Wasserstein gradient flows: modeling and applications

Petr Mokrov

Submitted to the Moscow Institute of Physics and Technology
on June 9, 2022

Abstract

Wasserstein gradient flows provide a powerful means of understanding and solving many diffusion equations. Specifically, Fokker-Planck equations, which model the diffusion of probability measures, can be understood as gradient descent over entropy functionals in Wasserstein space. This equivalence, introduced by Jordan, Kinderlehrer and Otto, inspired the so-called JKO scheme to approximate these diffusion processes via an implicit discretization of the gradient flow in Wasserstein space. Solving the optimization problem associated to each JKO step, however, presents serious computational challenges. We introduce a scalable method to approximate Wasserstein gradient flows, targeted to machine learning applications. Our approach relies on input-convex neural networks (ICNNs) to discretize the JKO steps, which can be optimized by stochastic gradient descent. Unlike previous work, our method does not require domain discretization or particle simulation. As a result, we can sample from the measure at each time step of the diffusion and compute its probability density. We demonstrate our algorithm's performance by computing diffusions following the Fokker-Planck equation and apply it to unnormalized density sampling as well as nonlinear filtering. The work is based on our article [57].

Keywords: optimal transport, Fokker-Planck equation, Wasserstein gradient flows.

Research Advisor:
Name: Evgeny Burnaev
Degree, title: Dr.Sc., Associate Professor

Градиентные потоки Вассерштайна: методы моделирования и применение в приложениях

Петр Мокров

Предоставлено в Московский физико-технический институт
Июнь 9, 2022

Реферат

Градиентные потоки Вассерштайна являются мощным средством, помогающим понять и решить некоторые диффузионные уравнения. В частности, уравнение Фоккера-Планка, которое описывает достаточно широкий класс диффузионных процессов, можно представить как градиентный спуск, минимизирующий энтропийный функционал в пространстве вероятностных мер с метрикой Вассерштайна. В пионерской работе Jordan, Kinderlehrer и Otto была предложена так называемая JKO схема для аппроксимации диффузионных процессов с помощью неявной временной дискретизации градиентного потока в пространстве Вассерштайна. Вместе с тем, задачи оптимизации, возникающие на каждом шаге JKO схемы, вычислительно трудозатратны. В настоящей работе предложен масштабируемый метод аппроксимации градиентных потоков Вассерштайна, который можно применять в приложениях машинного обучения. Для параметризации JKO шагов используются выпуклые нейронные сети (ICNNs), оптимизируемые с помощью стохастического градиентного спуска. В отличие от предшествующих работ, предложенный подход не требует дискретизации пространства и симуляции частиц. Описан алгоритм создания выборки из распределения градиентного потока в каждый момент времени и метод вычисления функции плотности этого распределения. В качестве вычислительного экспериментов было произведено моделирование некоторых диффузионных процессов, подчиняющихся уравнению Фоккера-Планка а также были решены практические задачи, связанные с семплированием из ненормируемой плотности и нелинейной фильтрацией. Работа основана на статье [57] автора настоящей диссертации с соавторами.

Ключевые слова: оптимальный транспорт, уравнение Фоккера-Планка, градиентные потоки Вассерштайна

Научный руководитель:
Имя: Евгений Владимирович Бурнаев
Ученое звание, должность: д.ф.-м.н., доцент

Contents

1 Introduction

Gradient flows, also known as *steppest descent curves* are very common field in evolution equations. Given a functional $F : X \to \mathbb{R}$ defined on a vector space X and initial point $x_0 \in X$ we seek for a curve $x(t)$, starting at x_0 and minimizing F as fast as possible (Thus we are to solve the equation $x'(t) = -\boldsymbol{\nabla} F(x(t))$ for $t \in \mathbb{R}_+$). Thorough gradient flows overview can be found in [71]. In the Euclidean case $X = \mathbb{R}^n$ gradient flows are equivalent to ODE. Discretization of this flow leads to the gradient descent minimization algorithm. The steppest descent curves could be considered in more general metric spaces [5]. Of the particular interest of our work are gradient flows on the probability measures space equipped with 2-Wasserstein metric also known as Wasserstein gradient flows. We introduce basic notations and statements as well as give brief theoretical review in the next subsections.

1.1 Basic Notations and Statements

$\mathcal{P}_2(\mathbb{R}^D)$ denotes the set of Borel probability measures on $\mathbb{R}^D$ with finite second moment. $\mathcal{P}_{2,ac}(\mathbb{R}^D)$ denotes its subset of probability measures absolutely continuous with respect to Lebesgue measure. The $\mathcal{B}(\mathbb{R}^D)$ are the Borel sets. For $\rho \in \mathcal{P}_{2,ac}(\mathbb{R}^D)$, we denote by $\frac{d\rho}{dx}(x)$ its density with respect to the Lebesgue measure.

Definition 1. *Let $\mu, \nu \in \mathcal{P}(\mathbb{R}^D)$. The* ***coupling*** *of μ and ν denoted as $\Pi(\mu, \nu)$ is the set of probability measures on $\mathbb{R}^D \times \mathbb{R}^D$ whose first and second marginals are μ and ν respectively, i.e:*

$$\Pi(\mu, \nu) = \left\{\pi \in \mathcal{P}(\mathbb{R}^D \times \mathbb{R}^D) \middle| \forall A \in \mathcal{B}(\mathbb{R}^D) : \pi(A \times \mathbb{R}^D) = \mu(A); \pi(\mathbb{R}^D \times A) = \nu(A)\right\}$$

Definition 2. *Let $T : \mathbb{R}^D \to \mathbb{R}^D$ be a measurable function, $\mu \in \mathbb{R}^D$. Then there exists the measure ν such, that $\forall A \in \mathcal{B}(\mathbb{R}^D) : \nu(A) = \mu(T^{-1}(A))$. We write $\nu = T\sharp\mu$ and designate $T\sharp$ to be the associated* ***push-forward*** *operator between measures.*

Our work deals with flows or curves $\{\mu_t\}_{t\in[0,T]}$ where $\mu_t \in \mathcal{P}_2(\mathbb{R}^D)$. Generally speaking, an arbitrary continuous sequence can be understood as a "flow" in probability measures space. However, in order a curve to be analyzable and to make sense from the geometrical point of view it should satisfy some continuity or smoothness properties. Therefore, we need to define a notion of closeness between probability measures. And the right notion of closeness which satisfy nice theoretical properties is the Wasserstein-2 metric defined below.

Definition 3. *The (squared) Wasserstein-2 metric $\mathcal{W}_2$ between $\mu, \nu \in \mathcal{P}_2(\mathbb{R}^D)$ is [78]:*

$$\mathcal{W}_2^2(\mu, \nu) \overset{def}{=} \min_{\pi \in \Pi(\mu,\nu)} \int_{\mathbb{R}^D \times \mathbb{R}^D} \|x - y\|_2^2 \, d\pi(x, y), \tag{1}$$

$\mathcal{W}_2$ turns out to be a metric in $\mathcal{P}_2(\mathbb{R}^D)$ and the $(\mathcal{P}_2, \mathcal{W}_2)$ constitutes complete separable metric space. It metrizes weak convergence coupled with second moment convergence [13].

Optimal transportation problem and Brenier's theorem. The equation (1) is known as optimal transport problem with quadratic cost function $c(x, y) = \|x-y\|_2^2$. A measure $\pi^* \in \Pi(\mu, \nu)$ which minimizes (1) (it can be not unique) is called optimal transport plan. The optimal transport problem was introduced by Kantorovich in 20-th century as the relaxed version of the optimal transport problem due to Monge, firstly considered in the 18-th century [58]:

$$\inf_{T:\nu=T\sharp\mu} \int\limits_{\mathbb{R}^d} \|x - T(x)\|_2^2 \mathrm{d}\mu(x) \tag{2}$$

The optimal map $T* : \mathbb{R}^D \to \mathbb{R}^D$ which minimizes (2) permits straightforward explanation. It shows the optimal way to "drag" probability mass of μ to ν given that the transportation cost is squared euclidean distance $\|x - y\|_2^2$. For general measures μ and ν the solution for the Monge's problem may not exist (in contrast to the Kantorovich's problem). However, for $\mu \in \mathcal{P}_{2,ac}(\mathbb{R}^D)$, there exists a μ-unique map $\nabla\psi^* : \mathbb{R}^D \to \mathbb{R}^D$ that is the gradient of a convex function $\psi^* : \mathbb{R}^D \to \mathbb{R} \sqcup \{\infty\}$ satisfying $\nabla\psi^*\sharp\mu = \nu$ [54]. Moreover, from the Brenier's theorem [16], [24, Theorem 2.12] it follows, that the $\nabla\psi^*$ forms the μ-unique optimal map T^* of the Monge problem (2) and $\pi^* = [\mathrm{id}_{\mathbb{R}^D}, \nabla\psi^*]\sharp\mu$ is the unique minimizer of (1), i.e.,

$$\mathcal{W}_2^2(\mu, \nu) = \int_{\mathbb{R}^D} \|x - \nabla\psi^*(x)\|_2^2 \, d\mu(x). \tag{3}$$

The equation (3) and Brenier's theorem play crucial role in our research, as they allow to recast the search for $\mathcal{W}_2$ distance as minimization with respect to the set of convex functions.

1.2 Wasserstein Gradient Flows

The idea of gradient flow in Wasserstein space $(\mathcal{P}_2(\mathbb{R}^D), \mathcal{W}_2)$ is similar to Euclidean case: the flow $\{\rho_t\}_{t\in[0,T]}$ should be good enough from geometrical and analytical perspectives and follows the "steepest descent" direction of a functional $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^2) \to \mathbb{R}$, but this time the notion of gradient is more complex.

In what follows $L^1[a, b]$ ($L^2[a, b]$) denotes functions $\mathbb{R}^D \to \mathbb{R}$ whose modulus (square) is Lebesgue integrable on the segment $[a, b]$. $L^2(\rho)$ denotes functions $\mathbb{R}^D \to \mathbb{R}$ which are square integrable with respect to measure ρ on $\mathbb{R}^D$. $\langle f, g\rangle_{L^2(\rho)} = \int\langle f, g\rangle \mathrm{d}\rho$ is the scalar product in the space $L^2(\rho)$; $\|\cdot\|_{L^2(\rho)}$ is the corresponding norm. $C_0^1(\mathbb{R}^D)$ designates the set of continuously differentiable functions supported on a compact. "A.e." stands for "almost everywhere".

Absolutely continuous flows and their geometry.

Definition 4. *A curve* $\gamma : [0, T] \to \mathcal{P}_2(\mathbb{R}^D)$ *is called* ***absolutely continuous*** *if* $\exists g \in L^1[0, T]$, $g \geq 0$*:*

$$\mathcal{W}_2(\gamma(t), \gamma(s)) \leq \int\limits_s^t g(\tau)\mathrm{d}\tau\,, \quad \forall s \leq t;\ s, t \in [0, T].$$

The absolute continuity guarantees the $\forall t \in [0, T]$ – a.e. existence of the metric derivative $|\gamma'|(t) \overset{\text{def}}{=} \lim\limits_{s\to t} \frac{\mathcal{W}_2(\gamma(t),\gamma(s))}{|t-s|}$ which is the basic property of a curve because it enables such notions as "length",

”speed” and so on.

It turns out, that the absolutely continuous flows permit a remarkable analytical characterization known as Ambrosio theorem [4, Theorem 2.29].

Theorem (Ambrosio) *Let ρ_t, $t \in [0,T]$ be an absolutely continuous flow and $|\rho'|(t) \in L^2[0,T]$. Then there exists a Borel vector field $b : [0,T] \times \mathbb{R}^D \to \mathbb{R}^D$ such, that:*

$$\begin{aligned} |\rho'|(t) &= \|b(\,\cdot\,,t)\|_{L^2(\rho_t)} \\ \partial_t \rho_t + \mathrm{div}(b\rho_t) &= 0 \end{aligned} \tag{4}$$

The (4) is known as continuity equation. For general ρ_t in can be understood in an integral sense:

$$\forall \phi \in C_0^1(\mathbb{R}^D) \; : \; \frac{d}{dt}\left(\int \phi \mathrm{d}\rho_t\right) = \int \langle b(\cdot,t), \nabla_x \phi(\cdot)\rangle \mathrm{d}\rho_t.$$

If ρ_t is absolutely continuous the (4) could be understood straightforwardly by substituting ρ_t with the corresponding probability density function $p_{\rho_t}(x,t) = \frac{d\rho_t}{dx}(x)$. In this case the div operator has conventional meaning from vector calculus: $\mathrm{div}(f(x,t)) = \sum_{i=1}^{D} \frac{\partial f}{\partial x_i}$.

