$\mathcal{E}(\rho) = \mathrm{D}_{\mathrm{KL}}(\rho\|\pi) = \int_{\mathbb{R}^d} \Big(\Phi(\boldsymbol{u})\rho + \rho(\boldsymbol{u})\ln\rho(\boldsymbol{u})\Big)\mathrm{d}\boldsymbol{u} + \ln Z\,. \tag{36}$$

Here ρ is a probability density function of u, $\pi = \frac{1}{Z}e^{-\Phi(\boldsymbol{u})}$ is a given target distribution and we assume $Z = \int e^{-\Phi(\boldsymbol{u})}d\boldsymbol{u} < +\infty$ is a normalization constant.

This equation is derived as a mean field limit of the Kalman Wasserstein sampler, aiming at identifying the unknown parameter $\boldsymbol{u}$ in the Bayesian setting. In particular, $\Phi(\boldsymbol{u})$ takes the form

$$\Phi(\boldsymbol{u}) = \frac{1}{2}\|\mathcal{G}(\boldsymbol{u}) - \boldsymbol{y}\|_{\Gamma}^2 + \frac{1}{2}\|\boldsymbol{u}\|_{\Gamma_0}^2\,, \tag{37}$$

where $\|\boldsymbol{u}\|_{\mathsf{A}} := \boldsymbol{u}^{\top}\mathsf{A}^{-1}\boldsymbol{u}$, $\mathcal{G}$ represents a forward map that relates the parameter $\boldsymbol{u}$ to the measurement $\boldsymbol{y}$, typically represented as a PDE operator. We assume that $\boldsymbol{u}$ follows a prior distribution $\mathcal{N}(0,\Gamma_0)$. The term $\frac{1}{2}\|\mathcal{G}(\boldsymbol{u}) - \boldsymbol{y}\|_{\Gamma}^2$ is commonly referred to as the likelihood function, and $\frac{1}{Z}e^{-\Phi(\boldsymbol{u})}$, with Z being the normalizing constant, represents the posterior distribution.

Then our goal of solving (34) is to sample from the posterior distribution. Instead of running a Lagenvin type dynamics as proposed in [21], we use our DeepJKO formulation. Note that $C(\rho)$ dose not have an explicit $\boldsymbol{u}$ dependence,

(23) simplifies to

$$\begin{aligned}
&\int_{\mathbb{R}^d} \rho(\tau,\boldsymbol{u})\boldsymbol{v}(\tau,\boldsymbol{u})^\top C(\rho(\tau,\boldsymbol{u}))^{-1}\boldsymbol{v}(\tau,\boldsymbol{u})\mathrm{d}\boldsymbol{u}\\
=&\int_{\mathbb{R}^d} \boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{u}))^\top C(\rho(\tau,\boldsymbol{T}(\tau,\boldsymbol{u})))^{-1}\boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{u}))\rho^n(\boldsymbol{u})\mathrm{d}\boldsymbol{u}\\
=&\int_{\mathbb{R}^d} \boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{u}))^\top C(\tau)^{-1}\boldsymbol{v}(\tau,\boldsymbol{T}(\tau,\boldsymbol{u}))\rho^n(\boldsymbol{u})\mathrm{d}\boldsymbol{u}\,.
\end{aligned}$$

It's important to highlight that $C(\tau) = C(\rho(\tau, \boldsymbol{T}(\tau, \boldsymbol{u})))$ is reliant on the moments of ρ and thus remains unaffected by variations in u. Consequently, the JKO scheme (24) rewrites to:

$$\begin{cases}
\min\limits_{\theta} \dfrac{1}{N}\displaystyle\sum_{j=1}^{N}\Big\{\int_0^1 \boldsymbol{v}_\theta(\tau,\boldsymbol{T}(\tau,\boldsymbol{u}_j)^\top C(\tau)^{-1}\boldsymbol{v}_\theta(\tau,\boldsymbol{T}(\tau,\boldsymbol{u}_j))\mathrm{d}\tau\\
\qquad\qquad +2\Delta t\left[\Phi(\boldsymbol{T}(1,\boldsymbol{u}_j))+\log\rho^n(\boldsymbol{u}_j)-\log\det(\nabla_u\boldsymbol{T}(1,\boldsymbol{u}_j))\right]\Big\}\,,\\
\text{where}\quad C(\tau)=\dfrac{1}{N}\displaystyle\sum_{j=1}^{N}(\boldsymbol{T}(\tau,\boldsymbol{u}_j)-\boldsymbol{m})\otimes(\boldsymbol{T}(\tau,\boldsymbol{u}_j)-\boldsymbol{m})\,,\quad \boldsymbol{m}=\dfrac{1}{N}\displaystyle\sum_{j=1}^{N}\boldsymbol{T}(\tau,\boldsymbol{u}_j)\,,\\
\text{s.t.}\quad \dfrac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{T}(\tau,\boldsymbol{u}_j)=\boldsymbol{v}_\theta(\tau,\boldsymbol{T}(\tau,\boldsymbol{u}_j))\,,\qquad \boldsymbol{T}(0,\boldsymbol{u}_j)=\boldsymbol{u}_j\,.\\
\qquad \dfrac{\partial}{\partial\tau}\log\det(\nabla_{\boldsymbol{u}}\boldsymbol{T}(\tau,\boldsymbol{u}_j))=\mathrm{div}(\boldsymbol{v})(\tau,\boldsymbol{T}(\tau,\boldsymbol{u}_j))\,,\quad \log\det(\nabla_{\boldsymbol{u}}\boldsymbol{T}(0,\boldsymbol{u}_j))=0\,.
\end{cases} \tag{38}$$

As a specific example, we consider a case where the forward map $\mathcal{G}$ is given by the one-dimensional elliptic boundary value problem:

$$-\frac{\mathrm{d}}{\mathrm{d}x}\left(e^{u_1}\frac{\mathrm{d}}{\mathrm{d}x}p_{\boldsymbol{u}}(x)\right)=1,\quad x\in[0,1]\,,$$

with boundary conditions $p_{\boldsymbol{u}}(0) = 0$ and $p_{\boldsymbol{u}}(1) = u_2$. This problem has been considered in [24, 21, 28]. The explicit solution for this problem is

$$p_{\boldsymbol{u}}(x)=u_2x+e^{-u_1}\left(-\frac{x^2}{2}+\frac{x}{2}\right)\,.$$

Then the forward map $\mathcal{G}$ is defined by

$$\mathcal{G}(\boldsymbol{u})=(p_{\boldsymbol{u}}(0.25),p_{\boldsymbol{u}}(0.75))^\top\,,\quad \text{where } \boldsymbol{u}=(u_1,u_2)^\top\,. \tag{39}$$

Then the Bayesian inverse problem is to find the distribution of the unknown $\boldsymbol{u}$ conditioned on the observation $\boldsymbol{y}$, assuming additive Gaussian noise. More precisely, we use $\Phi(\boldsymbol{u})$ defined in (37) and choose

$$\boldsymbol{y}=(27.5,79.7)^\top,\quad \Gamma=0.1^2\mathsf{I}_2,\quad \Gamma_0=10^2\mathsf{I}_2,$$

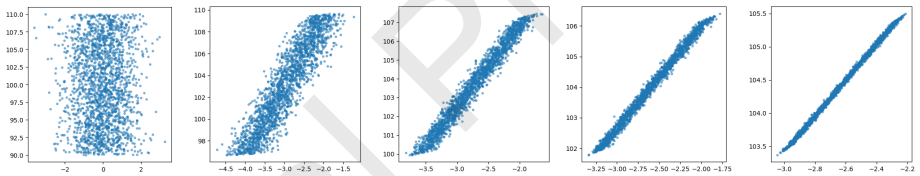
where $\mathsf{I}_2 \in \mathbb{R}^{2\times 2}$ is the identity matrix. We run the dynamics by initially sampling according to

$$\rho_0(\boldsymbol{u}) = \mathcal{N}(0,1) \times \mathcal{U}(90,110)\,, \tag{40}$$

That is, sample u_1 according to normal distribution $\mathcal{N}(0,1)$, and u_2 according to uniform distribution $\mathcal{U}(90,110)$.

In the computational approach, we set the matrix $C(\tau)^{-1}$ to be equal to $C(0)^{-1}$ to approximate the running costs. This approximation offers the advantage of reducing computational complexity by circumventing the need for backpropagation over the inverse cost function, thereby significantly increasing computation speed. Note that the approximation introduces an error of order Δt, which does not compromise the first order accuracy of the JKO scheme. The results of our experiment are presented in Figure 9, illustrating the density evolution from $t = 0$ to $t = 20$. The density converges to the stationary distribution, and the results are consistent with the numerical solution depicted in [21].

Figure 10 shows the loss plot and the contour plot of $\Phi(\boldsymbol{u})$. The loss plot displays the value of the loss function with the x-axis representing iterations and the y-axis representing the loss value. Here the loss function represents the relative entropy defined in (36). On the right, the figure illustrates the computed solution at $t = 20$, overlaid on the contour plot derived from $\Phi(\boldsymbol{u})$.

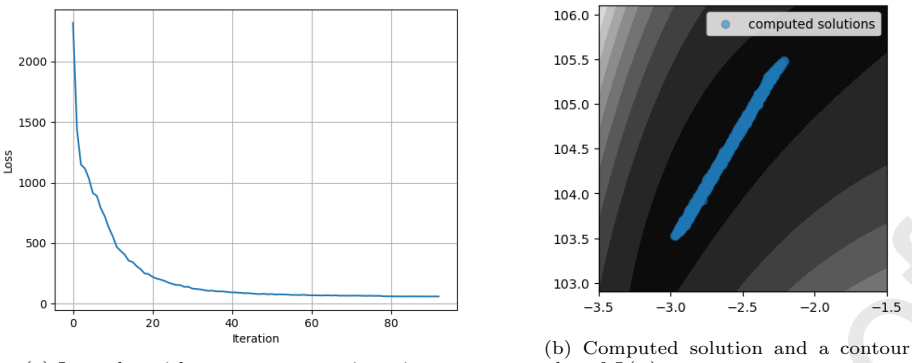


(a) $t = 0$ (b) $t = 2$ (c) $t = 4$ (d) $t = 7.5$ (e) $t = 20$

Figure 9: Computed solutions of a 2D Kalman-Wasserstein gradient flow (34).

5. Conclusion and discussion

In this paper, we introduce neural network-based implicit particle methods for computing high-dimensional Wasserstein-type gradient flows with linear and nonlinear mobility functions. Our approach centers around the Lagrangian formulation within the Jordan–Kinderlehrer–Otto (JKO) framework, where neural networks approximate the velocity field. Neural ODE techniques are harnessed to efficiently compute the time-implicit updates of both particle locations and the densities they carry. Additionally, we leverage an explicit recurrence relation to compute derivatives, significantly streamlining the backpropagation process. Our methodology exhibits versatility and can handle a broad spectrum of gradient flows while accommodating diverse potential functions and nonlinear mobility scenarios.



(a) Loss plot with respect to outer iterations

(b) Computed solution and a contour plot of $\Phi(\boldsymbol{u})$

Figure 10: The loss function values across iterations are depicted on the x-axis, while the computed solution at $t = 20$ is superimposed on the contour plot generated from $\Phi(\boldsymbol{u})$ (on the right).

There are several promising directions for future work. Firstly, there is potential for accelerating the training process. This could involve leveraging the properties of the velocity field as suggested in [32], or exploring transfer learning techniques [43], and improving resampling methods. Another avenue for exploration is the convexity of the problem, investigating how it depends on parameters like the step size Δt and network architectures, including the choices of neural network activation functions. Understanding the convexity properties can contribute to faster training and provide insights into the convergence properties of the algorithm. Additionally, we aim to extend this approach to accommodate more complex energy functionals and scenarios where a gradient flow structure is not fully presented or is only part of the physical dynamics. This extension would broaden the applicability of our method to a wider range of problems in simulating general physics oriented partial differential equations.