To clarify readers with the role of vector field b in the equation (4) we give the following example:

Example 1. *Let $b : [0,T] \times \mathbb{R}^D \to \mathbb{R}^D$ - is a smooth, bounded, with bounded gradient. Let $X_t(y)$ be the solution of the Cauchy problem:*

$$\begin{cases} \mathrm{d}x/\mathrm{d}t = b(x,t) \\ x(t=0) = y \end{cases}$$

Let $\rho \in \mathcal{P}_2(\mathbb{R}^D)$. Define the flow $\rho_t = X_t \sharp \rho$. Then the ρ_t, $t \in [0,T]$ is absolutely continuous and satisfy the continuity equation (4) *for the vector field b.*

From the example one can see, that the measures ρ_t are transferred along the vector field b. This intuition could be rigorously generalized. Let (ρ_t, b) satisfy the Ambrosio theorem. Then the succeeding ”displacement tangency” property [4, Proposition 2.33] holds true:

$$\lim_{h\to 0} \frac{\mathcal{W}_2\left(\rho_{t+h}, [x + hb(x,t)]\,\sharp\rho_t)\right)}{h} = 0$$

Following the equation above b is called tangent vector field for the curve ρ_t, $t \in [0,T]$.

Flat derivative in $\mathcal{P}_2(\mathbb{R}^2)$ and it’s gradient

Definition 5. *Let $U : \mathcal{P}_2(\mathbb{R}^D) \to \mathbb{R}$. We say, that U is of class C^1, if: $\exists \frac{\delta U}{\delta m} : \mathcal{P}_2(\mathbb{R}^D) \times \mathbb{R}^D \to \mathbb{R}$ - bounded and continuous such, that:*

$$U(\nu) - U(\mu) = \int_0^1 \int \frac{\delta U}{\delta m}(\mu + t(\nu - \mu), x) \cdot [\nu - \mu](\mathrm{d}x)\mathrm{d}t$$

The $\frac{\delta U}{\delta m}$ if called **flat derivative** [21, Chapter 1.4.2] or first variation [70, Chapter 8] of the functional U. The flat derivative shows the behavior of the functional U under small perturbations of measure. In particular, if U is of class C^1 and satisfies some additional assumptions,

$$U(\nu) - U(\mu) = \int \frac{\delta U}{\delta m}(x)[\nu - \mu](\mathrm{d}x) + o\left(\mathcal{W}_2(\nu, \mu)\right)$$

See [22, Proposition 5.44] for rigorous formulation.

Consider the x - gradient of the flat derivative. It turns out, that $\nabla_x \frac{\delta U}{\delta m}(\mu, x)$ could be understood as the gradient of the functional U in $(\mathcal{P}_2, \mathcal{W}_2)$. Let U is of class C^1 and $\nabla_x \frac{\delta U}{\delta m}(\mu, x)$ is bounded and continuous. Let ρ_t, $t \in [0, T]$ is an absolutely continuous curve with a tangent vector field b. Then, $\forall t \in [0, T]$ - a.e. [5, Equation 10.1.16]:

$$\frac{d}{dt} U(\rho_t) = \left\langle \nabla_x \frac{\delta U}{\delta m}(\rho_t, \,\cdot\,), b(\,\cdot\,, t) \right\rangle_{L^2(\rho_t)} \tag{5}$$

The equation (5) resembles the standard gradient definition of a functional defined on a Riemann manifold (note, that $b(\,\cdot\,, t)$ is the tangent vector of the flow ρ_t at time point t).

By now we have given a brief and intuitive introduction to the absolutely continuous flows in the $(\mathcal{P}_2, \mathcal{W}_2)$ and derivatives in probability measure space. More rigorous style of presentation could be found in [4, 5, 24, 70, 71] (a.c. flows) and [22, 21, 70] (gradients). Now we are ready to define the Wasserstein gradient flows.

Wasserstein gradient flows and Fokker-Planck equation. A natural idea inspired by (5) is to consider a curve in probability measures space which follows the ”gradient” of a functional $\mathcal{F}$ similar to gradient flows in Euclidean case §1.

Definition 6. *An absolutely continuous curve $\{\rho_t\}_{t \in [0,T]}$ follows the* ***Wasserstein gradient flow*** *of a functional $\mathcal{F}$ if it solves the continuity equation:*

$$\partial_t \rho_t = \operatorname{div}(\rho_t \nabla_x \frac{\delta \mathcal{F}}{\delta m}(\rho_t, x)), \qquad s.t.\ \rho_0 = \rho^0, \tag{6}$$

Wasserstein gradient flows are used in various applied tasks. For example, gradient flows are applied in training [9, 51, 32] or refinement [8] of implicit generative models. In reinforcement learning, gradient flows facilitate policy optimization [66, 83]. Other tasks include crowd motion modelling [53, 69, 61], dataset optimization [2], and in-between animation [33].

Many applications come from the connection between Wasserstein gradient flows and Stochastic differential equations (SDEs). Consider an $\mathbb{R}^D$-valued stochastic process $\{X_t\}_{t \in \mathbb{R}_+}$ governed by the following Itô SDE:

$$dX_t = -\nabla \Phi(X_t) dt + \sqrt{2\beta^{-1}} dW_t, \qquad \text{s.t. } X_0 \sim \rho^0 \tag{7}$$

where $\Phi : \mathbb{R}^D \to \mathbb{R}$ is the potential function, W_t is the standard Wiener process, and $\beta > 0$ is the magnitude. The solution of (7) is called an *advection-diffusion* process. It arises in various applications including physics [73], finance [28, 63], population dynamics [42, 17] and molecular discovery [3]. In machine learning the SDE in the form (7) appears in applications filtering [41, 27] and unnormalized posterior sampling via a discretization of the Langevin diffusion [80].

The marginal measure ρ_t of X_t which follows (7) at each time satisfies the *Fokker-Planck equation* with fixed diffusion coefficient:

$$\frac{\partial \rho_t}{\partial t} = \mathrm{div}(\nabla \Phi(x)\rho_t) + \beta^{-1}\Delta\rho_t, \qquad \text{s.t. } \rho_0 = \rho^0. \tag{8}$$

Equation (8) turns out to be the Wasserstein gradient flow (6) for $\mathcal{F}$ given by the Fokker-Planck free energy functional [39]

$$\mathcal{F}_{\mathrm{FP}}(\rho) = \mathcal{U}(\rho) - \beta^{-1}\mathcal{E}(\rho), \tag{9}$$

where $\mathcal{U}(\rho) = \int_{\mathbb{R}^D} \Phi(x)d\rho(x)$ is the *potential energy* and $\mathcal{E}(\rho) = -\int_{\mathbb{R}^D} \log\frac{d\rho}{dx}(x)d\rho(x)$ is the *entropy*. As the result, to solve the SDE (7), one may compute the Wasserstein gradient flow of the Fokker-Planck equation with the free-energy functional $\mathcal{F}_{\mathrm{FP}}$ given by (9).

1.3 JKO Scheme

Modelling of the processes which satisfy (8) (and correspondingly (7)) is the primal goal of our work. However, the closed-form solutions are generally intractable, necessitating numerical approximation techniques. Jordan, Kinderlehrer, and Otto proposed a method — later abbreviated as *JKO integration* — to approximate the dynamics of ρ_t in (8) [39]. It consists of a time-discretization update of the continuous flow given by:

$$\rho^{(k)} \leftarrow \operatorname*{arg\,min}_{\rho\in\mathcal{P}_2(\mathbb{R}^D)} \left[\mathcal{F}(\rho) + \frac{1}{2h}\mathcal{W}_2^2(\rho^{(k-1)}, \rho)\right] \tag{10}$$

where $\rho^{(0)} = \rho^0$ is the initial condition and $h > 0$ is the time-discretization step size. The discrete time gradient flow converges to the continuous one as $h \to 0$, i.e., $\rho^{(k)} \approx \rho_{kh}$. The method was further developed in [5, 71], but performing JKO iterations in practice remains challenging thanks to the minimization with respect to probability measures and presence of the term $\mathcal{W}_2$ in the optimization objective.

1.4 Input Convex Neural Networks

The basic ingredient which helps us to mitigate problems with the JKO scheme is Input Convex Neural Networks (ICNNs). ICNNs are parametric models based on deep neural networks which approximates the set of convex functions and could be optimized via standard deep learning optimization techniques [7].

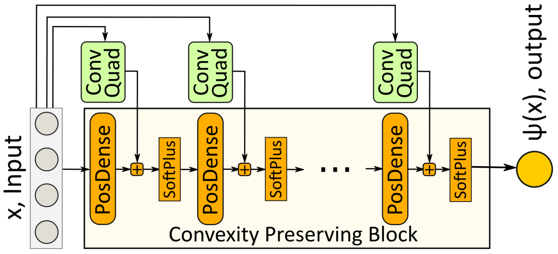


Figure 1: Dense Input Convex Neural Network structure. Credits: [44]

Our implementation of ICNNs

based on Dense ICNN architecture depicted on Figure 1, proposed by [44]. It consists of two types of blocks:

- **ConvQuad** blocks, which represent convex quadratic function of the input, i.e. ConvQuad(x) $=$ $\langle Ax, x\rangle + \langle b, x\rangle + c$, where A is a $\dim(x) \times \dim(x)$ PSD matrix of rank r (this parameter regulate the complexity of the model).
- **PosDense** blocks are linear layers with non-negative weights.

The convexity of the proposed architecture follows from convex function arithmetic [15]. Note, that one can substitute the *SoftPlus* activations in the Figure 1 with arbitrary convex monotone functions. Actually, in our work we use *SoftPlus* activations as it shows better experimental results compared to alternatives.

ICNNs are known to represent rather rich family of convex functions [26, Theorem 1]. Embedding them into the JKO objective (10) with help of Brenier's theorem gives us a way to approximate the Fokker-Planck gradient flow (8). The details are given in further sections. Our main contributions are summarized as follows.

Contributions. We propose a scalable parametric method to approximate Wasserstein gradient flows with Fokker-Planck functional $\mathcal{F}_{FP}$ (9) via JKO stepping using input-convex neural networks (ICNNs) [7]. Specifically, we leverage Brenier's theorem to bypass the costly computation of the Wasserstein distance, and parametrize the optimal transport map as the gradient of an ICNN. Given sample access to the initial measure ρ_0, we use stochastic gradient descent (SGD) to sequentially learn time-discretized JKO dynamics of ρ_t. The trained model can sample from a continuous approximation of ρ_t and compute its density $\frac{d\rho_t}{dx}(x)$. We demonstrate performance by computing diffusion following the Fokker-Planck equation and applying it to unnormalized density sampling as well as nonlinear filtering.

2 Literature Review

One way to compute the diffusion which satisfies (8) is to use a fixed discretization of the domain and apply standard numerical integration methods [25, 60, 19, 23, 48] to get ρ_t. For example, [61] proposes a method to approximate the diffusion based on JKO stepping and entropy-regularized optimal transport. However, these methods are limited to small dimensions since the discretization of space grows exponentially.

An alternative to domain discretization is stochastic particle simulation which exploits the SDE form (7). It involves drawing random samples (particles) from the initial distribution and simulating their evolution via standard methods such as Euler-Maruyama scheme [43, §9.2]. After convergence, the particles are approximately distributed according to the stationary distribution, but no density estimate is readily available.

Another way to avoid discretization is to parameterize the density of ρ_t. Most methods approximate only the first and second moments ρ_t, e.g., via Gaussian approximation. Kalman filtering approaches can then compute the dynamics [41, 47, 40, 72]. More advanced Gaussian mixture approximations [75, 1] or more general parametric families have also been studied [74, 79]. In [59], variational methods are used to minimize the divergence between the predictive and the true density.

The closest to our approach is a line of works related to JKO scheme. A common method to perform JKO steps is to discretize the spatial domain. For support size $\lessapprox 10^6$, (10) can be solved by standard optimal transport algorithms [62]. Another original idea combining spatial discretization with set of convex functions discretization proposed in [11]. However, in dimensions $D \geq 3$, discrete supports can hardly approximate continuous distributions and hence the dynamics of gradient flows. To tackle this issue, [31] propose a stochastic parametric method to approximate the density of ρ_t. The authors regularize the Wasserstein distance in the JKO step to ensure strict convexity and solve the unconstrained dual problem via stochastic program on a finite linear subset of basis functions. The method is biased and yields *unnormalized* probability density without direct sample access.

Recently, several competitive works [3, 17, 30, 14] have appeared which utilize ideas similar to ours. While [3] (independently) come up with the optimization procedure analogous to Algorithm 1, [17] consider an alternative task but also used ICNN powered JKO. Actually, the authors of [17] try to recover the function Φ from the equation (7) given experimental dynamics. [30] is somewhat generalizes our approach by inserting variational formulation of f-divergence into the JKO objective which allows them to reduce time consumption and consider broader class of Wasserstein gradient flow functionals. [14] substitute Wassetstein distance in the equation (10) with sliced Wassetstein distance and try to utilize the obtained surrogate in practice.

The rise of ICNNs. Since its origin at [7] , Input Convex Neural Networks have been gathering close attention in machine learning community especially in the spheres related to Optimal Transport. [37] use ICNNs to build flow-based models resembling normalizing flows, [44, 52] build generative models based on optimal transport map via ICNNs, [29, 46] solve high-dimensional Wasserstein barycenter problem, [18, 81] utilize ICNNs in context of Model Predictive Control (MPC) problem. Despite of apparent success some researches [45] report several performance issues with Input Convex Neural Networks. That's why the problem of convex functions modeling still is of independent interest.

3 Computational Methodology

We now describe our approach to compute Wasserstein gradient flows via JKO stepping with ICNNs.

3.1 JKO Reformulation via Optimal Push-forwards Maps

Our key idea is to replace the optimization (10) over probability measures by an optimization over convex functions, an idea inspired by [11]. Thanks to Brenier's theorem, for any $\rho \in \mathcal{P}_{2,ac}$ there exists a unique $\rho^{(k-1)}$-measurable gradient $\nabla\psi : \mathbb{R}^D \to \mathbb{R}^D$ of a convex function ψ satisfying

$\rho = \nabla\psi\sharp\rho^{(k-1)}$. We set $\rho = \nabla\psi\sharp\rho^{(k-1)}$ and rewrite (10) as an optimization over convex ψ:

$$\psi^{(k)} \leftarrow \operatorname*{arg\,min}_{\text{Convex }\psi}\left[\mathcal{F}(\nabla\psi\sharp\rho^{(k-1)}) + \frac{1}{2h}\mathcal{W}_2^2(\rho^{(k-1)}, \nabla\psi\sharp\rho^{(k-1)})\right]. \tag{11}$$

To proceed to the next step of JKO scheme, we define $\rho^{(k)} \stackrel{\text{def}}{=} \nabla\psi^{(k)}\sharp\rho^{(k-1)}$.

Since ρ is the pushforward of $\rho^{(k-1)}$ by the gradient of a convex function $\nabla\psi$, the $\mathcal{W}_2^2$ term in (11) can be evaluated explicitly, simplifying the Wasserstein-2 distance term in (11):

$$\psi^{(k)} \leftarrow \operatorname*{arg\,min}_{\text{Convex }\psi}\left[\mathcal{F}(\nabla\psi\sharp\rho^{(k-1)}) + \frac{1}{2h}\int_{\mathbb{R}^D}\|x - \nabla\psi(x)\|_2^2 d\rho^{(k-1)}(x)\right]. \tag{12}$$

This formulation avoids the difficulty of computing Wasserstein-2 distances. An additional advantage is that we can *sample* from $\rho^{(k)}$. Since $\rho^{(k)} = [\nabla\psi^{(k)} \circ \cdots \circ \nabla\psi^{(1)}]\sharp\rho^0$, one may sample $x_0 \sim \rho^{(0)}$, and then $\nabla\psi^{(k)} \circ \cdots \circ \nabla\psi^{(1)}(x_0)$ gives a sample from $\rho^{(k)}$. Moreover, if functions $\psi^{(\cdot)}$ are strictly convex, then gradients $\nabla\psi^{(\cdot)}$ are invertible. In this case, the *density* $\frac{d\rho^{(k)}}{dx}$ of $\rho^{(k)} = \nabla\psi^{(k)} \circ \cdots \circ \nabla\psi^{(1)}\sharp\rho^0$ is computable by the change of variables formula (assuming $\psi^{(\cdot)}$ are twice differentiable)

$$\frac{d\rho^{(k)}}{dx}(x_k) = [\det \nabla^2\psi^{(k)}(x_{k-1})]^{-1} \cdots [\det \nabla^2\psi^{(1)}(x_0)]^{-1} \cdot \frac{d\rho^{(0)}}{dx}(x_0), \tag{13}$$

where $x_i = \nabla\psi^{(i)}(x_{i-1})$ for $i = 1, \dots, k$ and $\frac{d\rho^{(0)}}{dx}$ is the density of $\rho^{(0)}$.

3.2 Stochastic Optimization for JKO via ICNNs

In general, the solution $\psi^{(k)}$ of (12) is intractable since it requires optimization over all convex functions. To tackle this issue, [11] discretizes the space of convex function. The approach also requires discretization of measures $\rho^{(k)}$ limiting this method to small dimensions.

We propose to parametrize the search space using input convex neural networks (ICNNs) [7] satisfying a universal approximation property among convex functions [26]. ICNNs are parametric models of the form $\psi_\theta : \mathbb{R}^D \to \mathbb{R}$ with ψ_θ convex w.r.t. the input. ICNNs are constructed from neural network layers, with restrictions on the weights and activation functions to preserve the input-convexity, see [7, §3.1] or [44, §B.2]. The parameters are optimized via deep learning optimization techniques such as SGD.

The JKO step then becomes finding the optimal parameters θ^* for ψ_θ:

$$\theta^* \leftarrow \operatorname*{arg\,min}_{\theta}\left[\mathcal{F}(\nabla\psi_\theta\sharp\rho^{(k-1)}) + \frac{1}{2h}\int_{\mathbb{R}^D}\|x - \nabla\psi_\theta(x)\|_2^2 d\rho^{(k-1)}(x)\right]. \tag{14}$$

If the functional $\mathcal{F}$ can be estimated stochastically using random batches from $\rho^{(k-1)}$, then SGD can be used to optimize θ. $\mathcal{F}_{\text{FP}}$ given by (9) is an example of such a functional:

Theorem 1 (Estimator of $\mathcal{F}_{\text{FP}}$). *Let $\rho \in \mathcal{P}_{2,ac}(\mathbb{R}^D)$ and $T : \mathbb{R}^D \to \mathbb{R}^D$ be a diffeomorphism. For* *a random batch* $x_1, \dots, x_N \sim \rho$*, the expression* $[\widehat{\mathcal{U}_T}(x_1, \dots, x_N) - \beta^{-1}\widehat{\Delta\mathcal{E}_T}(x_1, \dots, x_N)]$*, where*

$$\widehat{\mathcal{U}_T}(x_1, \dots, x_N) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{n=1}^{N}\Phi\big(T(x_n)\big) \textit{ and}$$

$$\widehat{\Delta\mathcal{E}_T}(x_1, \dots, x_N) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{n=1}^{N}\log|\det\nabla T(x_n)|,$$

is an estimator of $\mathcal{F}_{FP}(T\sharp\rho)$ *up to constant (w.r.t.* T*) shift given by* $\beta^{-1}\mathcal{E}(\rho)$*.*

Proof. $\widehat{\mathcal{U}_T}$ is a straightforward unbiased estimator for $\mathcal{U}(T\sharp\rho)$. Let p and p_T be the densities of ρ and $T\sharp\rho$. Since T is a diffeomorphism, we have $p_T(y) = p(x)\cdot|\det\nabla T(x)|^{-1}$ where $x = T^{-1}(y)$. Using the change of variables formula, we write

$$\begin{aligned}
\mathcal{E}(T\sharp\rho) &= -\int_{\mathbb{R}^D} p_T(y)\log p_T(y)dy \\
&= -\int_{\mathbb{R}^D} p(x)\cdot|\det\nabla T(x)|^{-1}\log\left[p(x)\cdot|\det\nabla T(x)|^{-1}\right]\cdot|\det\nabla T(x)|dx \\
&= -\int_{\mathbb{R}^D} p(x)\log p(x)dx + \int_{\mathbb{R}^D} p(x)\log|\det\nabla T(x)|dx \\
&= \mathcal{E}(\rho) + \int_{\mathbb{R}^D} p(x)\log|\det\nabla T(x)|dx, \\
\implies \Delta\mathcal{E}_T(\rho) &\stackrel{\text{def}}{=} \mathcal{E}(T\sharp\rho) - \mathcal{E}(\rho) = \int_{\mathbb{R}^D}\log|\det\nabla T(x)|d\rho(x)
\end{aligned}$$

which explains that $\widehat{\Delta\mathcal{E}_T}$ is an unbiased estimator of $\Delta\mathcal{E}_T(\rho)$. As the result, $\widehat{\mathcal{U}_T} - \beta^{-1}\widehat{\Delta\mathcal{E}_T}$ is an estimator for $\mathcal{F}_{\text{FP}}(T\sharp\rho) = \mathcal{U}(T\sharp\rho) - \beta^{-1}\mathcal{E}(T\sharp\rho)$ up to a shift of $\beta^{-1}\mathcal{E}(\rho)$. □

To apply Theorem 1 to our case, we take $T \leftarrow \nabla\psi_\theta$ and $\rho \leftarrow \rho^{(k-1)}$ to obtain a stochastic estimator for $\mathcal{F}_{\text{FP}}(\nabla\psi_\theta\sharp\rho^{(k-1)})$ in (14). Here, $\beta^{-1}\mathcal{E}(\rho^{(k-1)})$ is θ-independent and constant since $\rho^{(k-1)}$ is fixed, so the offset of the estimator plays no role in the optimization w.r.t. θ.

Algorithm 1 details our stochastic JKO method for $\mathcal{F}_{\text{FP}}$. The training is done solely based on random samples from the initial measure ρ^0: its density is not needed.

This algorithm assumes $\mathcal{F}$ is the Fokker-Planck diffusion energy functional. However, our method admits straightforward generalization to any $\mathcal{F}$ that can be stochastically estimated; studying such functionals is a promising avenue for future work.

3.3 Computing the Density of the Diffusion Process

Our algorithm provides a *computable density* for $\rho^{(k)}$. As discussed in §3.1, it is possible to sample from $\rho^{(k)}$ while simultaneously computing the density of the samples. However, this approach does not provide a direct way to evaluate $\frac{d\rho^{(k)}}{dx}(x_k)$ for arbitrary $x_k \in \mathbb{R}^D$. We resolve this issue below.

If a convex function is strongly convex, then its gradient is bijective on $\mathbb{R}^D$. By the change of variables formula for $x_k \in \mathbb{R}^D$, it holds $\frac{d\rho^{(k)}}{dx}(x_k) = \frac{d\rho^{(k-1)}}{dx}(x_{k-1})\cdot[\det\nabla^2\psi^{(k)}(x_{k-1})]^{-1}$ where

Algorithm 1: Fokker-Planck JKO via ICNNs

```
Input  : Initial measure $\rho^0$ accessible by samples;
         JKO discretization step $h > 0$, number of JKO steps $K > 0$;
         target potential $\Phi(x)$, diffusion process temperature $\beta^{-1}$;
         batch size $N$;
Output : trained ICNN models $\{\psi^{(k)}\}_{k=1}^{K}$ representing JKO steps
for $k = 1, 2, \ldots, K$ do
    $\psi_\theta \leftarrow$ basic ICNN model;
    for $i = 1, 2, \ldots$ do
        Sample batch $Z \sim \rho^0$ of size N ;          /* WGF sampling procedure */
        $X \leftarrow \boldsymbol{\nabla}\psi^{(k-1)} \circ \cdots \circ \boldsymbol{\nabla}\psi^{(1)}(Z)$;
        $\widehat{\mathcal{W}_2^2} \leftarrow \frac{1}{N} \sum_{x \in X} \|\boldsymbol{\nabla}\psi_\theta(x) - x\|_2^2$;
        $\widehat{\mathcal{U}} \leftarrow \frac{1}{N} \sum_{x \in X} \Phi\big(\boldsymbol{\nabla}\psi_\theta(x)\big)$;
        $\widehat{\Delta\mathcal{E}} \leftarrow \frac{1}{N} \sum_{x \in X} \log\det \nabla^2\psi_\theta(x)$;
        $\widehat{\mathcal{L}} \leftarrow \frac{1}{2h}\widehat{\mathcal{W}_2^2} + \widehat{\mathcal{U}} - \beta^{-1}\widehat{\Delta\mathcal{E}}$;
        Perform a gradient step over $\theta$ by using $\frac{\partial\widehat{\mathcal{L}}}{\partial\theta}$;
    $\psi^{(k)} \leftarrow \psi_\theta$
```

$x_k = \nabla\psi^{(k)}(x_{k-1})$. To compute x_{k-1}, one needs to solve the convex optimization problem:

$$x_k = \nabla\psi^{(k)}(x_{k-1}) \qquad \Longleftrightarrow \qquad x_{k-1} = \operatorname*{arg\,max}_{x \in \mathbb{R}^D} \big[\langle x, x_k\rangle - \psi^{(k)}(x)\big]. \tag{15}$$

If we know the density of ρ^0, to compute the density of $\rho^{(k)}$ at x_k we solve k convex problems

$$x_{k-1} = \operatorname*{arg\,max}_{x \in \mathbb{R}^D} \big[\langle x, x_k\rangle - \psi^{(k)}(x)\big] \qquad \ldots \qquad x_0 = \operatorname*{arg\,max}_{x \in \mathbb{R}^D} \big[\langle x, x_1\rangle - \psi^{(1)}(x)\big]$$

to obtain $x_{k-1}, \ldots, x_0$ and then evaluate the density as

$$\frac{d\rho_k}{dx}(x_k) = \frac{d\rho^0}{dx}(x_0) \cdot \big[\prod_{i=1}^{k} \det \nabla^2\psi^{(i)}(x_{i-1})\big]^{-1}.$$

Note the steps above provide a general method for tracing back the position of a particle along the flow, and density computation is simply a byproduct.

3.4 Nonlinear Filtering Theory

In this subsection we give a theoretical introduction to the Nonlinear Filtering as well as describe our practical implementation details.

Consider a diffusion process X_t governed by the Fokker-Planck equation (8). At times

$t_1 < t_2 < \cdots < t_K$ we obtain noisy observations from the process X_t:

$$Y_k = X_{t_k} + v_k,$$

with $v_k \sim \mathcal{N}(0, \sigma^2)$.The goal is to compute the predictive distribution of the process X_t at time $t \geq t_K$ given observations $Y_{1:K}$. We denote the associated probability density function as $p_{t,X}(x|Y_{1:K})$ and use the notation $\mu_{p_{t,X}(\,\cdot\,|Y_{1:K})}$ for the predictive distribution (probability measure) itself.

The dynamics of $p_{t,X}(x|Y_{1:K})$ is described by two alternating patterns. For each $k \in \{0, 1, \dots, K\}$ and $t > t_k$ the predictive distribution $\mu_{p_{t,X}(\,\cdot\,|Y_{1:k})}$ is called *marginal prior* and follows the equations (8) which govern the process X_t on the time segment $[t_k, t)$ with initial distribution $p_{t_k,X}(x|Y_{1:k})$.

If $t_k = t$ then the $\mu_{p_{t,X}(\,\cdot\,|Y_{1:k})}$ is called *marginal posterior* and it's probability density function follows the equation (in the below $p_{t,Y}(y)$ is the density of the process Y (X with noise) at time moment t):

$$\begin{aligned} p_{t_k,X}(x|Y_{1:k}) &= \frac{p_{t_k,X}(x|Y_{1:k-1}, Y_k)p_{t_k,Y}(Y_k|Y_{1:k-1})}{p_{t_k,Y}(Y_k|Y_{1:k-1})} = \frac{p_{t_k,X,Y}(x, Y_k|Y_{1:k-1})}{p_{t_k,Y}(Y_k|Y_{1:k-1})} \\ &= \frac{p_{t_k,Y}(Y_k|X_{t_k} = x, Y_{1:k-1})p_{t_k,X}(x|Y_{1:k-1})}{p_{t_k,Y}(Y_k|Y_{1:k-1})} \\ &= \frac{p_{t_k,Y}(Y_k|X_{t_k} = x)p_{t_k,X}(x|Y_{1:k-1})}{p_{t_k,Y}(Y_k|Y_{1:k-1})} \\ &\propto p(Y_k|X_{t_k} = x) \cdot p_{t_k,X}(x|Y_{1:k-1}). \end{aligned} \tag{16}$$

Predictive distribution update. For $k = 1, \dots, K$, we sequentially obtain the predictive distribution $\mu_{p_{t_k,X}(\,\cdot\,|Y_{1:k})}$ at time moment t_k using the previous predictive distribution $\mu_{p_{t_{k-1},X}(\,\cdot\,|Y_{1:k-1})}$ at time moment t_{k-1}. First, given sample access to $p_{t_{k-1},X}(x|Y_{1:k-1})$, we approximate the diffusion on the segment $[t_{k-1}, t_k)$ with initial distribution $\mu_{p_{t_{k-1},X}(\,\cdot\,|Y_{1:k-1})}$ by our Algorithm 1 to get access to *marginal prior* $\mu_{p_{t_k,X}(\,\cdot\,|Y_{1:k-1})}$. In particular, we perform n_k JKO steps of size $h_k = \frac{t_k - t_{k-1}}{n_k}$ and obtain ICNNs $\psi_1^{(k)}, \dots, \psi_{n_k}^{(k)}$ (approximately) satisfying

$$\mu_{p_{t_k,X}(\,\cdot\,|Y_{1:k-1})} = [\nabla\psi_{n_k}^{(k)} \circ \cdots \circ \nabla\psi_1^{(k)}]\sharp\mu_{p_{t_{k-1},X}(\,\cdot\,|Y_{1:k-1})} \tag{17}$$

In what follows we define $B_k \overset{def}{=} \nabla\psi_{n_k}^{(k)} \circ \cdots \circ \nabla\psi_1^{(k)}$.

Let $x_k \in \mathbb{R}^D$ and sequentially define $x_{i-1} = B_i^{-1}(x_i)$ for $i = k, \dots, 1$. We derive:

$$\begin{aligned} \underline{p_{t_k,X}(x_k|Y_{1:k})} &\overset{(16)}{\propto} \\ p(Y_k|X_{t_k} = x_k) \cdot p_{t_k,X}(x_k|Y_{1:k-1}) &\overset{(17)}{=} \\ p(Y_k|X_{t_k} = x_k) \cdot [\det \nabla B_k(x_{k-1})]^{-1} \cdot \underline{p_{t_{k-1},X}(x_{k-1}|Y_{1:k-1})} &\overset{(16)}{\propto} \\ &\dots \\ \prod_{i=1}^{k} p(Y_i|X_{t_i} = x_i) \cdot [\prod_{i=1}^{k} \det \nabla B_i(x_{i-1})]^{-1} \cdot \underline{p_{t_0,X}(x_0)} & \end{aligned} \tag{18}$$

where we substitute (16) sequentially for $k, k-1, \dots, 1$. As the result, from (18) we obtain the unnormalized density of *marginal posterior* predictive distribution $\mu_{p_{t_k,X}(\,\cdot\,|Y_{1:k})}$ at point x_k. In order to sample from this predictive distribution (to train the diffusion on the next segment $[t_k, t_{k+1})$) we use Metropolis-Hastings algorithm. For completeness, we recall this prominent result and general theory behind it.

Metropolis-Hastings Algorithm. Let $\pi : \mathbb{R}^D \to \mathbb{R}_+$ be a probability density function computable up to a multiplying constant $\pi(x) \propto \overline{\pi}(x)$. The Metropolis-Hastings algorithm [68] proposes a generic way to create irreducible and aperiodic Markov chain which converges to π. Let us briefly describe the main concepts which constitute the algorithm. At first, we introduce a family of proposal distributions $\{q_x(\cdot), x \in \mathbb{R}^D\}$, i.e $\int\limits_{\mathbb{R}^D} q_x(y)\mathrm{d}y = 1$ for any $x \in \mathbb{R}^D$. Then we construct a chain $x^{(1)}, x^{(2)}, \dots$ using the following procedure 2:

```
Algorithm 2: Metropolis-Hastings algorithm
    Input   :Unnormalized density $\pi(\cdot)$; family of proposal distributions $q_x(\cdot)$ $(x \in \mathbb{R}^D$
    Output :Sequence $x^{(1)}, x^{(2)}, x^{(3)}, \dots$ of samples from $\pi$
    Select $x^{(0)} \in \mathbb{R}^D$
    for $i = 1, 2, \dots$ do
        Sample $y \sim q_{x^{(i-1)}}$;
        Compute $\alpha(x^{(i-1)}, y) = \min\left(1, \frac{\pi(y)q_y(x^{(i-1)})}{\pi(x^{(i-1)})q_{x^{(i-1)}}(y)}\right)$
        With probability $\alpha(x^{(i-1)}, y)$ set $x^{(i)} \leftarrow y$; otherwise set $x^{(i)} \leftarrow x^{(i-1)}$
```

It was Metropolis et.al. [55] who showed that under some mild assumptions on the proposal distributions family the transitions in the Algorithm 2 preserve the stationary density π. Overall, it is sufficient for the distributions $q_x(\cdot)$ to be supported everywhere on $\mathbb{R}^D$.

To sample from $p_{t_k,X}(x_k|Y_{1:k})$ we use Algorithm 2 with π equal to unnormalized density (18). We note that computing $\pi(x_k)$ for $x_k \in \mathbb{R}^D$ is not easy since it requires computing preimages $x_{k-1}, \dots, x_0$ by inverting $B_k, B_{k-1}, \dots, B_1$. As the consequence, this makes computation of acceptance probability $\alpha(\cdot, \cdot)$ hard. To resolve this issue let's consider an arbitrary family of distributions $\{\xi_x(\cdot), x \in \mathbb{R}^D\}$ supported everywhere on $\mathbb{R}^D$. Define a family of proposal distributions $\{q_x(\cdot), x \in \mathbb{R}^D\}$ in a following way:

$$q_x \stackrel{\text{def}}{=} (B_k \circ B_{k-1} \circ \dots \circ B_1)\sharp\mu_{p_{0,X}}. \tag{19}$$

In other words, the proposal distribution q_x is given by the gradient flow on time interval $[0, t_k)$ with initial distribution ξ_x instead of $p_{0,X}$. Therefore, the sampling from q_x is equivalent to sampling from gradient flow, described in the Algorithm 1. Moreover, as a byproduct of sampling procedure we automatically obtain the intermediate flow preimages $(y_0, y_1, \dots, y_{k-1})$ of the point y under gradient flow transformations, i.e. $y_{k-1} = (B_k)^{-1}(y), y_{k-2} = (B_{k-1})^{-1}(y_{k-1}), \dots, y_0 = (B_1)^{-1}(y_1)$. We are left to compute acceptance probability $\alpha(x^{(i-1)}, y)$ in the equations below. In what follows we write $y = y_k$ and $x^{(i-1)} = x_k^{(i-1)}$.

$$\frac{\pi(y)q_y(x^{(i-1)})}{\pi(x^{(i-1)})q_{x^{(i-1)}}(y)}\Bigg|_{\pi(\cdot)\propto p_{t_k,X}(x|Y_{1:k})} \overset{18,\,19}{=}$$

$$\frac{p_{0,X}(y_0)\prod_{\tau=1}^{k}\frac{p_{t_\tau,Y}(Y_\tau|X_{t_\tau}=y_\tau)}{p_{t_\tau,Y}(Y_\tau|Y_{1:\tau-1})}\prod_{j=1}^{K}\text{det}\left[\nabla B_j(x_{j-1}^{(i-1)})\right]}{p_{0,X}(x_0^{(i-1)})\prod_{\tau=1}^{k}\frac{p_{t_\tau,Y}(Y_\tau|X_{t_\tau}=x_\tau^{(i-1)})}{p_{t_\tau,Y}(Y_\tau|Y_{1:\tau-1})}\prod_{j=1}^{K}\text{det}\left[\nabla B_j(y_{j-1})\right]}\cdot\frac{\xi_y(x_0^{(i-1)})\prod_{j=1}^{K}\text{det}\left[\nabla B_j(y_{j-1})\right]}{\xi_{x^{(i-1)}}(y_0)\prod_{j=1}^{K}\text{det}\left[\nabla B_j(x_{j-1}^{(i-1)})\right]} =$$

$$\frac{\xi_y(x_0^{(i-1)})p_{0,X}(y_0)}{\xi_{x^{(i-1)}}(y_0)p_{0,X}(x_0^{(i-1)})}\cdot\frac{\prod_{\tau=1}^{k}p_{t_\tau,Y}(Y_\tau|X_{t_\tau}=y_\tau)}{\prod_{\tau=1}^{k}p_{t_\tau,Y}(Y_\tau|X_{t_\tau}=x_\tau^{(i-1)})} \tag{20}$$

As we can see, all det terms in $\alpha(x, y)$ vanish and the formula 20 contains only known intermediate flow preimages of the points y and $x^{(i-1)}$ and can be straightforwardly computed given initial distributions and conditional noise distributions $p_{t,Y}(Y|X)$ in appropriate observation time moments $t_1, \dots, t_k$. In our particular case under consideration $p_{t_i,Y}(Y_i|X_{t_i} = x), i \in \{1, 2, \dots k\}$ are probability density functions of normal distributions $p_{\mathcal{N}(Y_i,\sigma^2)}(x)$ (recall that $Y_i \sim \mathcal{N}(X_{t_k}, \sigma^2)$).

As the class of "hidden" proposal distributions ξ_x in our experiments we use the initial x-independent gradient flow distribution, i.e $\xi_x(\,\cdot\,) = p_{0,X}(\,\cdot\,)$. This choice further simplifies the acceptance probability:

$$\frac{\pi(y)q_y(x^{(i-1)})}{\pi(x^{(i-1)})q_{x^{(i-1)}}(y)} = \frac{\prod_{i=1}^{k}p_{t_i,Y}(Y_i|X_{t_i}=y_i)}{\prod_{i=1}^{k}p_{t_i,Y}(Y_i|X_{t_i}=x_i)} \tag{21}$$

We summarize all our findings in the Algorithm 3 in the Appendix I. It represents a modified version of Algorithm 1 with embedded MCMC sampling technique based on Metropolis-Hastings algorithm. Note, that in order to decorrelate the MCMC chain we introduce decorrelation parameter K_d and take each K_d-th sample. Also we warm-up the chain omitting the first N_w samples.

We leave the section with a note, that the formula 20 permits an alternative choice for "hidden" proposal distribution ξ_x which potentially could improve the stability and quality of MCMC. This opens a direction for future research.

4 Experiments and Results

In this section, we evaluate our method on toy and real-world applications. Our code is written in PyTorch and is publicly available at

`https://github.com/PetrMokrov/Large-Scale-Wasserstein-Gradient-Flows`

The experiments are conducted on a GTX 1080Ti. In most cases, we performed several random restarts to obtain mean and variation of the considered metric. As the result, experiments require about 100-150 hours of computation.

4.1 General Experimental Details

Neural network architectures. In all experiments, we use the DenseICNN [44, Appendix B.2] architecture for ψ_θ in Algorithm 1 with *SoftPlus* activations. The network ψ_θ is twice differentiable w.r.t. the input x and has bijective gradient $\nabla\psi_\theta : \mathbb{R}^D \to \mathbb{R}^D$ with positive semi-definite Hessian $\nabla^2\psi_\theta(x) \succeq 0$ at each x. We use automatic differentiation to compute $\nabla\psi_\theta$ and $\nabla^2\psi_\theta$. Throughout our experiments we set the number of hidden layers to be equal 2 and vary the width of the model depending on the task. We use Adam optimizer with learning rate decreasing with the number of JKO steps. We initialize the ICNN models either via pretraining to satisfy $\boldsymbol{\nabla}\psi_\theta(x) \approx x$ or by using parameters θ obtained from the previous JKO step.

Metric. To qualitatively compare measures, we use the symmetric Kullback-Leibler divergence

$$\text{SymKL}(\rho_1, \rho_2) \overset{\text{def}}{=} \text{KL}(\rho_1 \| \rho_2) + \text{KL}(\rho_2 \| \rho_1), \tag{22}$$

where $\text{KL}(\rho_1 \| \rho_2) \overset{\text{def}}{=} \int_{\mathbb{R}^D} \log \frac{d\rho_1}{d\rho_2}(x) d\rho_1(x)$ is the Kullback-Leibler divergence. For particle-based methods, we obtain an approximation of the distribution by kernel density estimation.

Competitive methods details. For $\lfloor$Dual JKO$\rceil$, we used the implementation provided by the authors with default hyper-parameters. For $\lfloor$EM PR$\rceil$ we implemented the Proximal Recursion operator following the pseudocode of [20] and used the default hyper-parameters but we increased the number of particles for fair comparison with the vanilla $\lfloor$EM$\rceil$ algorithm. Note we limited the number of particles to $N = 10^4$ because of the high computational complexity of the method. For $\lfloor$SVGD$\rceil$, we used the official implementation available at

`https://github.com/dilinwang820/Stein-Variational-Gradient-Descent`

In particle-based simulations $\lfloor$EM$\rceil$, $\lfloor$BBF$\rceil$ and $\lfloor$EM PR$\rceil$ we used the particle propagation timestep $dt = 10^{-3}$.

We estimate the SymKL (22) using Monte Carlo (MC) on 10^4 samples. In our method, MC estimate is straightforward since the method permits both sampling and computing the density. In particle-based methods, we use kernel density estimator to approximate the density utilizing `scipy` implementation of `gaussian_kde` with `bandwidth` chosen by Scott's rule. In $\lfloor$Dual JKO$\rceil$, we employ importance sampling procedure and normalization constant estimation as detailed in [31].

We set β to be equal to 1 throughout our experiments.

4.2 Convergence to Stationary Solution

Starting from an arbitrary initial measure ρ^0, an advection-diffusion process (8) converges to the unique stationary solution ρ^* [67] with density

$$\frac{d\rho^*}{dx}(x) = Z^{-1} \exp(-\beta\Phi(x)), \tag{23}$$

D	M	l	w
2	5	10	256
4	6	10	384
6	7	10	512
8	8	10	512
10	9	10	512
12	10	10	1024
13	10	10	512
32	10	6	1024

Table 1: Hyper-parameters in the convergence exp.

where $Z = \int_{\mathbb{R}^D} \exp(-\beta\Phi(x))dx$ is the normalization constant. This property makes it possible to compute the symmetric KL between the distribution to which our method converges and the ground truth, provided Z is known.

We use $\mathcal{N}(0, 16I_D)$ as the initial measure ρ^0 and a random Gaussian mixture $\frac{1}{N_p}\sum_{m=1}^{M}\mathcal{N}(\mu_m, I_D)$, where $\mu_1, \dots, \mu_M \sim \text{Uniform}\big([-\frac{l}{2}, \frac{l}{2}]^D\big)$ as the stationary measure ρ^*. We set the width w of used ICNNs ψ_θ depending on dimension D. The parameters are summarized in Table 1. In our method, we perform $K = 40$ JKO steps with step size $h = 0.1$. We compare with a particle simulation method (with $10^3, 10^4, 10^5$ particles) based on the Euler-Maruyama $\lfloor\text{EM}\rceil$ approximation [43, §9.2]. We repeat the experiment 5 times and report the averaged results in Figure 2.

Each JKO step uses 1000 gradient descent iterations of Algorithm 1. For dimensions $D = 2, 4, \dots, 12$ the first 20 JKO transitions are optimized with $lr = 5 \cdot 10^{-3}$ and the remaining steps use $lr = 2 \cdot 10^{-3}$. For qualitative experiments in $D = 13, 32$ we perform 50 and 70 JKO steps with step size $h = 0.1$. The learning rate setup in these cases is similar to quantitative experiment setting but has additional stage with $lr = 5 \cdot 10^{-4}$ on the final JKO steps. The batch size is $N = 512$.

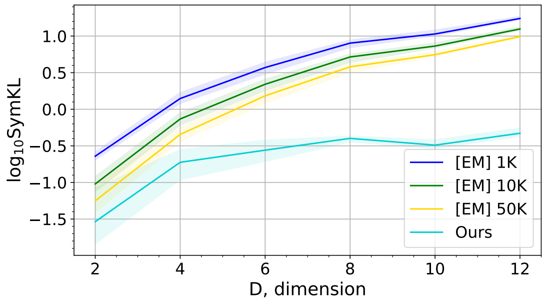
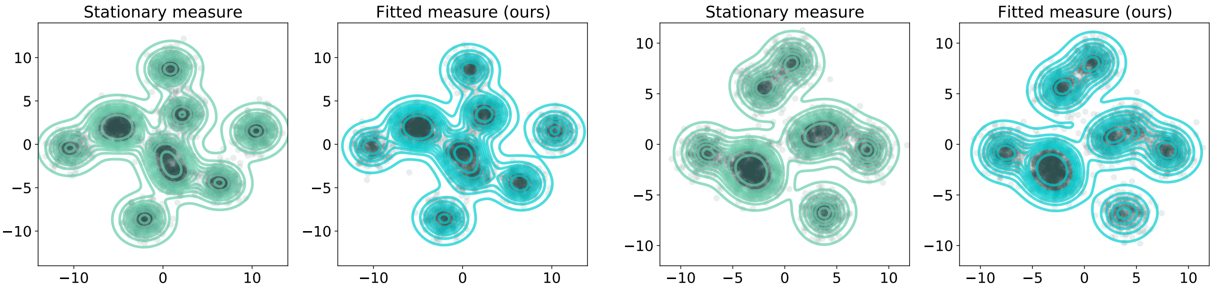


Figure 2: SymKL between the computed and the stationary measure in $D = 2, 4, \dots 12$

In Figure 3, we present qualitative results of our method converging to the ground truth in $D = 13, 32$.



(a) Dimension $D = 13$

(b) Dimension $D = 32$

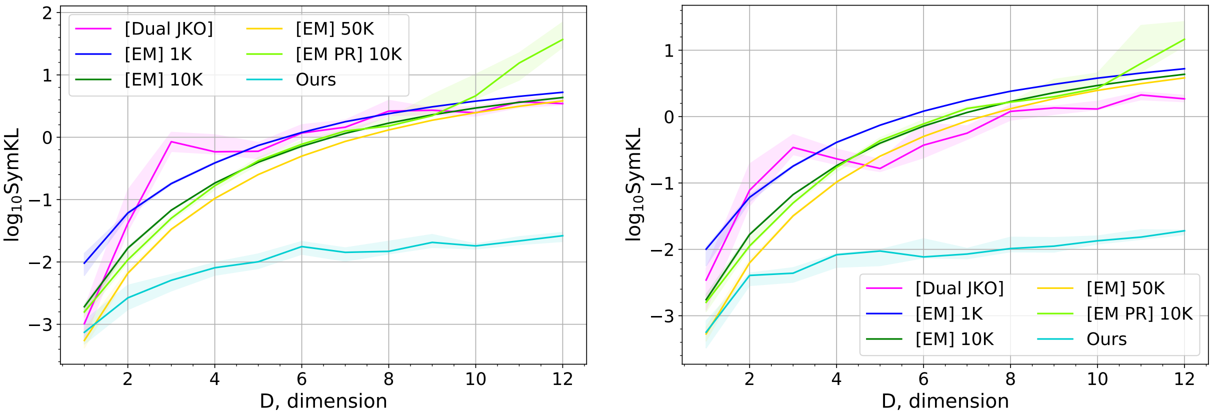
Figure 3: Projections to 2 first PCA components of the true stationary measure and the measure approximated by our method in dimensions $D = 13$ (on the left) and $D = 32$ (on the right).

4.3 Modeling Ornstein-Uhlenbeck Processes

Ornstein-Uhlenbeck processes are advection-diffusion processes (8) with $\Phi(x) = \frac{1}{2}(x-b)^T A(x-b)$ for symmetric positive definite $A \in \mathbb{R}^{D\times D}$ and $b \in \mathbb{R}^D$. They are among the few examples where we know ρ_t for any $t \in \mathbb{R}^+$ in closed form, when the initial measure ρ^0 is Gaussian [77]. This allows to quantitatively evaluate the computed dynamics of the process, not just the stationary measure.

We choose matrices $A \in \mathbb{R}^{D\times D}$ to be randomly generated using `sklearn.datasets.make_spd_matrix`. Vectors $b \in \mathbb{R}^D$ are sampled from standard Gaussian measure. All ICNNs ψ_θ have $w = 64$ and we train each of them for 500 iterations per JKO step with $lr = 5 \cdot 10^{-3}$ and batch size $N = 1024$. For the entropy regularized JKO method we follow the recommendations by authors of [31]. The only difference is that for each dimension we select the support of the kernels which results in the best SymKL metric value. The details are given in our code.

We set ρ^0 to be the standard Gaussian measure $\mathcal{N}(0, I_D)$ and approximate the dynamics of the process by our method with JKO step $h = 0.05$ and compute SymKL between the true ρ_t and the approximate one at time $t = 0.5$ and $t = 0.9$. We repeat the experiment 15 times in dimensions $D = 1, 2 \ldots, 12$ and report the performance at in Figure 4. The baselines are $\lfloor$EM$\rceil$ with $10^3, 10^4, 5 \times 10^4$ particles, EM particle simulation endowed with the Proximal Recursion operator $\lfloor$EM PR$\rceil$ with 10^4 particles [20], and the parametric dual inference method [31] for JKO steps $\lfloor$Dual JKO$\rceil$. The detailed comparison for times $t = 0.1, 0.2, \ldots 1$ is given in Appendix II.



(a) Time $t = 0.5$

(b) Time $t = 0.9$

Figure 4: SymKL values between the computed measure and the true measure ρ_t at $t = 0.5$ (on the left) and $t = 0.9$ (on the right) in dimensions $D = 1, 2, \ldots, 12$. Best viewed in color.

4.4 Unnormalized Posterior Sampling in Bayesian Logistic Regression

An important task in Bayesian machine learning to which our algorithm can be applied is sampling from an unnormalized posterior distribution. Given the model parameters $x \in \mathbb{R}^D$ with the prior distribution $p_0(x)$ as well as the conditional density $p(\mathcal{S}|x) = \prod_{m=1}^{M} p(s_m|x)$ of the data $\mathcal{S} =$

Dataset	Accuracy		Log-Likelihood	
	Ours	⌈SVGD⌋	Ours	⌈SVGD⌋
covtype	0.75	0.75	-0.515	-0.515
german	0.67	0.65	-0.6	-0.6
diabetis	0.775	0.78	-0.45	-0.46
twonorm	0.98	0.98	-0.059	-0.062
ringnorm	0.74	0.74	-0.5	-0.5
banana	0.55	0.54	-0.69	-0.69
splice	0.845	0.85	-0.36	-0.355
waveform	0.78	0.765	-0.485	-0.465
image	0.82	0.815	-0.43	-0.44

Table 2: Comparison of our method with ⌈SVGD⌋ [50] for Bayesian log. regression.

Dataset	w	lr	$iter$	batch	K
covtype	512	$2 \cdot 10^{-5}$	10^4	1024	6
german	512	$2 \cdot 10^{-4}$	5000	512	5
diabetis	128	$5 \cdot 10^{-5}$	6000	1024	16
twonorm	512	$5 \cdot 10^{-5}$	5000	1024	7
ringnorm	512	$5 \cdot 10^{-5}$	5000	1024	2
banana	128	$2 \cdot 10^{-4}$	5000	1024	5
splice	512	$2 \cdot 10^{-3}$	2000	512	5
waveform	512	$5 \cdot 10^{-5}$	5000	512	2
image	512	$5 \cdot 10^{-5}$	5000	512	5

Table 3: Hyper-parameters we use in Bayesian log. regression experiment.

$\{s_1, \dots, s_M\}$, the posterior distribution is given by

$$p(x|\mathcal{S}) = \frac{p(\mathcal{S}|x)p_0(x)}{p(\mathcal{S})} \propto p(\mathcal{S}|x)p_0(x) = p_0(x) \cdot \prod_{m=1}^{M} p(s_m|x).$$

Computing the normalization constant $p(\mathcal{S})$ is in general intractable, underscoring the need for estimation methods that sample from $p(\mathcal{S}|x)$ given the density only up to a normalizing constant.

In our context, sampling from $p(x|\mathcal{S})$ can be solved similarly to the task in §4.2. From (23), it follows that the advection-diffusion process with temperature $\beta > 0$ and $\Phi(x) = -\frac{1}{\beta} \log \big[p_0(x) \cdot p(\mathcal{S}|x)\big]$ has $\frac{d\rho^*}{dx}(x) = p(x|\mathcal{S})$ as the stationary distribution. Thus, we can use our method to approximate the diffusion process and obtain a sampler for $p(x|\mathcal{S})$ as a result.

The potential energy $\mathcal{U}(\rho) = \int_{\mathbb{R}^D} \Phi(x) d\rho(x)$ can be estimated efficiently by using a trick similar to the ones in stochastic gradient Langevin dynamics [80], which consists in resampling samples in $\mathcal{S}$ uniformly. For evaluation, we consider the Bayesian linear regression setup of [50]. We use the 8 datasets from [56]. The number of features ranges from 2 to 60 and the dataset size from 700 to 7400 data points. We also use the Covertype dataset[1] with 500K data points and 54 features. The prior on regression weights w is given by $p_0(w|\alpha) = \mathcal{N}(w|0, \alpha^{-1})$ with $p_0(\alpha) = \text{Gamma}(\alpha|1, 0.01)$, so the prior on parameters $x = [w, \alpha]$ of the model is given by $p_0(x) = p_0(w, \alpha) = p_0(w|\alpha) \cdot p_0(\alpha)$. To remove positiveness constraint on α we consider $[w, \log(\alpha)]$ as the regression model parameters instead of $[w, \alpha]$. To learn the posterior distribution $p(x|S_{\text{train}})$ we use JKO step size $h = 0.1$. Let $iter$ denote the number of gradient steps over θ per each JKO step and K denote the overall number of JKO steps. The used hyper-parameters for each dataset are summarized in Table 3.

To estimate the log-likelihood and accuracy of the predictive distribution on S_{test} based on $p(x|S_{\text{train}})$, we use straightforward MC estimate on 2^{12} random parameter samples. We randomly split each dataset into train $\mathcal{S}_{\text{train}}$ and test $\mathcal{S}_{\text{test}}$ ones with ratio 4:1 and apply the inference on the posterior $p(x|\mathcal{S}_{\text{train}})$. In Table 2, we report accuracy and log-likelihood of the predictive distribution on $\mathcal{S}_{\text{test}}$. As the baseline, we use particle-based Stein Variational Gradient Descent [50]. We use

---

[1] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

the author's implementation with the default hyper-parameters.

4.5 Nonlinear Filtering

We demonstrate the application of our method to filtering a nonlinear diffusion. See §3.4 for the theoretical introduction and algorithmic details.

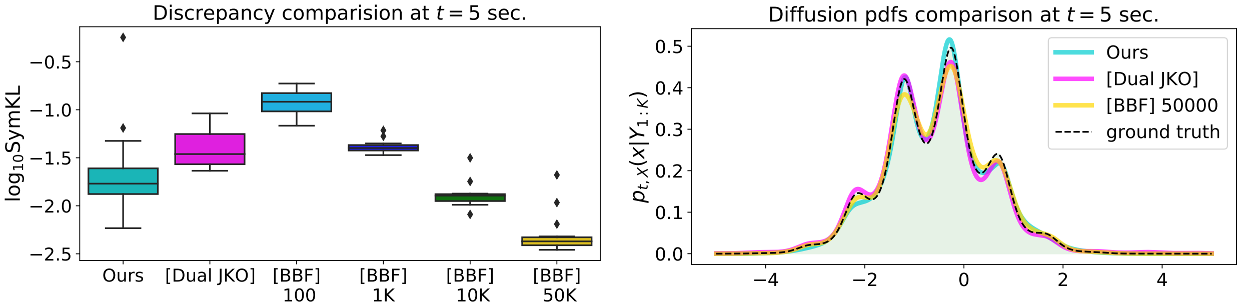
For evaluation, we consider the experimental setup of [31, §6.3]. We assume that the 1-dimensional diffusion process X_t has potential function $\Phi(x) = \frac{1}{\pi}\sin(2\pi x) + \frac{1}{4}x^2$ which makes the process highly nonlinear. We simulate nonlinear filtering on the time interval $t_{\text{start}} = 0$ sec., $t_{\text{fin}} = 5$ sec. and take the noise observations each 0.5 sec. The noise variance is $\sigma^2 = 1$ and the distribution of X_0 is $\mathcal{N}(X_0|0,1)$. Specifically, to obtain the noise observations $Y_k = X_{t_k} + v_k$ from the process, we simulate a particle X_0 randomly sampled from the initial measure $\mathcal{N}(0,1)$ by using Euler-Maruyama method to obtain the trajectory X_t. At observation times $t_1 = 0.5, \dots, t_9 = 4.5$ we add random noise $v_k \sim \mathcal{N}(0,1)$ to obtain observations $Y_1, \dots, Y_9$.

We predict the conditional density $p_{t_{\text{final}},X}(x|Y_{1:9})$ and compare the prediction with ground truth obtained with numerical integration method by Chang and Cooper [25], who use a fine discrete grid. When implementing their method, we construct regular fine grid on the segment $[-5, 5]$ with 2000 points and numerically solve the SDE with timestep $dt = 10^{-3}$. At observation times t_k, $k \in 1, \dots 9$ we multiply the obtained probability density function $p_{t_k,X}(x|Y_{1:k-1})$ by the density of the normal distribution $p(Y_k|X_{t_k} = x)$ estimated at the grid which results in unnormalized $p_{t_k,X}(x|Y_{1:k})$. After normalization on the grid, $p_{t_k,X}(x|Y_{1:k})$ can be used in the new diffusion round on time interval $[t_k, t_{k+1}]$. At final time t_{fin} we estimate SymKL between the true distribution and ones obtained via other competitive methods by numerically integrating (22) on the grid.

As the baselines, we use ⌊Dual JKO⌉ [31] as well as the Bayesian Bootstrap filter ⌊BBF⌉ [34], which combines particle simulation with bootstrap resampling at observation times. We implement ⌊BBF⌉ following the original article [34]. Particle propagation performed via Euler-Maruyama method with timestep $dt = 10^{-3}$. The final distribution $p(X_{t_{\text{fin}}}|Y_{1:9})$ is estimated using kernel density estimator. For ⌊Dual JKO⌉ we use the code provided by the authors with the default hyper-parameters.

In our method, we use JKO step size $h = 0.1$ and model it by ICNN with width $w = 256$. Each JKO step takes 700 optimization iterations with $lr = 5 \cdot 10^{-3}$ and batch size $N = 1024$. At observation times t_k, $k \in 1, 2, \dots 9$ we use the Metropolis-Hastings algorithm 2 with acceptance probability α calculated by (20). Starting from the randomly sampled $x^{(1)}$ we skip the first 1000 values of the Markov Chain generated by the algorithm which allows the series to converge to the distribution of interest $p_{t_k,X}(x|Y_{1:k})$. We take each second element from the chain in order to decorrelate the samples. To simultaneously sample the batch of size N, we run N chains in parallel. To compute SymKL, we normalize the resulting distribution $p(X_{t_{\text{fin}}}|Y_{1:9})$ on the Chang-Cooper support grid.

We repeat the experiment 15 times. In Figure 5a, we report the SymKL between predicted density and true $p(X_{t_{\text{fin}}}|Y_{1:9})$. We visually compare the fitted and true conditional distributions in Figure 5b.



(a) SymKL values.

(b) Visualized probability density functions.

Figure 5: Comparison of the predicted conditional density and true $p(X_{t_{\text{fin}}}|Y_{1:9})$.

5 Discussion

5.1 Further applications

We apply our method to common Bayesian tasks such as unnormalized posterior sampling (§4.4) and nonlinear filtering (§4.5). Below we mention several other potential applications:

- **Population dynamics.** In this task, one needs to recover the potential energy $\Phi(x)$ included in the Fokker-Planck free energy functional $\mathcal{F}_{\text{FP}}$ based on samples from the diffusion obtained at timesteps $t_1, \dots, t_n$, see [36]. This setting can be found in computational biology, see §6.3 of [36]. A recent paper [17] utilizes ICNN-powered JKO to model population dynamics and successfully models single-cell RNA sequencing data.
- **Reinforcement learning**. Wasserstein gradient flows provide a theoretically-grounded way to optimize an agent policy in reinforcement learning, see [66, 83]. The idea of the method is to maximize the expected total reward (see (10) in [83]) using the gradient flow associated with the Fokker-Planck functional (see (12) in [83]). The authors of the original paper proposed discrete particle approximation method to solve the underlying JKO scheme. Substituting their approach with our ICNN-based JKO can potentially improve the results.
- **Refining Generative Adversarial Networks**. In the GAN setting, given trained generator G and discriminator D, one can improve the samples from G by D via considering a gradient flow w.r.t. entropy-regularized f-divergence between real and generated data distribution (see [8], in particular, formula (4) for reference). Using KL-divergence makes the gradient flow consistent with our method: the functional $\mathcal{F}$ defining the flow has only entropic and potential energy terms. The usage of our method instead of particle simulation may improve the generator model.
- **Molecular Discovery.** In [3], in parallel to our work the JKO-ICNN scheme is proposed. The authors consider the molecular discovery as an application. The task is to increase the *drug-likeness* of a given distribution ρ of molecules while staying close to the original distribution ρ_0. The task reduces to optimizing the functional $\mathcal{F}(\rho) = \mathbb{E}_{x\sim\rho}\Phi(x) + \mathcal{D}(\rho, \rho_0)$ for a certain

potential Φ (V - in the notation of [3]) and a discrepancy $\mathcal{D}$. The authors applied the JKO-ICNN method to minimize $\mathcal{F}$ on MOSES [64] molecular dataset and obtained promising results.

5.2 Complexity of training and sampling.

Let T be the number of operations required to evaluate ICNN $\psi_\theta(x)$, and assume that the evaluation of $\Phi(x)$ in the potential energy $\mathcal{U}$ takes $O(1)$ time.

Recall that computing the gradient is a small constant factor harder than computing the function itself [49]. Thus, evaluation of $\nabla\psi_\theta(x) : \mathbb{R}^D \to \mathbb{R}^D$ requires $O(T)$ operations and evaluating the Hessian $\nabla^2\psi_\theta(x) : \mathbb{R}^D \to \mathbb{R}^{D\times D}$ takes $O(DT)$ time. To compute $\log\det\nabla^2\psi_\theta(x)$, we need $O(D^3)$ extra operations. Sampling from $\rho^{(k-1)} = \nabla\psi^{(k-1)} \circ \cdots \circ \nabla\psi^{(1)}\sharp\rho_0$ involves pushing $x_0 \sim \rho^0$ forward by a sequence of ICNNs $\psi^{(\cdot)}$ of length $k-1$, requiring $O\big((k-1)T\big)$ operations. The forward pass to evaluate the JKO step objective $\widehat{\mathcal{L}}$ in Algorithm 1 requires $O(DT+D^3)$ operations, as does the backward pass to compute the gradient $\frac{\partial\widehat{\mathcal{L}}}{\partial\theta}$ w.r.t. θ.

Operation	Time Complexity
Eval. ψ_θ,$\nabla\psi_\theta$,$\nabla^2\psi_\theta$	$T, O(T), O(DT)$
Eval. $\log\det\nabla^2\psi_\theta$	$O(DT+D^3)$
Sample $x \sim \rho^{(k)}$	$O\big((k-1)T\big)$
Eval. $\widehat{\mathcal{L}}$ on $x \sim \rho^{(k)}$	$O(DT + D^3)$
Eval. $\frac{\partial\widehat{\mathcal{L}}}{\partial\theta}$ on $x \sim \rho^{(k)}$	$O(DT+D^3)$
Sample $x \sim \rho^{(k)}$ and Eval. $\frac{d\rho^{(k)}}{dx}(x)$	$O\big((k-1)(TD+D^3)\big)$

Table 4: Complexity of operations in our method for computing JKO steps via ICNNs.

The *memory complexity* is more difficult to characterize, since it depends on the autodiff implementation. It does not exceed the time complexity and is linear in the number of JKO steps k.

Wall-clock times. All particle-based methods considered in §4 and $\lfloor$Dual JKO$\rceil$ require from several seconds to several minutes CPU computation time. Our method requires from several minutes to few hours on GPU, the time is explained by the necessity to train a new network at each step.

Advantages. Due to using continuous approximation, our method scales well to high dimensions, as we show in §4.2 and §4.3. After training, we can produce infinitely many samples $x_k \sim \rho^{(k)}$, together with their trajectories $x_{k-1}, x_{k-2}, \dots, x_0$ along the gradient flow. Moreover, the densities of samples in the flow $\frac{d\rho^{(k)}}{dx}(x_k)$, $\frac{d\rho^{(k-1)}}{dx}(x_{k-1}), \dots, \frac{d\rho^{(0)}}{dx}(x_0)$ can be evaluated immediately.

In contrast, particle-based and domain discretization methods do not scale well with the dimension (Figure 4) and provide no density. Interestingly, despite its parametric approximation, $\lfloor$Dual JKO$\rceil$ performs comparably to particle simulation and worse than ours (see additionally [31, Figure 3]).

Limitations. To train k JKO steps, our method requires time proportional to k^2 due to the increased complexity of sampling $x \sim \rho^{(k)}$. This may be disadvantageous for training long diffusions. In addition, for very high dimensions D, exact evaluation of $\log\det\nabla^2\psi_\theta(x)$ is time-consuming.

Future work. To reduce the computational complexity of sampling from $\rho^{(k)}$, at step k one may regress an invertible network $H : \mathbb{R}^D \to \mathbb{R}^D$ [10, 38] to satisfy $H(x_0) \approx \nabla\psi^{(k)} \circ \cdots \circ \nabla\psi^{(1)}(x_0)$ and use $H\sharp\rho_0 \to \rho^{(k)}$ to simplify sampling. An alternative is to use variational inference [12, 65, 82] to approximate $\rho^{(k)}$. These ideas have already been partially exploited by [30]. To mitigate the computational complexity of computing $\log\det\nabla\psi_\theta(x)$, fast approximation can be used [76, 35, 3]. More broadly, developing ICNNs with easily-computable exact Hessians is a critical avenue for further research as ICNNs continue to gain attention in machine learning [52, 44, 46, 37, 29, 6].

6 Conclusions

The novel approach for modelling Fokker-Planck equation is proposed. It is based on JKO scheme combined with recently developed Input Convex Neural Networks which provide a rich set of convex functions amenable to optimization procedures. The experiments shows that the methods works competitively in several model and real-world tasks including Bayesian logistic regression and nonlinear filtering. The advantages and limitations of the approach are analyzed and possible future directions are proposed.

7 Author Contribution

The author implemented the code covering the research (the implementation of ICNN was taken from [44]) as well as contributed to the development of theoretical and practical methods underlining research methodology and applications. In particular, it was the author who proposed Nonlinear Filtering problem reformulation via ICNN powered JKO (see §3.4). Additionally, the author helped a lot with preparing the text of [57] which forms the basis of the work under consideration.

8 Publications

The work is primarily based on the paper [57], written by the author of the work with co-authors.

9 Acknowledgements

The author thanks Lingxiao Li, Aude Genevay and Justin Solomon for useful discussions and comments when preparing the article [57]. The author expresses special gratitude to Alexander Korotin who helped a lot with the manuscript and granted a lot of suggestions regarding theoretical statements and implementation details. Evgeny Burnaev provided the author with help and support throughout the work.

Bibliography

[1] Juha Ala-Luhtala, Simo Särkkä, and Robert Piché. Gaussian filtering and variational approximations for Bayesian smoothing in continuous-discrete stochastic dynamic systems. *Signal Processing*, 111:124–136, 2015.

[2] David Alvarez-Melis and Nicolò Fusi. Gradient flows in dataset space. *arXiv preprint arXiv:2010.12760*, 2020.

[3] David Alvarez-Melis, Yair Schiff, and Youssef Mroueh. Optimizing functionals on the space of probabilities with input convex neural networks. *arXiv preprint arXiv:2106.00774*, 2021.

[4] Luigi Ambrosio and Nicola Gigli. *A User's Guide to Optimal Transport*, pages 1–155. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[5] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[6] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[7] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

[8] Abdul Fatir Ansari, Ming Liang Ang, and Harold Soh. Refining deep generative models via Wasserstein gradient flows. *arXiv preprint arXiv:2012.00780*, 2020.

[9] Michael Arbel, Anna Korba, Adil Salim, and Arthur Gretton. Maximum mean discrepancy gradient flow. *arXiv preprint arXiv:1906.04370*, 2019.

[10] Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

[11] Jean-David Benamou, Guillaume Carlier, Quentin Mérigot, and Edouard Oudet. Discretization of functionals involving the Monge–Ampère operator. *Numerische mathematik*, 134(3):611–636, 2016.

[12] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[13] V.I. Bogachev. *Measure Theory*. Measure Theory. Springer Berlin Heidelberg, 2007.

[14] Clément Bonet, Nicolas Courty, François Septier, and Lucas Drumetz. Sliced-wasserstein gradient flows, 2021.

[15] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[16] Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

[17] Charlotte Bunne, Laetitia Meng-Papaxanthos, Andreas Krause, and Marco Cuturi. Proximal optimal transport modeling of population dynamics, 2021.

[18] Felix Bünning, Adrian Schalbetter, Ahmed Aboudonia, Mathias Hudoba de Badyn, Philipp Heer, and John Lygeros. Input convex neural networks for building MPC. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pages 251–262. PMLR, 07 – 08 June 2021.

[19] Martin Burger, José A Carrillo, and Marie-Therese Wolfram. A mixed finite element method for nonlinear diffusion equations. *Kinetic & Related Models*, 3(1):59, 2010.

[20] Kenneth F. Caluya and Abhishek Halder. Proximal recursion for solving the fokker-planck equation, 2019.

[21] Pierre Cardaliaguet and Alessio Porretta. *An Introduction to Mean Field Game Theory*, pages 1–158. Springer International Publishing, Cham, 2020.

[22] René Carmona and François Delarue. *Probabilistic Theory of Mean Field Games with Applications I*. Springer, 2018.

[23] José A Carrillo, Alina Chertock, and Yanghong Huang. A finite-volume method for nonlinear nonlocal equations with a gradient flow structure. *Communications in Computational Physics*, 17(1):233–258, 2015.

[24] Villani Cédric. *Topics in optimal transportation / Cédric Villani*. Graduate studies in mathematics. American mathematical society, Providence, Rhode Island, C 2003.

[25] J.S Chang and G Cooper. A practical difference scheme for fokker-planck equations. *Journal of Computational Physics*, 6(1):1–16, 1970.

[26] Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach. *arXiv preprint arXiv:1805.11835*, 2018.

[27] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[28] Nicole El Karoui, Shige Peng, and Marie Claire Quenez. Backward stochastic differential equations in finance. *Mathematical finance*, 7(1):1–71, 1997.

[29] Jiaojiao Fan, Amirhossein Taghvaei, and Yongxin Chen. Scalable computations of wasserstein barycenter via input convex neural networks. *arXiv preprint arXiv:2007.04462*, 2020.

[30] Jiaojiao Fan, Qinsheng Zhang, Amirhossein Taghvaei, and Yongxin Chen. Variational wasserstein gradient flow, 2021.

[31] Charlie Frogner and Tomaso Poggio. Approximate inference with Wasserstein gradient flows. In *International Conference on Artificial Intelligence and Statistics*, pages 2581–2590. PMLR, 2020.

[32] Yuan Gao, Yuling Jiao, Yang Wang, Yao Wang, Can Yang, and Shunkang Zhang. Deep generative learning via variational gradient flow. In *International Conference on Machine Learning*, pages 2093–2101. PMLR, 2019.

[33] Yuan Gao, Guangzhen Jin, and Jian-Guo Liu. Inbetweening auto-animation via fokker-planck dynamics and thresholding. *arXiv preprint arXiv:2005.08858*, 2020.

[34] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. 1993.

[35] Insu Han, Dmitry Malioutov, and Jinwoo Shin. Large-scale log-determinant computation through stochastic chebyshev expansions. In *International Conference on Machine Learning*, pages 908–917. PMLR, 2015.

[36] Tatsunori Hashimoto, David Gifford, and Tommi Jaakkola. Learning population-level diffusions with generative rnns. In *International Conference on Machine Learning*, pages 2417–2426. PMLR, 2016.

[37] Chin-Wei Huang, Ricky TQ Chen, Christos Tsirigotis, and Aaron Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. *arXiv preprint arXiv:2012.05942*, 2020.

[38] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

[39] Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

[40] Simon J Julier, Jeffrey K Uhlmann, and Hugh F Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of 1995 American Control Conference-ACC'95*, volume 3, pages 1628–1632. IEEE, 1995.

[41] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. 1961.

[42] Søren Klim, Stig Bousgaard Mortensen, Niels Rode Kristensen, Rune Viig Overgaard, and Henrik Madsen. Population stochastic modelling (psm)—an r package for mixed-effects models based on stochastic differential equations. *Computer methods and programs in biomedicine*, 94(3):279–289, 2009.

[43] Peter E. Kloeden. *Numerical solution of stochastic differential equations / Peter E. Kloeden, Eckhard Platen*. Applications of mathematics; v. 23. Springer, Berlin, 1992.

[44] Alexander Korotin, Vage Egiazarian, Arip Asadulaev, Alexander Safin, and Evgeny Burnaev. Wasserstein-2 generative networks. In *International Conference on Learning Representations*, 2021.

[45] Alexander Korotin, Lingxiao Li, Aude Genevay, Justin Solomon, Alexander Filippov, and Evgeny Burnaev. Do neural optimal transport solvers work? a continuous wasserstein-2 benchmark. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[46] Alexander Korotin, Lingxiao Li, Justin Solomon, and Evgeny Burnaev. Continuous wasserstein-2 barycenter estimation without minimax optimization. In *International Conference on Learning Representations*, 2021.

[47] Harold Kushner. Approximations to optimal nonlinear filters. *IEEE Transactions on Automatic Control*, 12(5):546–556, 1967.

[48] Hugo Lavenant, Sebastian Claici, Edward Chien, and Justin Solomon. Dynamical optimal transport on discrete surfaces. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018.

[49] Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.

[50] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. *arXiv preprint arXiv:1608.04471*, 2016.

[51] Antoine Liutkus, Umut Simsekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter. Sliced-Wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *International Conference on Machine Learning*, pages 4104–4113. PMLR, 2019.

[52] Ashok Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason Lee. Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pages 6672–6681. PMLR, 2020.

[53] Bertrand Maury, Aude Roudneff-Chupin, and Filippo Santambrogio. A macroscopic crowd motion model of gradient flow type. *Mathematical Models and Methods in Applied Sciences*, 20(10):1787–1821, 2010.

[54] Robert J McCann et al. Existence and uniqueness of monotone measure-preserving maps. *Duke Mathematical Journal*, 80(2):309–324, 1995.

[55] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[56] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, pages 41–48. Ieee, 1999.

[57] Petr Mokrov, Alexander Korotin, Lingxiao Li, Aude Genevay, Justin Solomon, and Evgeny Burnaev. Large-scale wasserstein gradient flows. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[58] G. Monge. *Mémoire sur la théorie des déblais et des remblais*. Imprimerie royale, 1781.

[59] Manfred Opper. Variational inference for stochastic differential equations. *Annalen der Physik*, 531(3):1800233, 2019.

[60] Lorenzo Pareschi and Mattia Zanella. Structure preserving schemes for nonlinear fokker–planck equations and applications. *Journal of Scientific Computing*, 74(3):1575–1600, 2018.

[61] Gabriel Peyré. Entropic approximation of Wasserstein gradient flows. *SIAM Journal on Imaging Sciences*, 8(4):2323–2351, 2015.

[62] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[63] Eckhard Platen and Nicola Bruti-Liberati. *Numerical solution of stochastic differential equations with jumps in finance*, volume 64. Springer Science & Business Media, 2010.

[64] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alán Aspuru-Guzik, and Alex Zhavoronkov. Molecular sets (MOSES): A benchmarking platform for molecular generation models. *Frontiers in Pharmacology*, 11:1931, 2020.

[65] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[66] Pierre H Richemond and Brendan Maginnis. On Wasserstein reinforcement learning and the Fokker-Planck equation. *arXiv preprint arXiv:1712.07185*, 2017.

[67] Hannes. Risken. *The Fokker-Planck Equation: Methods of Solution and Applications (Springer Series in Synergetics)*. Springer,, 1996.

[68] Christian P Robert and George Casella. The Metropolis—Hastings algorithm. In *Monte Carlo Statistical Methods*, pages 231–283. Springer, 1999.

[69] Filippo Santambrogio. Gradient flows in Wasserstein spaces and applications to crowd movement. *Séminaire Équations aux dérivées partielles (Polytechnique)*, pages 1–16, 2010.

[70] Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55(58-63):94, 2015.

[71] Filippo Santambrogio. Euclidean, Metric, and Wasserstein gradient flows: an overview, 2016.

[72] Simo Sarkka. On unscented kalman filtering for state estimation of continuous-time nonlinear systems. *IEEE Transactions on automatic control*, 52(9):1631–1641, 2007.

[73] Kazimierz Sobczyk. *Stochastic differential equations: with applications to physics and engineering*, volume 40. Springer Science & Business Media, 2013.

[74] Tobias Sutter, Arnab Ganguly, and Heinz Koeppl. A variational approach to path estimation and parameter inference of hidden diffusion processes. *The Journal of Machine Learning Research*, 17(1):6544–6580, 2016.

[75] Gabriel Terejanu, Puneet Singla, Tarunraj Singh, and Peter D Scott. A novel gaussian sum filter method for accurate solution to the nonlinear filtering problem. In *2008 11th International Conference on Information Fusion*, pages 1–8. IEEE, 2008.

[76] Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of tr(f(a)) via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

[77] P Vatiwutipong and N Phewchean. Alternative way to derive the distribution of the multivariate ornstein–uhlenbeck process. *Advances in Difference Equations*, 2019(1):1–7, 2019.

[78] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[79] Michail D Vrettas, Manfred Opper, and Dan Cornford. Variational mean-field algorithm for efficient inference in large systems of stochastic differential equations. *Physical Review E*, 91(1):012148, 2015.

[80] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[81] Shu Yang and B. Wayne Bequette. Optimization-based control using input convex neural networks. *Computers and Chemical Engineering*, 144:107143, 2021.

[82] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[83] Ruiyi Zhang, Changyou Chen, Chunyuan Li, and Lawrence Carin. Policy optimization as Wasserstein gradient flows. In *International Conference on Machine Learning*, pages 5737–5746. PMLR, 2018.

I Nonlinear Filtering Algorithm

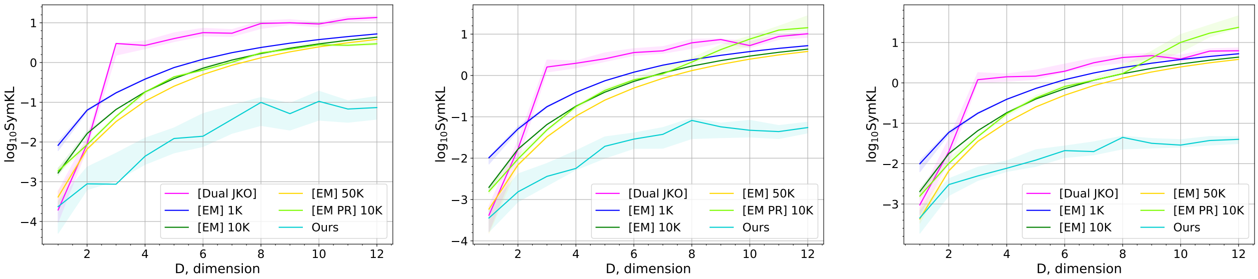
The Algorithm 3 represents a modified version of 1 with embedded MCMC sampling technique based on Metropolis-Hastings algorithm. Note, that in order to decorrelate the MCMC chain we introduce decorrelation parameter K_d and take each K_d-th sample. Also we warm-up the chain omitting the first N_w samples. See §3.4 for reference.

Algorithm 3: Nonlinear filtering with ICNN JKO

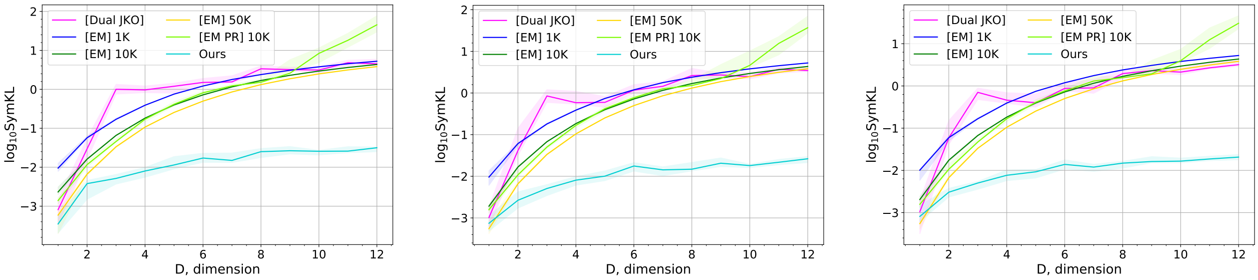
```
Input  : Initial diffusion distribution $p_{0,X}$, observations $Y_{1:K}$, observation times $t_{1:K}$, diffusion final
         time $t_f$, JKO discretization step $h$, target potential $\Phi(x)$, diffusion process temperature $\beta^{-1}$,
         ICNN model $\psi_\theta$, batch size $N$, MCMC decorrelation parameter $K_d$, MCMC warm-up
         parameter $N_w$
Output : $[B_1, B_2, \dots, B_{K+1}]$ : sequence of ICNN blocks representing gradient flow between
         observation time moments $[0, t_1), [t_1, t_2), \dots [t_{K-1}, t_K)$ and $[t_K, t_f)$
$t_c \leftarrow 0$ ; $t_{K+1} \leftarrow t_f$;
Sample initial batch $X \sim p_{0,X}$;
$i_{\text{model}} \leftarrow 1$ ; $k \leftarrow 1$ ; $n_{\text{omit}} \leftarrow N_w$;
while $t_c < t_f$ do
    for $i = 1, 2, \dots$ do
        if $t_c < t_1$ then
            Sample batch $X \sim p_{0,X}$
        else
            for $i_{batch} = 1, 2, \dots, n_{omit}$ do
                for $i_{samp} = 1, 2, \dots, N$ do
                    $x_{k-1} \leftarrow X[i_{\text{samp}}]$;
                    Sample $y_0 \sim \rho_{\mathcal{N}(0,1)}$;
                    $y_{k-1} \leftarrow B_{k-1} \circ \dots \circ B_1(y_0)$;        /* store $y_1, \dots y_{k-2}$ with $y_{k-1}$ */
                    $\alpha \leftarrow \min\left(1, \frac{\rho_{\mathcal{N}(0,1)}(x_0) p_{0,X}(y_0)}{\rho_{\mathcal{N}(0,1)}(y_0) p_{0,X}(x_0)} \prod_{\tau=1}^{k-1} \frac{p_{\mathcal{N}(Y_\tau,1)}(y_\tau)}{p_{\mathcal{N}(Y_\tau,1)}(x_\tau)}\right)$;
                    With probability $\alpha$ set $X[i_{\text{samp}}] \leftarrow y_{k-1}$;
            $n_{\text{omit}} \leftarrow K_d$
        $\widehat{\mathcal{W}_2^2} \leftarrow \frac{1}{N} \sum_{x \in X} \|\nabla \psi_\theta(x) - x\|_2^2$;
        $\widehat{\mathcal{U}} \leftarrow \frac{1}{N} \sum_{x \in X} \Phi\big(\nabla \psi_\theta(x)\big)$;
        $\widehat{\Delta\mathcal{E}} \leftarrow -\frac{1}{N} \sum_{x \in X} \log\det \nabla^2 \psi_\theta(x)$;
        $\mathcal{L} \leftarrow \frac{1}{2h} \widehat{\mathcal{W}_2^2} + \widehat{\mathcal{U}} + \beta^{-1} \widehat{\Delta\mathcal{E}}$ ; Perform a gradient step over $\theta$ by using $\frac{\partial \mathcal{L}}{\partial \theta}$;
    $\psi_{i_{\text{model}}}^{(k)} \leftarrow \psi_\phi$;
    $i_{\text{model}} \leftarrow i_{\text{model}} + 1$;
    $t_c \leftarrow t_c + h$;
    if $t_c = t_k$ then
        $B_k \leftarrow \nabla \psi_{i_{\text{model}}}^{(k)} \circ \dots \circ \nabla \psi_1^{(k)}$;
        $i_{\text{model}} \leftarrow 1$ ; $n_{\text{omit}} \leftarrow N_w$ ; $k \leftarrow k + 1$;
```

II Additional Experiments

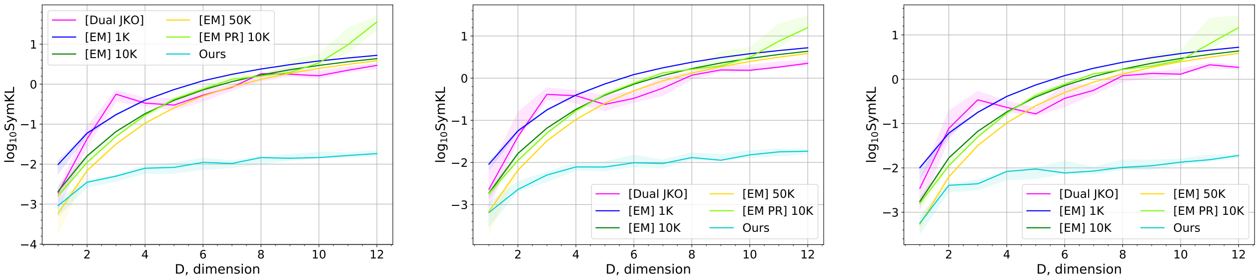
In Figure 6, we compare the true distribution ρ_t with the predicted distribution via the competitive methods when modelling Ornstein-Uhlenbeck processes (§4.3). The comparison is given for time $t = 0.1, 0.2, \ldots, 1.0$.



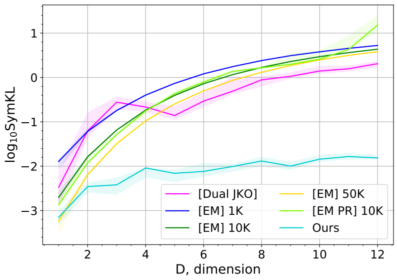
(a) Time $t = 0.1$ (b) Time $t = 0.2$ (c) Time $t = 0.3$



(d) Time $t = 0.4$ (e) Time $t = 0.5$ (f) Time $t = 0.6$



(g) Time $t = 0.7$ (h) Time $t = 0.8$ (i) Time $t = 0.9$



(j) Time $t = 1.0$

Figure 6: SymKL values between the computed measures and the true measure at $t = 0.1, 0.2, \ldots, 1$ in dimensions $D = 1, 2, \ldots 12$. Best viewed in color.