Acknowledgements

W. Lee acknowledges support from the National Institute of Standards and Technology (NIST) under award number 70NANB22H021 and partially supported by AFOSR YIP award No. FA9550-23-1-0087. L. Wang acknowledges support from NSF grant DMS-1846854. W. Li's work is supported by AFOSR MURI FP 9550-18-1-502, AFOSR YIP award No. FA9550-23-1-0087, NSF RTG: 2038080, and NSF DMS-2245097. L. Wang and W. Li would like to express their gratitude for the hospitality extended by AIM, where part of the work was discussed.

References

[1] J. Benamou and Y. Brenier, *A computational fluid mechanics solution*

to the Monge-Kantorovich mass transfer problem, Numer. Math., 84 (2000), pp. 375–393.

[2] J. Benamou, G. Carlier, and M. Laborde, *An augmented Lagrangian approach to Wasserstein gradient flows and applications*, ESAIM: PROCEEDINGS AND SURVEYS, 54 (2016), pp. 1–17.

[3] A. L. Bertozzi, *The mathematics of moving contact lines in thin liquid films*, Notices of the AMS, 45 (1998), pp. 689–697.

[4] N. M. Boffi and E. Vanden-Eijnden, *Probability flow solution of the fokker–planck equation*, Machine Learning: Science and Technology, 4 (2023), p. 035012.

[5] M. Burger, M. Di Francesco, and Y. Dolak-Struss, *The keller–segel model for chemotaxis with prevention of overcrowding: Linear vs. nonlinear diffusion*, SIAM Journal on Mathematical Analysis, 38 (2006), pp. 1288–1315.

[6] J. W. Cahn, *On spinodal decomposition*, Acta metallurgica, 9 (1961), pp. 795–801.

[7] C. Cances, T. O. Gallouët, and G. Todeschi, *A variational finite volume scheme for wasserstein gradient flows*, Numerische Mathematik, 146 (2020), pp. 437–480.

[8] J. Carrillo, S. Lisini, G. Savaré, and D. Slepčev, *Nonlinear mobility continuity equations and generalized displacement convexity*, Journal of Functional Analysis, 258 (2010), pp. 1273–1309.

[9] J. A. Carrillo, K. Craig, and F. S. Patacchini, *A blob method for diffusion*, Calculus of Variations and Partial Differential Equations, 58 (2019), pp. 1–53.

[10] J. A. Carrillo, K. Craig, L. Wang, and C. Wei, *Primal dual methods for Wasserstein gradient flows*, Found. Comput. Math., 22 (2022), pp. 389–443.

[11] J. A. Carrillo, D. Matthes, and M.-T. Wolfram, *Lagrangian schemes for Wasserstein gradient flows*, in Handbook of Numerical Analysis, vol. 22, Elsevier, 2021, pp. 271–311.

[12] J. A. Carrillo and J. S. Moll, *Numerical simulation of diffusive and aggregation phenomena in nonlinear continuity equations by evolving diffeomorphisms*, SIAM Journal on Scientific Computing, 31 (2010), pp. 4305–4329.

[13] J. A. Carrillo, H. Ranetbauer, and M.-T. Wolfram, *Numerical simulation of nonlinear continuity equations by evolving diffeomorphisms*, Journal of Computational Physics, 327 (2016), pp. 186–202.

[14] J. A. Carrillo, L. Wang, and C. Wei, *Structure preserving primal dual methods for gradient flows with nonlinear mobility transport distances*, (2023).

[15] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, *Neural ordinary differential equations*, Advances in neural information processing systems, 31 (2018).

[16] J. Dolbeault, B. Nazaret, and G. Savaré, *A new class of transport distances between measures*, Calculus of Variations and Partial Differential Equations, 34 (2009), pp. 193–231.

[17] S. Fagioli and O. Tse, *On gradient flow and entropy solutions for nonlocal transport equations with nonlinear mobility*, Nonlinear Analysis, 221 (2022), p. 112904.

[18] J. Fan, A. Taghvaei, and Y. Chen, *Variational wasserstein gradient flow*, arXiv preprint arXiv:2112.02424, (2021).

[19] G. Fu, S. Osher, and W. Li, *High order spatial discretization for variational time implicit schemes: Wasserstein gradient flows and reaction-diffusion systems*, arXiv:2303.08950 [math.NA], (2023).

[20] W. Gangbo and R. J. McCann, *The geometry of optimal transportation*, Acta Math., 177 (1996), pp. 113–161.

[21] A. Garbuno-Inigo, F. Hoffmann, W. Li, and A. M. Stuart, *Interacting Langevin Diffusions: Gradient Structure and Ensemble Kalman Sampler*, SIAM Journal on Applied Dynamical Systems, 19 (2020), pp. 412–441.

[22] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, *Ffjord: Free-form continuous dynamics for scalable reversible generative models*, arXiv preprint arXiv:1810.01367, (2018).

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

[24] M. Herty and G. Visconti, *Kinetic methods for inverse problems.*, Kinetic & Related Models, 12 (2019).

[25] Z. Hu, C. Liu, Y. Wang, and Z. Xu, *Energetic Variational Neural Network Discretizations to Gradient Flows*, 2022.

[26] R. Jordan, D. Kinderlehrer, and F. Otto, *The variational formulation of the fokker–planck equation*, SIAM Journal on Mathematical Analysis, 29 (1998), pp. 1–17.

[27] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, nature, 521 (2015), pp. 436–444.

[28] Q. Li, L. Wang, and Y. Yang, *Differential-equation constrained optimization with stochasticity*, arXiv preprint arXiv:2305.04024, (2023).

[29] W. Li, J. Lu, and L. Wang, *Fisher information regularization schemes for Wasserstein gradient flows*, J. Comput. Phys., 416 (2020), pp. 109449, 24.

[30] A. T. Lin, W. Li, S. Osher, and G. Montúfar, *Wasserstein proximal of gans*, in Geometric Science of Information, F. Nielsen and F. Barbaresco, eds., Cham, 2021, Springer International Publishing, pp. 524–533.

[31] S. Liu, W. Li, H. Zha, and H. Zhou, *Neural parametric fokker–planck equation*, SIAM Journal on Numerical Analysis, 60 (2022), pp. 1385–1449.

[32] X. Liu, C. Gong, and Q. Liu, *Flow straight and fast: Learning to generate and transfer data with rectified flow*, arXiv preprint arXiv:2209.03003, (2022).

[33] P. Mokrov, A. Korotin, L. Li, A. Genevay, J. M. Solomon, and E. Burnaev, *Large-scale wasserstein gradient flows*, Advances in Neural Information Processing Systems, 34 (2021), pp. 15243–15256.

[34] D. Onken, S. W. Fung, X. Li, and L. Ruthotto, *Ot-flow: Fast and accurate continuous normalizing flows via optimal transport*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 9223–9232.

[35] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational physics, 378 (2019), pp. 686–707.

[36] L. Ruthotto, S. J. Osher, W. Li, L. Nurbekyan, and S. W. 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 (2020), pp. 9183–9193.

[37] Z. Shen, Z. Wang, S. Kale, A. Ribeiro, A. Karbasi, and H. Hassani, *Self-consistency of the fokker planck equation*, in Conference on Learning Theory, PMLR, 2022, pp. 817–841.

[38] J. Sirignano and K. Spiliopoulos, *Dgm: A deep learning algorithm for solving partial differential equations*, Journal of computational physics, 375 (2018), pp. 1339–1364.

[39] A. Vidal, S. Wu Fung, L. Tenorio, S. Osher, and L. Nurbekyan, *Taming hyperparameter tuning in continuous normalizing flows using the JKO scheme*, Scientific Reports, 13 (2023), p. 4501.

[40] C. Villani et al., *Optimal transport: old and new*, vol. 338, Springer, 2009.

[41] L. Wang and M. Yan, *Hessian informed mirror descent*, Journal of Scientific Computing, 92 (2022), p. 90.

[42] C. Xu, X. Cheng, and Y. Xie, *Normalizing flow neural networks by JKO scheme*, in Thirty-seventh Conference on Neural Information Processing Systems, 2023.

[43] W. Xu, Y. Lu, and L. Wang, *Transfer learning enhanced deeponet for long-time prediction of evolution equations*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 10629–10636.

[44] B. Yu and W. E, *The deep ritz method: a deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics, 6 (2018), pp. 1–12.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: