Journal of Computational Physics 561 (2026) 114983

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A bidirectional DeepParticle method for efficiently solving low-dimensional transport map problems

Tan Zhang [a], Zhongjian Wang [b], Jack Xin [c], Zhiwen Zhang [a,d,*]

[a] Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong SAR, China
[b] Division of Mathematical Sciences, Nanyang Technological University, 21 Nanyang Link, 637371, Singapore
[c] Department of Mathematics, University of California, Irvine, CA, 92697, USA
[d] Materials Innovation Institute for Life Sciences and Energy (MILES), HKU-SIRI, Shenzhen, 518045, P.R. China

ARTICLE INFO

Keywords:
Particle method
Optimal transport
Bidirectional mappings
Deep neural networks
Keller–Segel system
One-step generation

ABSTRACT

This paper aims to efficiently compute transport maps between probability distributions arising from particle-based representations of bio-physical problems. We develop a Bidirectional DeepParticle (BDP) method to learn and generate solutions under varying physical parameters, where solutions are approximated as empirical measures of particles that adaptively concentrate in high-gradient regions. The core idea of the BDP method is to learn both forward and reverse maps (between a uniform reference distribution and a non-trivial target distribution) by minimizing the discrete 2-Wasserstein (W2) distance and optimizing the transition map using a mini-batch optimization technique. We present numerical results to demonstrate the effectiveness of the BDP method for learning and generating solutions to the Keller–Segel chemotaxis systems in the presence of laminar flows and Kolmogorov flows with chaotic streamlines in three-dimensional (3D) space. Compared to recent representative single-step flow matching and generative models (rectified flow and shortcut diffusion models), the BDP method achieves superior accuracy with compact neural networks. We also find that for high-dimensional target distributions (4D and above, e.g., Gaussian mixtures), single-step diffusion models exhibit better scalability than the BDP method in terms of W2 accuracy.

1. Introduction

The evaluation of discrepancies between probability distributions represents a fundamental challenge in machine learning. For instance, generative models such as generative adversarial networks (GANs) and variational autoencoders (VAEs) [1–3] seek to transform data points into latent codes that conform to a basic (Gaussian) distribution, enabling the generation and manipulation of data. Representation learning is based on the premise that if a sufficiently smooth function can map a structured data distribution to a simple distribution, it is likely to carry meaningful semantic interpretations, which are advantageous for various downstream learning tasks. Conversely, domain transfer methods identify mappings to shift points between two distinct, empirically observed data distributions, targeting tasks such as image-to-image translation, style transfer, and domain adaptation [4–8]. These tasks can be formulated as finding a transport map between the two distributions:

* Corresponding author.
E-mail addresses: thta@connect.hku.hk (T. Zhang), zhongjian.wang@ntu.edu.sg (Z. Wang), jxin@math.uci.edu (J. Xin), zhangzw@hku.hk (Z. Zhang).

https://doi.org/10.1016/j.jcp.2026.114983
Received 17 April 2025; Received in revised form 1 April 2026; Accepted 27 April 2026
Available online 29 April 2026

Given two empirical samples/observations X_0 and X_1 with $X_0 \sim \pi_0$ and $X_1 \sim \pi_1$ on the metric space X and Y, find a transport map $f : X \to Y$, such that $f(X_0) \sim \pi_1$ when $X_0 \sim \pi_0$.

In recent years, flow models utilizing neural ordinary differential equations (ODEs) and score-based diffusion models with stochastic differential equations (SDEs) [9–11] have been employed to solve transport problems between distributions. A numerical ODE/SDE solver is trained and used to simulate the inference process. These models typically introduce a virtual time t into the transformation between the two distributions, discretizing this virtual time to facilitate training and generation. This discrete process requires multiple calls to the neural network output, increasing the time needed for inference. Although these methods produce high-quality samples, they necessitate an iterative inference procedure, often involving dozens to hundreds of forward passes through the neural network, resulting in slow and costly generation. To expedite the sampling process, it is essential to reduce the number of discrete steps required, allowing the entire sampling to be completed in just a few or even a single step. In doing so, these models sacrifice some performance of the original multi-step diffusion methods in favor of greater generation efficiency. Moreover, if the generation process is limited to a single step, these models can also be adapted to solve the optimal transport (OT) problem[12–15].

The primary challenge lies in identifying suitable metrics that possess both good statistical and optimization properties for finding transport maps. The Wasserstein distance, based on OT, has been employed for this purpose in various machine learning problems [7,16–18]. A particularly notable property of the Wasserstein distance is its applicability between distributions that do not share the same support, which is often the case when working with empirical distributions. In our previous works [19,20], we developed the DeepParticle (DP) method to learn and generate solutions based on particle-based representation. For Keller–Segel (KS) chemotaxis systems [21] that depend on physical parameters (e.g., flow amplitude in the advection-dominated regime and evolution time), DP minimizes the 2-Wasserstein distance between the source and target distributions. Unlike the multi-step iterative process in diffusion models, DP requires only a single call to a neural network to approach the target distribution. Essentially, this constitutes a one-step process to map an initial distribution π_0 to a target distribution π_1 at given physical parameters.

In this work, we further develop an efficient deep learning approach, the Bidirectional DeepParticle (BDP) method, to learn and solve the physically parameterized transport map problems. On top of the uni-directional map in DP [19,20], BDP will make the network learn both the forward mapping $f : X \to Y$ and the reverse mapping $g : Y \to X$ simultaneously to improve the performance and stability. Since the use of a costly transition matrix is unavoidable when calculating the 2-Wasserstein distance, we carry out a mini-batch technique during the training process [22,23]. We will introduce this technique and analyze its error in approximating the 2-Wasserstein distance. Additionally, we compare the BDP performance with two representative single-step diffusion (flow-matching) models in generating target distributions from Keller-Segel chemotaxis data, and a mixture of Gaussians. In lower dimensions, such as 2 and 3 space dimensions in physics, or when data volume is not too large, we observed that DP models can achieve better accuracies than single-step diffusion and flow-matching models while remaining efficient. However, the accuracies of DP models (in Wasserstein distance) decrease with increasing dimension beyond 3, while the single-step diffusion and flow-matching models are much less sensitive. This critical phenomenon may generalize to other data sets and is worth further study. For example, whether the critical dimension is almost universal or problem-dependent.

The rest of the paper is organized as follows. In Section 2, we present our DeepParticle method with the bidirectional idea to learn the forward and reverse transport maps between π_0 and π_1. In connection with OT, we briefly review the framework and algorithm of two single-step models, Rectified flow [12] and Shortcut model [13], for the subsequent comparison study. In Section 3, we show numerical results to demonstrate the performance of our method and compare it with the single-step models. Finally, concluding remarks and future work are in Section 4.

2. Bidirectional deep particle method

In this section, we would like to introduce our BDP method and its corresponding network architecture to learn the features of the transport map between two distributions π_0 and π_1. Compared with recent diffusion models (e.g., DDPM [24], DDIM [25]), the sampling process in Deep Particle methods can be viewed as being completed in only a single step. The mapping error of the Deep Particle methods is measured based on the 2-Wasserstein distance (W2).

2.1. 2-Wasserstein distance

Given distributions π_0 and π_1 defined in the metric space X and Y, we attempt to find a transport map $f_*^0 : X \to Y$ such that $f_*^0(\pi_0) = \pi_1$, where $*$ denotes the push-forward of the transport map. For any function $f_* : X \to Y$, the 2-Wasserstein distance between $f_*(\pi_0)$ and π_1 can be defined by:

$$W_2(f_*(\pi_0), \pi_1) := \left(\inf_{\gamma \in \Gamma(\pi_0, \pi_1)} \int_{X \times Y} c_1(f_*(x), y)^2 d\gamma(x, y) \right)^{\frac{1}{2}}, \tag{1}$$

where $\Gamma(\pi_0, \pi_1)$ denotes the collection of all measures on $X \times Y$ with marginals $f_*(\pi_0)$ and π_1 on the first and second factors and c_1 denotes the metric (distance) on Y. Similarly, for any function $g_* : Y \to X$, we could define

$$W_2(g_*(\pi_1), \pi_0) := \left(\inf_{\gamma \in \Gamma(\pi_1, \pi_0)} \int_{Y \times X} c_0(g_*(y), x)^2 d\gamma(y, x) \right)^{\frac{1}{2}}, \tag{2}$$

where $\Gamma(\pi_1, \pi_0)$ denotes the collection of all measures on $Y \times X$ with marginals $g_*(\pi_1)$ and π_0 on the first and second factors and c_0 denotes the metric (distance) on X.

In practical implementation, the distribution π_0 and π_1 is approximated by the empirical distribution functions with the particles N (samples), i.e. $\pi_0 = \frac{1}{N}\sum_{i=1}^{N}\delta_{x_i}, \pi_1 = \frac{1}{N}\sum_{i=1}^{N}\delta_{y_i}$. It is known that any joint distribution in $\Gamma(\pi_0, \pi_1)$ can be approximated by a $N \times N$ stochastic transition matrix [26], $\gamma = (\gamma_{ij})_{i,j}$,which satisfies

$$\forall i, j,\ \gamma_{ij} \geq 0;\ \forall i,\ \sum_{j=1}^{N}\gamma_{i,j} = 1;\ \forall j,\ \sum_{i=1}^{N}\gamma_{i,j} = 1. \tag{3}$$

Then we can obtain the discretization of the 2-Wasserstein distance (1):

$$\hat{W}_2(f) := \Big(\inf_{\gamma\in\Gamma^N}\frac{1}{N}\sum_{i,j}^{N}c_1(f(x_i), y_j)^2\gamma_{i,j}\Big)^{\frac{1}{2}}. \tag{4}$$

2.2. Methodology and network architecture

Given the training datasets $\{x_i\}_{i=1}^{M} \subset \mathbb{R}^d$ and $\{y_j\}_{j=1}^{M} \subset \mathbb{R}^d$, we have derived (4) to be minimized using gradient descent. However, directly inputting all training data samples incurs significant memory costs for the transition matrix $\gamma \in \mathbb{R}^{M\times M}$. To mitigate this issue, we employ the mini-batch technique commonly used in deep learning literature. Specifically, we select $N < M$ sub-samples from the data computed by the interacting particle method in each iteration of the training process, and we resample these sub-samples every 1000 iterations.

In solving problems such as KS systems [21] that involve real physical parameters, we expect the network to effectively represent these parameters and learn how changes in them affect the target distribution. In this context, more than one set of training data ($\{x_i\}_{i=1}^{N}$ and $\{y_j\}_{j=1}^{N}$ consists of one set of data) should be assimilated. We denote the total number of distinct groups of physical parameters as N_{dict}. This means that the network will have N_{dict} pairs of i.i.d. samples of input and output distribution, denoted by $\{x_{i,r}\}_{i=1}^{N}$ and $\{y_{j,r}\}_{j=1}^{N}$ for $r = 1\cdots N_{dict}$. Correspondingly, we express these different physical parameters in terms of $\{\sigma_r\}_{r=1}^{N_{dict}}$ and let them be the inputs along with each set of $\{x_{i,r}\}_{i=1}^{N}$. This just means the input for the forward network is $\{(x_{i,r}, \sigma_r)\}_{i=1}^{N}$.

In addition, we expect this network to learn both the mapping from π_0 to π_1 and the mapping from π_1 to π_0, ensuring that these two mappings are consistent with each other. Consequently, there are effectively two sub-networks within the overall network architecture, and we include an error term in the loss function to verify this consistency. To be specific, for the network $f_*^\theta : X \to Y$ and $g_*^\vartheta : Y \to X$, the loss function can be represented by:

$$Loss = \hat{W}_2(f_*^\theta) + \hat{W}_2(g_*^\vartheta) + \lambda \cdot MSE(\boldsymbol{x}, g_*^\vartheta \circ f_*^\theta(\boldsymbol{x};\sigma)),\ with \tag{5}$$

$$\hat{W}_2(f_*^\theta) = \frac{1}{N\cdot N_{dict}}\sum_{r=1}^{N_{dict}}\Big(\inf_{\gamma_r\in\Gamma^N}\sum_{i,j=1}^{N}|f_*^\theta(x_{i,r};\sigma_r) - y_{j,r}|^2\gamma_{ij,r}\Big), \tag{6}$$

$$\hat{W}_2(g_*^\vartheta) = \frac{1}{N\cdot N_{dict}}\sum_{r=1}^{N_{dict}}\Big(\inf_{\varphi_r\in\Gamma^N}\sum_{i,j=1}^{N}|g_*^\vartheta(y_{i,r};\sigma_r) - x_{j,r}|^2\varphi_{ij,r}\Big), \tag{7}$$

$$MSE(\boldsymbol{x}, g_*^\vartheta \circ f_*^\theta(\boldsymbol{x};\sigma)) = \frac{1}{N\cdot N_{dict}}\sum_{r=1}^{N_{dict}}\sum_{i=1}^{N}||x_{i,r} - g_*^\vartheta(f_*^\theta(x_{i,r};\sigma_r);\sigma_r)||^2, \tag{8}$$

where θ and ϑ denote the parameters of two sub-networks respectively, $MSE(\cdot,\cdot)$ represents the mean square error between two groups of data, and $\lambda \in \mathbb{R}$ denotes its coefficient.

Compared to the network architecture with only one forward mapping f_*^θ, introducing the network g_*^ϑ can enhance the stability of the learned mapping (transition matrix). For example, we consider there exist two learned forward mapping networks $f_*^{\theta_1}$ and $f_*^{\theta_2}$, such that the difference between them is $f_*^{\theta_1}(x_1;\sigma) = f_*^{\theta_2}(x_2;\sigma) = y_1, f_*^{\theta_1}(x_2;\sigma) = f_*^{\theta_2}(x_1;\sigma) = y_2$. This situation may arise because we use the mini-batch technique during the training process. Since the mapping between the two points is just swapped, the two networks will get the same result when calculating the 2-Wasserstein distance.

For the model with two sub-networks, f_*^θ and g_*^ϑ, since the training of f_*^θ and g_*^ϑ is independent and performed simultaneously, the two different forward networks $f_*^{\theta_1}$ and $f_*^{\theta_2}$ will have different performance in the *MSE* loss term under a fixed g_*^ϑ. For example, if the network g_*^ϑ has $g_*^\vartheta(y_1;\sigma) = x_3,\ g_*^\vartheta(y_2;\sigma) = x_4$, and $x_3 \neq x_4$, then it will occur the case that

$$g_*^\vartheta(f_*^{\theta_1}(x_1;\sigma);\sigma) = g_*^\vartheta(y_1;\sigma) = x_3, \tag{9}$$

$$g_*^\vartheta(f_*^{\theta_2}(x_1;\sigma);\sigma) = g_*^\vartheta(y_2;\sigma) = x_4, \tag{10}$$

and this will lead to different *MSE* loss terms by following (8). At this point, the network will tend to retain the one with the smaller error, thus avoiding the instability of the learned mapping caused by this.

For the architecture of the two sub-networks, we utilize fully connected networks (Multilayer Perceptrons). Each sub-network consists of three latent layers, with each layer having a width of 40 units. The activation function used is $tanh(\cdot)$. Then the relationship between two adjacent layers l_i and l_{i+1} can be represented by:

$$l_{i+1} = tanh(W_i l_i + b_i), \tag{11}$$

where W_i is the weight matrix and b_i is the bias vector of layer l_i. We perform similar operations for the output layer, but at this time, we do not use the activation function.

2.3. Mini-batch technique and relation to OT

In this subsection, we aim to explain the rationale behind using the mini-batch technique during the training process, as well as to analyze the errors induced by this approach. Directly computing the 2-Wasserstein distance between empirical probability distributions with M points has a complexity of $O(M^3 \log M)$ [7], indicating that it is impractical for large data scenarios. To reduce this complexity, a promising technique is to regularize the Wasserstein distance with an entropic term. This enables the use of the efficient Sinkhorn-Knopp algorithm, which can be implemented in parallel and has a lower computational complexity of $O(M^2)$ [27]. However, this complexity is still prohibitive for many large-scale applications.

To train a neural network on large-scale datasets using the 2-Wasserstein distance, several works have proposed leveraging a mini-batch computation of OT distances and back-propagating the resulting gradient into the network. If we divide the data into k batches, each with a batch size of N, this strategy results in a complexity of $O(kN^2)$, enabling the network to handle large datasets. However, the trade-off is that averaging several OT quantities between mini-batches introduces a deviation from the original OT problem. In the context of the OT problem with the entire dataset, the corresponding 2-Wasserstein distance can be described by

$$\hat{W}_2(f(\boldsymbol{x}^{(M)}), \boldsymbol{y}^{(M)}) := \Big(\inf_{\gamma \in \Gamma^M} \frac{1}{M} \sum_{i,j}^{M} c_1(f(x_i), y_j)^2 \gamma_{i,j} \Big)^{\frac{1}{2}}, \tag{12}$$

where $f(\boldsymbol{x}^{(M)}) := \{f(x_i)\}_{i=1}^M \subset \mathbb{R}^d$, and $\boldsymbol{y}^{(M)} := \{y_i\}_{i=1}^M \subset \mathbb{R}^d$. And in practice, we separate $\boldsymbol{x}^{(M)}$ into k batches, $\boldsymbol{x}_1^{(N)}, \cdots, \boldsymbol{x}_k^{(N)}$ and each batch has N data samples ($M = kN$). Similarly, we separate $\boldsymbol{y}^{(M)}$ and obtain $\boldsymbol{y}_1^{(N)}, \cdots, \boldsymbol{y}_k^{(N)}$. The averaged empirical optimal transport distance of the data with batch size N can be represented by

$$\hat{W}_2^{(k)}(f) := \frac{1}{k} \sum_{h=1}^{k} \hat{W}_2(f(\boldsymbol{x}_h^{(N)}), \boldsymbol{y}_h^{(N)}). \tag{13}$$

There have been some previous theoretical results about the general non-asymptotic guarantees for the quality of the approximation $\hat{W}_2^{(k)}(f)$ in terms of the expected L_1 error and L_2 error. Recall that, for $\alpha > 0$, the covering number $\mathcal{N}(X, \alpha)$ of X is defined as the minimal number of closed balls with radius α and centers in X that is needed to cover X.

Proposition 1 (Theorem 2 in [23]). *Let $\hat{W}_2^{(k)}(f)$ be as in (13) for any choice of $k \in \mathbb{N}^+$, then for any integer $q \geq 2$ and $l_{\max} \in \mathbb{N}$:*

$$\mathbb{E}[|\hat{W}_2^{(k)}(f) - W_2(f(\pi_0), \pi_1)|] \leq 2\Psi_q^{\frac{1}{2}} N^{-\frac{1}{4}}, \tag{14}$$

where $\Psi_q := 8q^4 (diam(X))^2 (q^{-2(l_{\max}+1)} \sqrt{M} + \sum_{l=0}^{l_{\max}} q^{-2l} \sqrt{\mathcal{N}(X, q^{-l} diam(X))})$.

It can be observed that the expected L_1 error has a decay rate $O(N^{-\frac{1}{2p}})$ with respect to the batch size. And it has been shown that, in the Euclidean case, the optimal value for q is $q = 2$. The mean square error also has an upper bound.

Proposition 2 (Theorem 5 in [23]). *Let $\hat{W}_2^{(k)}(f)$ be as in (13) for any choice of $k \in \mathbb{N}^+$. Then for any integer $q \geq 2$, the mean squared error of the empirical optimal transport distance can be bounded as ($M = kN$)*

$$\mathbb{E}[|\hat{W}_2^{(k)}(f) - W_2(f(\pi_0), \pi_1)|^2] \leq 18\Psi_q N^{-\frac{1}{2}} = O(N^{-\frac{1}{2}}). \tag{15}$$

Theorem 1. *Let P denote the loss function used during the training process of the BDP method (as mentioned in Algorithm 2 line 14), then P converges to the loss function P_0, which is implemented without the mini-batch technique, on average in L_2 as the batch size N increases and has a convergence rate $O(N^{-\frac{1}{2}})$, i.e.*

$$\mathbb{E}[|\frac{1}{k} \sum_{h=1}^{k} P_h - P_0|^2] = O(N^{-\frac{1}{2}}), \ where \tag{16}$$

$$P_0 := \sum_{r=1}^{N_{dict}} \Big[\sum_{i,j}^{M} \big(|f_\theta(x_{i,r}; \sigma_r) - y_{j,r}|^2 \gamma_{ij,r}^f + |g_\vartheta(y_{i,r}; \sigma_r) - x_{j,r}|^2 \gamma_{ij,r}^g \big) + \frac{\lambda}{M} \sum_{i=1}^{M} |g_\vartheta(f_\theta(x_{i,r}; \sigma_r); \sigma_r) - x_{i,r}|^2 \Big]. \tag{17}$$

Proof.

$$\frac{1}{k}\sum_{h=1}^{k} P_h = \sum_{r=1}^{N_{dict}} \Big[\hat{W}_2^{(k)}(f) + \hat{W}_2^{(k)}(g) + \frac{\lambda}{kN}\sum_{h=1}^{k}\sum_{i=1}^{N} |g_\vartheta(f_\theta(x_{i,r}^{(h)}, \sigma_r), \sigma_r) - x_{i,r}^{(h)}|^2\Big]$$
$$= \sum_{r=1}^{N_{dict}} \Big[\hat{W}_2^{(k)}(f) + \hat{W}_2^{(k)}(g) + \frac{\lambda}{M}\sum_{i=1}^{M} |g_\vartheta(f_\theta(x_{i,r}, \sigma_r), \sigma_r) - x_{i,r}|^2\Big]. \tag{18}$$

$$P_0 = \sum_{r=1}^{N_{dict}} \Big[\hat{W}_2(f(\boldsymbol{x}^{(M)}), \boldsymbol{y}^{(M)}) + \hat{W}_2(g(\boldsymbol{y}^{(M)}), \boldsymbol{x}^{(M)}) + \frac{\lambda}{M}\sum_{i=1}^{M} |g_\vartheta(f_\theta(x_{i,r}; \sigma_r); \sigma_r) - x_{i,r}|^2\Big]. \tag{19}$$

Then, we obtain that

$$\mathbb{E}[|\frac{1}{k}\sum_{h=1}^{k} P_h - P_0|^2] = \mathbb{E}[|\sum_{r=1}^{N_{dict}} [\hat{W}_2^{(k)}(f) + \hat{W}_2^{(k)}(g) - \hat{W}_2(f(\boldsymbol{x}^{(M)}), \boldsymbol{y}^{(M)}) - \hat{W}_2(g(\boldsymbol{y}^{(M)}), \boldsymbol{x}^{(M)})]|^2]$$
$$\le 2\mathbb{E}[\sum_{r=1}^{N_{dict}} [|\hat{W}_2^{(k)}(f) - \hat{W}_2(f(\boldsymbol{x}^{(M)}), \boldsymbol{y}^{(M)})|^2 + |\hat{W}_2^{(k)}(g) - \hat{W}_2(g(\boldsymbol{y}^{(M)}), \boldsymbol{x}^{(M)})|^2]]$$
$$= 2\sum_{r=1}^{N_{dict}} [\mathbb{E}[|\hat{W}_2^{(k)}(f) - \hat{W}_2(f(\boldsymbol{x}^{(M)}), \boldsymbol{y}^{(M)})|^2] + \mathbb{E}[|\hat{W}_2^{(k)}(g) - \hat{W}_2(g(\boldsymbol{y}^{(M)}), \boldsymbol{x}^{(M)})|^2]]$$
$$\le 4\sum_{r=1}^{N_{dict}} \Big[\mathbb{E}[|\hat{W}_2^{(k)}(f) - W_2(f(\pi_0), \pi_1)|^2] + \mathbb{E}[|W_2(f(\pi_0), \pi_1) - \hat{W}_2(f(\boldsymbol{x}^{(M)}), \boldsymbol{y}^{(M)})|^2]$$
$$+ \mathbb{E}[|\hat{W}_2^{(k)}(g) - W_2(g(\pi_1), \pi_0)|^2] + \mathbb{E}[|W_2(g(\pi_1), \pi_0) - \hat{W}_2(g(\boldsymbol{y}^{(M)}), \boldsymbol{x}^{(M)})\Big]$$
$$\le \sum_{r=1}^{N_{dict}} 72(\Psi_{f,r} + \Psi_{g,r})(N^{-\frac{1}{2}} + (kN)^{-\frac{1}{2}}) \le \sum_{r=1}^{N_{dict}} 144(\Psi_{f,r} + \Psi_{g,r})N^{-\frac{1}{2}} = O(N^{-\frac{1}{2}}), \tag{20}$$

where $\Psi_{f,r}$ and $\Psi_{g,r}$ are constants and determined by Proposition 2 Algorithm 1. □

Algorithm 1 Mini-batch technique for a statistical approximation of $W_2(f(\pi_0), \pi_1)$.

1: **Input:** Probability measures π_0, π_1, batch-size N and number of batch k.
2: **for** $i = 1 \cdots k$ **do**
3: Sample i.i.d $\{x_i\}_{i=1}^N \sim \pi_0, \{y_j\}_{j=1}^N \sim \pi_1$.
4: Compute $\hat{W}_2^{(i)} \leftarrow \hat{W}_2(f(\boldsymbol{x}^{(N)}), \boldsymbol{y}^{(N)})$
5: **end for**
6: **Return:** $\hat{W}_2^{(k)}(f) \leftarrow \frac{1}{k}\sum_{h=1}^{k} \hat{W}_2^{(h)}$.

The mini-batch approximation of the Wasserstein gradient is indeed biased, due to the nonlinear structure of the Wasserstein distance. However, we show that in low-dimensional settings, increasing the batch size guarantees convergence of the approximation; see also [28] for the approximation rate of i.i.d samples to their continuous counterpart. Since the dimension considered in this paper is low, increasing the batch size can in a certain sense be regarded as approaching the full-batch regime rather than being a typical mini-batch. The focus of this paper is to develop an accurate method for low-dimensional problems, rather than pursuing a solution that performs reasonably well in high-dimensional scenarios.

2.4. Brief introduction to related approaches

2.4.1. Generative diffusion models

The generative diffusion model [11,24] is a probabilistic model that gradually reconstructs the target data distribution π_1 from a simple prior distribution π_0 through a sequence of iterative denoising steps. Iterative denoising is favored both theoretically, as it provides a stable and tractable way to transform high-entropy noise into structured data, and computationally, because it can be optimized efficiently via score matching and reverse-time simulation. Consequently, the progressive denoising process establishes a flexible framework connecting the noise distribution and the target data distribution

$$\text{forward process:} \quad dX_t = F(X_t, t)dt + G(t)dW_t, \tag{21}$$

$$\text{reverse process:} \quad dX_t = [F(X_t, t) - G(t)^2\nabla_{X_t}\log p_t(X_t)]dt + G(t)dW_t, \tag{22}$$

where $\nabla_{X_t}\log p_t(X_t)$ denotes the score function of the noisy distribution at time t and W_t is a standard Wiener process. Standard generative diffusion models typically require hundreds to thousands of iterative denoising steps during inference, which are extremely time-consuming. Therefore, single-step generative methods are designed to avoid multi-step iterative refinement and directly map a latent code or prior distribution to the target data distribution within a single model evaluation.

Algorithm 2 The BDP method.

Input: Randomly initialize weight parameters f_θ and g_ϑ in the network. Probability measures π_0, π_1, batch-size N , number of physical parameters groups N_{dict} and number of batch k.
for $h = 1 \cdots k$ **do**
  for $r = 1 \cdots N_{dict}$ **do**
    Sample i.i.d $\{x_{i,r}\}_{i=1}^N \sim \pi_0, \{y_{j,r}\}_{j=1}^N \sim \pi_1$ w.r.t. physical parameter σ_r.
    Set $\gamma_{ij,r}^f, \gamma_{ij,r}^g \leftarrow \delta_{ij}$, i.e. initialize permutation matrices;
  end for
  if not the first training mini-batch **then**
    for $r = 1 \cdots N_{dict}$ **do**
      Solve the Earth Movers distance problem between $f_\theta(\boldsymbol{x}_r)$ and $\boldsymbol{y}_r$, and update the permutation matrix $\boldsymbol{\gamma}_r^f$ i.e. $\boldsymbol{\gamma}^f \leftarrow ot.emd(\boldsymbol{a}, \boldsymbol{b}, ot.dist(f_\theta(\boldsymbol{x}_r, \sigma_r)), \boldsymbol{y}_r)$, where $\boldsymbol{a} = \boldsymbol{b} = (\frac{1}{N}, \cdots, \frac{1}{N})^T \in \mathbb{R}^N$ and using the OT package
      Solve the Earth Movers distance problem between $g_\vartheta(\boldsymbol{y}_r, \sigma_r)$ and $\boldsymbol{x}_r$, and update the permutation matrix $\boldsymbol{\gamma}_r^g$
    end for
  end if
  repeat
    Compute the loss $P = \sum_{r=1}^{N_{dict}} \sum_{i,j}^N \left(|f_\theta(x_{i,r}, \sigma_r) - y_{j,r}|^2 \gamma_{ij,r}^f + |g_\vartheta(y_{i,r}, \sigma_r) - x_{j,r}|^2 \gamma_{ij,r}^g \right) + \frac{\lambda}{N} \sum_{i=1}^N |g_\vartheta(f_\theta(x_{i,r}, \sigma_r), \sigma_r) - x_{i,r}|^2$
    $\theta \leftarrow \theta - \delta_1 \nabla_\theta P$,where δ_1 is the learning rate
    $\vartheta \leftarrow \vartheta - \delta_2 \nabla_\vartheta P$, where δ_2 is the learning rate
    Repeat the process (8)-(11) and update the permutation matrices
  until given steps for each training mini-batch;
end for
Return: Save the trained model.

2.4.2. Rectified flow

The rectified flow [12] is an ODE model that transports distribution π_0 to π_1 by following straight line paths as much as possible. Straight paths are preferred both theoretically, as they represent the shortest distance between two endpoints, and computationally, because they can be simulated exactly without the need for time discretization. Consequently, flows along straight paths bridge the gap between one-step and continuous-time models

$$dZ_t = v(Z_t, t)dt. \tag{23}$$

The drift v drives the flow to follow the direction $(X_1 - X_0)$ of the linear path pointing from X_0 to X_1 as much as possible, by solving a simple least squares regression problem:

$$\min_v \int_0^1 \mathbb{E}[||(X_1 - X_0) - v(X_t, t)||^2]dt \quad with\ X_t = tX_1 + (1-t)X_0. \tag{24}$$

The main algorithm of rectified flow can be divided into the following steps:

- Input: Draw (X_0, X_1) from π_0 and π_1 and initialize v^θ with parameter θ.
- Training: Solve the least squares regression problem (24) with $t \sim U(0, 1)$ and get $v^{\hat{\theta}}$.
- Sampling: Draw (Z_0, Z_1) following $dZ_t = v^{\hat{\theta}}(Z_t, t)dt$ starting from $Z_0 \sim \pi_0$.
- Reflow: Start from $(Z_0, Z_1) = (X_0, X_1)$ and repeat the previous three steps.

2.4.3. Shortcut diffusion model

Shortcut diffusion model [13] is a family of generative models that use a single network and training phase to produce high-quality samples in single or multiple sampling steps. Unlike distillation or consistency models, shortcut models are trained in a single training run without a schedule. It defines X_t as a linear interpolation between a data point $X_1 \sim \pi_1$ and a noise point $X_0 \sim N(\mathbf{0}, \mathbb{I})$. The velocity field v_t is the direction from the noise to the data point, i.e.

$$X_t = (1-t)X_0 + tX_1,\ and\ v_t = X_1 - X_0. \tag{25}$$

Given only X_0 renders v_t a random variable because there are multiple plausible pairs (X_0, X_1) and different values that the velocity can take on. The shortcut diffusion model would like to train a single model that supports different sampling budgets, by conditioning the model not only on the timestep t but also on a desired step size d. Conditioning on d allows shortcut models to account for the future curvature, and jump to the correct next point rather than going off track. They refer to the normalized direction from X_t towards the correct next point X'_{t+d} as the shortcut $s(X_t, t, d)$, i.e. $X'_{t+d} = X_t + s(X_t, t, d)$.

By leveraging an inherent self-consistency property of shortcut models, namely that one shortcut step equals two consecutive shortcut steps of half the size, i.e.

$$s(X_t, t, 2d) = s(X_t, t, d)/2 + s(X'_{t+d}, t, d)/2. \tag{26}$$

In practice, they approximate $s(X_t, t, d)$ by $s^\theta(X_t, t, d)$ which is learned by the neural network. During the training process, they split the batch into a fraction that is trained with $d = 0$ and another fraction with randomly sampled $d > 0$ targets and arrive the combined loss function as

$$\mathcal{L}^S(\theta) = \mathbb{E}_{(t,d)\sim p(t,d)}[||s^\theta(X_t, t, 0) - (X_1 - X_0)||^2 + ||s^\theta(X_t, t, 2d) - s_{target}||^2], \tag{27}$$

where $s_{target} = s^\theta(X_t, t, d)/2 + s^\theta(X'_{t+d}, t, d)/2$. In the practical implementation, d takes discrete values and has a minimum unit $\frac{1}{M}$, and the main algorithm can be divided into the following steps:

- Input: Initialize $X_0 \sim N(\mathbf{0}, \mathbb{I})$, $X_1 \sim \pi_1$, and $(t, d) \sim p(t, d)$.
- Training: Learn and obtain θ by minimize the loss function (27) (For the first k batch, set $d = 0, s_{target} = X_1 - X_0$).
- Sampling: Set $X \sim N(\mathbf{0}, \mathbb{I})$, $d = \frac{1}{M}$, $t = 0$. For $n \in [0, \cdots, M-1]$, compute $X \leftarrow X + s^\theta(X, t + nd, d)$.

3. Numerical experiments

In this section, we present several numerical examples comparing the performance of our method with several other one-step models.

3.1. KS simulation and generation in the presence of 3D Laminar flow

The KS model, a partial differential equation system describing chemotaxis-driven aggregation in Dictyostelium discoideum [21]. A common form of the KS model can be represented by:

$$\rho_t = \nabla \cdot (\mu \nabla \rho - \chi \rho \nabla c), \quad \epsilon c_t = \Delta c - k^2 c + \rho, \tag{28}$$

where ρ is the density of the bacteria, c is the concentration of the chemo-attractant, and μ, χ, ϵ, k are non-negative constants. The coupled processes in (28) capture the motion of bacteria that diffuse with mobility μ and drift in the direction of ∇c with velocity $\chi \nabla c$.

When parameters $(\epsilon, k) = \mathbf{0}$, the equation related to the concentration c in (28) is reduced to a simple Poisson equation $\rho = -\Delta c$. When proper boundary conditions are imposed on c, the classical solution assumes the convolution form $c = -\mathcal{K} * \rho$ with $\mathcal{K}$ representing the Green's function of the Laplacian. By substituting this solution into the equation related to the density ρ in (28), we can obtain that

$$\rho_t = \mu \Delta \rho + \chi \nabla \cdot (\rho \nabla (\mathcal{K} * \underline{\rho})). \tag{29}$$

The original KS system is reduced to a non-local non-linear advection-diffusion PDE. To capture chemotactic transport phenomena in fluid dynamic settings such as marine hydrodynamics, previous investigations have focused on the modified transport equation

$$\mathcal{L}_{\boldsymbol{v}} \rho = \mu \Delta \rho + \chi \nabla \cdot (\rho \nabla (\mathcal{K} * \underline{\rho})), \tag{30}$$

where $\mathcal{L}_{\boldsymbol{v}} := \partial_t \rho + \nabla \cdot (\boldsymbol{v} \rho)$ represents the advective Lie derivative accounting for fluid velocity field $\boldsymbol{v}$. Here we consider the KS model with advection term, and $\boldsymbol{v}$ is a divergence-free velocity field. Under the influence of the environmental velocity field $\boldsymbol{v}$, the movement of the organism and the blow-up behavior of the model are also affected and deserve to be investigated. Eq. (30) can be approximated by the interacting particle system below:

$$d\boldsymbol{X}_j = -\frac{\chi M}{J} \nabla_{\boldsymbol{X}_j} \sum_{i=1, i\neq j}^{J} \mathcal{K}(|\boldsymbol{X}_i - \boldsymbol{X}_j|)\, dt + \boldsymbol{v}(\boldsymbol{X}_j)\, dt + \sqrt{2\mu}\, d\boldsymbol{W}_j, \ j = 1, \cdots, J, \tag{31}$$

where M is the conserved total mass and $\{\boldsymbol{W}_j\}_{j=1}^J$ are independent d-dimensional Brownian motions. As $J \to \infty$, (30) becomes the macroscopic limit (McKean–Vlasov equation) of (31).

In practice, numerical instability emerges in the chemo-attractant component $\sum_{i=1, i\neq j}^{J} \mathcal{K}(|\boldsymbol{X}_i - \boldsymbol{X}_j|)$ as particles concentrate, due to unbounded kernel contributions at small interparticle distances. To avoid this situation, we replace the original kernel $\mathcal{K}(\cdot)$ with a smoothed approximation $\mathcal{K}_\delta(\cdot)$, such that $\mathcal{K}_\delta(z) \to \mathcal{K}(z)$ as $\delta \to 0$, and δ here is a regularization parameter. For example, we can define $\mathcal{K}_\delta(z) = \mathcal{K}(z) \cdot \frac{|z|^2}{|z|^2 + \delta^2}$ and obtain the following regularized SDE system:

$$d\boldsymbol{X}_j = -\frac{\chi M}{J} \nabla_{\boldsymbol{X}_j} \sum_{i=1, i\neq j}^{J} \mathcal{K}_\delta(|\boldsymbol{X}_i - \boldsymbol{X}_j|)\, dt + \boldsymbol{v}(\boldsymbol{X}_j)\, dt + \sqrt{2\mu}\, d\boldsymbol{W}_j, \ j = 1, \cdots, J. \tag{32}$$

A well-studied KS model is a 2-dimensional system described by (29) with $\mu = \chi = 1$. The total mass of this system satisfies the conservation law, i.e.

$$\frac{d}{dt} \int_{\mathbb{R}^2} \rho(\boldsymbol{x}, t) d\boldsymbol{x} = 0. \tag{33}$$

This is also true for the system described by (30) if the velocity field $\boldsymbol{v}$ is divergence-free. If we denote $M := \int_{\mathbb{R}^2} \rho(\boldsymbol{x}, t) d\boldsymbol{x}$, then the second moment will have a fixed derivative with respect to time t, i.e.

$$\frac{d}{dt} \int_{\mathbb{R}^2} |x|^2 \rho(x, t) dx = \frac{M}{2\pi}(8\pi - M), \tag{34}$$

Table 1
BDP performance in 3D Laminar flow as λ varies (best in bold).

Sigma	W_2 distance					
	$\lambda = 0$	$\lambda = 10^{-5}$	$\lambda = 10^{-4}$	$\lambda = 10^{-3}$	$\lambda = 10^{-2}$	$\lambda = 10^{-1}$
$\sigma = 10^{(\circ)}$	0.0043	0.0034	0.0028	**0.0027**	0.0057	0.0096
$\sigma = 30^{(\blacktriangle)}$	0.0046	0.0061	0.0049	**0.0025**	0.0031	0.0109
$\sigma = 50^{(\blacktriangle)}$	0.0062	0.0085	0.0069	**0.0061**	0.0063	0.0115
$\sigma = 80^{(\blacktriangle)}$	0.0127	0.0129	0.0151	**0.0112**	0.0147	0.0171
$\sigma = 100^{(\blacktriangle)}$	0.0059	0.0135	0.0101	0.0079	**0.0051**	0.0074
$\sigma = 120^{(\blacktriangle)}$	0.0078	0.0194	0.0114	0.0088	**0.0062**	0.0102
$\sigma = 150^{(\circ)}$	0.0136	0.0270	0.0151	0.0171	**0.0049**	0.0223
$\sigma = 200^{(\bullet)}$	**0.2734**	0.4093	0.3513	0.3788	0.4366	0.5534

Table 2
Model comparison in 3D Laminar flow (best in bold).

Sigma \ Model	W_2 distance (network parameter size $3k$)				
	DM	DM	DM	DP	DP
	Rectified flow	Shortcut		Bi-direction	
$\sigma = 10^{(\circ)}$	0.0137	0.0202	0.0184	0.0044	**0.0027**
$\sigma = 30^{(\blacktriangle)}$	0.0158	0.0250	0.0187	0.0046	**0.0025**
$\sigma = 50^{(\blacktriangle)}$	0.0208	0.0218	0.0232	0.0064	**0.0061**
$\sigma = 80^{(\blacktriangle)}$	0.0174	0.0285	0.0217	0.0127	**0.0112**
$\sigma = 100^{(\blacktriangle)}$	0.0204	0.0284	0.0235	0.0137	**0.0051**
$\sigma = 120^{(\blacktriangle)}$	0.0275	0.0312	0.0390	0.0078	**0.0062**
$\sigma = 150^{(\circ)}$	0.0398	0.0522	0.0455	0.0249	**0.0049**
$\sigma = 200^{(\bullet)}$	0.2859	0.3121	0.3429	**0.2580**	0.2734

where 8π is called the critical mass of the KS system.

It is well-established that for smooth initial data, the system will develop a finite-time blow-up when the initial mass $M > 8\pi$. It has been proved that if the total mass is smaller than the critical mass, the system (29) and the system (30) will have a global smooth solution with smooth initial data. In situations involving supercritical mass, numerical experiments indicate that if advection is sufficiently large, it can prevent the solutions from blowing up.

We now consider three space dimensions (3D) and a divergence-free velocity field to be a laminar flow, i.e.

$$\boldsymbol{v}(x, y, z) = \sigma \cdot \Big(\exp(-y^2 - z^2),\ 0,\ 0 \Big)^T, \tag{35}$$

which describes a flow of amplitude σ traveling along the x-direction with speed depending on the radial position in the yz-plane.

The training data is generated by the regularized interacting particle method (32). We first let the network learn the dependence of the aggregation patterns on the amplitude of the advection field σ while fixing the evolution time $T = 0.02$. The physical parameter σ of the data samples sent for training is isometrically distributed between $\sigma = 10$ and $\sigma = 150$. Since we are interested in the performance of small-sized models so that they can be implemented very efficiently, we set the network parameter size to $3,000$ in subsequent experiments. Before comparing the performance of various models, we investigate the dependence of the new BDP model on λ, the coefficient of the identical verification error term.

In Table 1 and subsequent tables, we use different superscript symbols to indicate different types of parameters. Here, $\circ, \blacktriangle, \bullet$ represent the data samples that are used (generated) in *training*, *interpolation*, and *extrapolation*, respectively, under the corresponding physical parameters. From Table 1, we see that for the majority of cases, as σ increases, the value of λ corresponding to the minimum *2-Wasserstein distance* also increases, and the corresponding model achieves optimal performance. Therefore, in the following experiments, we test the performance of the BDP model with $\lambda \approx 10^{-4}\sigma$ as a guideline for choosing λ.

Table 2 shows the 2-Wasserstein distance between different models and the reference distribution generated by the interacting particle method. In this table, we use DM to denote diffusion/flow matching model, and DP to denote our deep particle method. The same parameter size $(3,000)$ was used for all methods. In the third line, Rectified flow and Shortcut mean two models with single-step generation techniques, which have been mentioned above. Bi-direction refers to the BDP method, which learns both forward and reverse mappings.

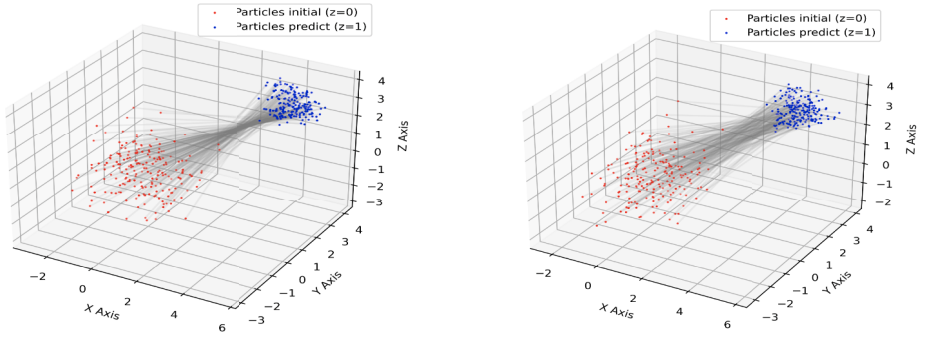
From Table 3, we observe that as the mini-batch size increases, the W2 error of the model's predictions gradually decreases. In our subsequent numerical experiments, we selected $N = 2000$. Increasing the batch size further would reduce computational efficiency, and the gains in accuracy would no longer be significant. From Table 4, we can see that the two Deep Particle methods perform better than other single step models. The two Deep Particle methods both perform well and exhibit similar performance. The inference times of all the models are listed in the following table. The differences among one-step methods (one call to the network during inference) are minor (i.e., of the same order), mainly due to programming languages for implementation: (DM, Rectified flow, and shortcut DM) in Pytorch, and (DP, BDP) in JAX.

Table 3
BDP performance in 3D Laminar flow as mini-batch size N varies ($\sigma = 120$).

Mini-batch size	$N = 500$	$N = 1000$	$N = 1500$	$N = 2000$
W2 error	0.0138	0.0104	0.0088	0.0062

Table 4
Inference times in seconds of DM and one-step models.

Model	Diffusion Model	Rectified flow	Shortcut	DeepParticle	BDP
Inference time	49 s	0.48 s	0.54 s	0.24 s	0.26 s



(a) Deep Particle Mapping (b) Bidirectional Deep Particle Mapping

Fig. 1. Comparison between the DP and BDP mapping.

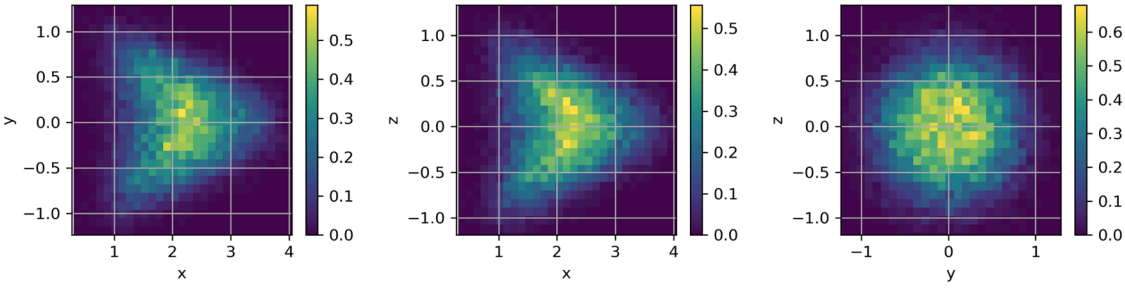


Fig. 2. Cross sectional views of reference in 3D Laminar flow at $\sigma = 150$.

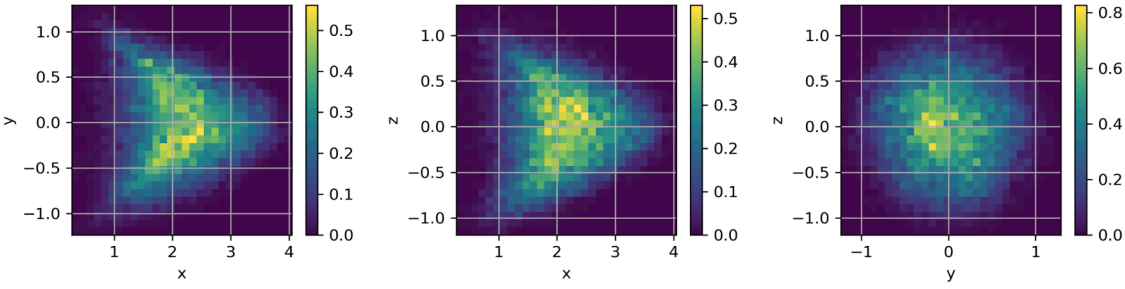


Fig. 3. Cross sectional views of DP method in 3D Laminar flow at $\sigma = 150$.

Comparing Fig. 1a and b, we observe a reduction in the crossover phenomenon between the initial data points and the corresponding prediction after introducing the network g_*^{ϑ} and the self-consistency MSE error term. This implies that, although the overall W2 error may remain unchanged, the BDP model is more inclined to search for the optimal mapping.

Figs. 2–6 compare the particle distributions generated by different models with the reference in the case of $\sigma = 150$ in 3D laminar flow. We can see that the DP model demonstrates superior accuracy in capturing the tail behavior compared to other single-step methods.

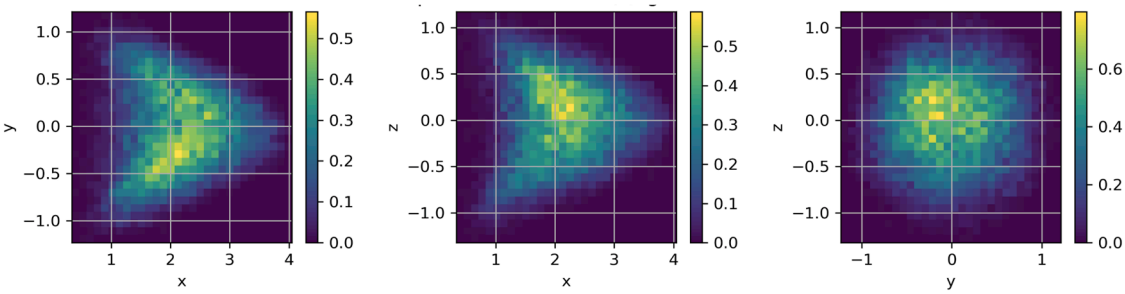


Fig. 4. Cross sectional views of BDP method in 3D Laminar flow at $\sigma = 150$.

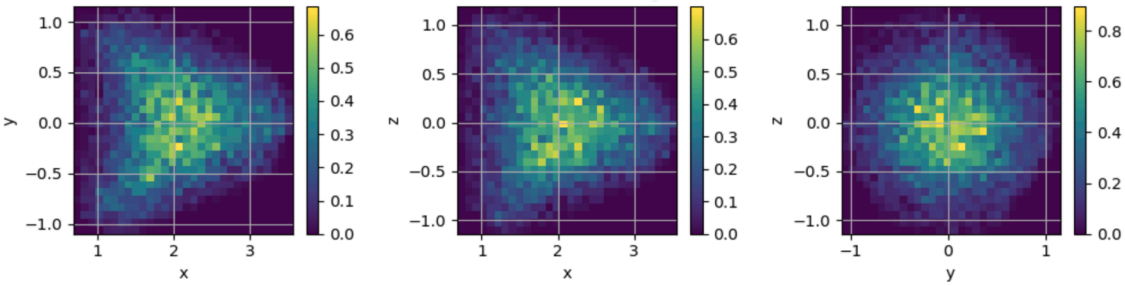


Fig. 5. Cross sectional views of Rectified flow in 3D Laminar flow at $\sigma = 150$.

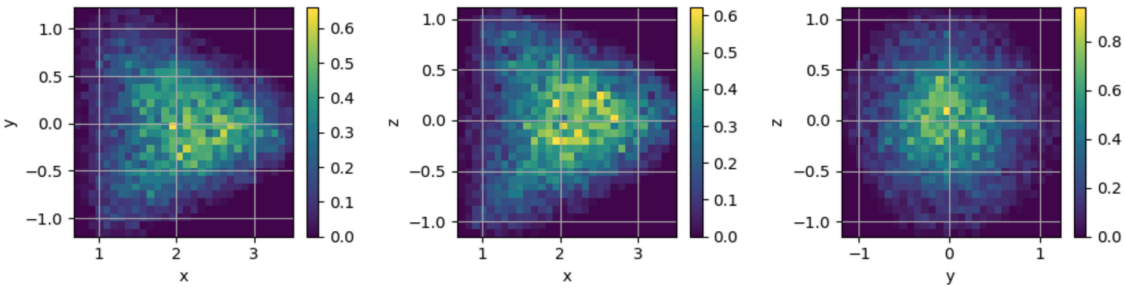


Fig. 6. Cross sectional views of Shortcut model in 3D Laminar flow at $\sigma = 150$.

3.2. KS in 3D Kolmogorov flow

We consider the KS model in 3D Kolmogorov flow to generate chemotaxis patterns while the organisms travel and aggregate in chaotic streamlines. The fluid velocity field is:

$$\boldsymbol{v}(x, y, z) = \sigma \cdot \Big(\sin(2\pi z),\ \sin(2\pi x),\ \sin(2\pi y) \Big)^T. \tag{36}$$

The rest of the experimental setup is the same as in the previous subsection on 3D laminar flow. We similarly compare the performance of different models at small network parameter sizes.

Table 5 shows the performance of different models in comparison. Notations remain the same as in Table 2. It is seen that DP and BDP methods perform much better than the diffusion model (DM) and its recent one-step adaptations. In Fig. 7 to 11, we plot the distribution generated by DP models and single-step DM models. By comparing with the reference generated by the IPM, we observe that the distributions learned by DP models come much closer to the target.

3.3. Mixtures of Gaussian

In this subsection, we compare the performance of BDP and single-step diffusion (flow matching) models in the case of Gaussian mixtures in different space dimensions. For a mixture of m Gaussian distributions on $\mathbb{R}^n$, the target distribution is in closed form. Let the probability density function of a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ be:

$$p(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})). \tag{37}$$

Table 5
Model performance in 3D Kolmogorov flow.

Sigma \ Model	W_2 distance (network parameter size $3k$)				
	DM	DM	DM	DP	DP
	Rectified flow	Shortcut		Bi-direction	
$\sigma = 10^{(\circ)}$	0.0184	0.0219	0.0221	0.0113	**0.0066**
$\sigma = 30^{(\blacktriangle)}$	0.0140	0.0196	0.0182	0.0064	**0.0058**
$\sigma = 50^{(\blacktriangle)}$	0.0297	0.0417	0.0343	**0.0060**	0.0127
$\sigma = 80^{(\blacktriangle)}$	0.0311	0.0471	0.0397	0.0167	**0.0153**
$\sigma = 100^{(\blacktriangle)}$	0.0362	0.0537	0.0404	0.0178	**0.0147**
$\sigma = 120^{(\blacktriangle)}$	0.0402	0.0569	0.0489	0.0241	**0.0237**
$\sigma = 150^{(\circ)}$	0.0543	0.0696	0.0628	0.0494	**0.0318**
$\sigma = 200^{(\bullet)}$	0.9556	1.1124	1.0376	**0.6841**	0.8042

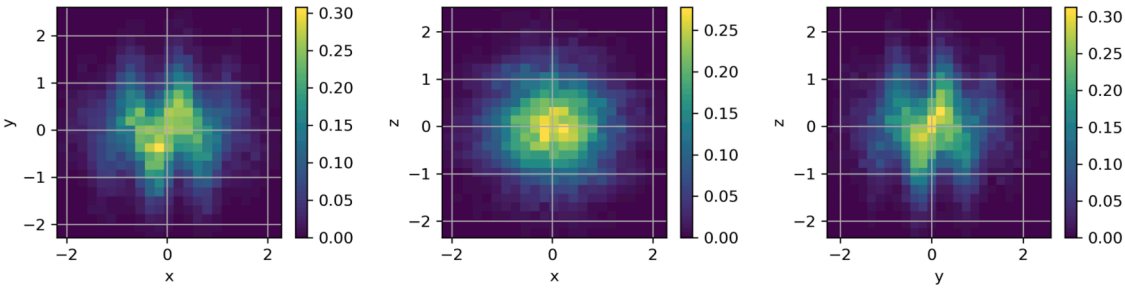


Fig. 7. Cross sectional views of reference in 3D Kolmogorov flow at $\sigma = 100$.

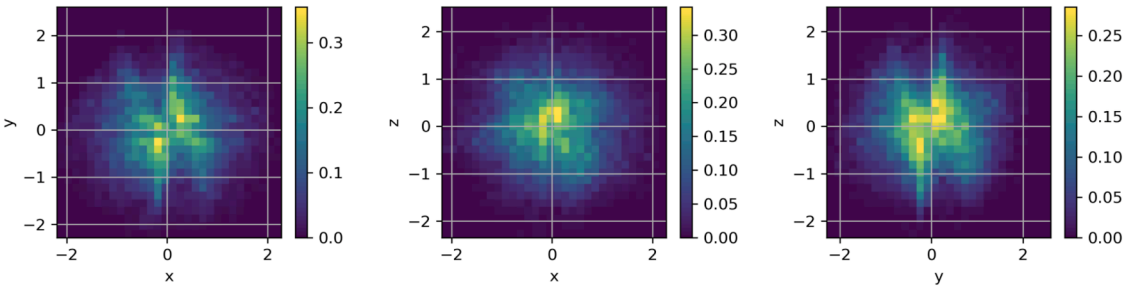


Fig. 8. Cross sectional views of DP method in 3D Kolmogorov flow at $\sigma = 100$.

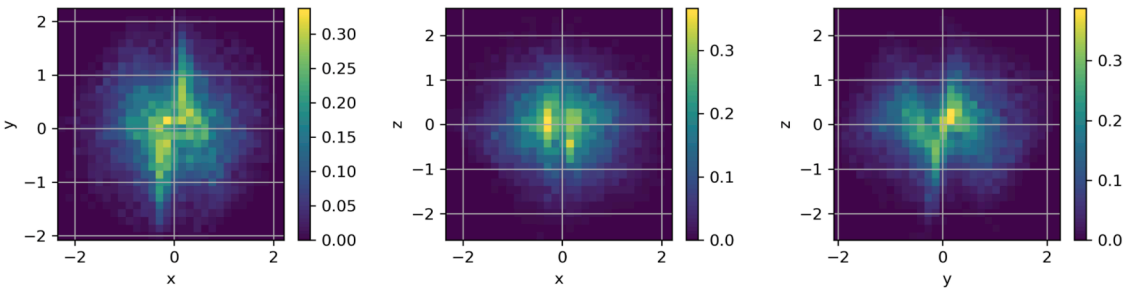


Fig. 9. Cross sectional views of BDP method in 3D Kolmogorov flow at $\sigma = 100$.

Then, the closed form of the Gaussian mixture can be represented by

$$p_{GM}(\boldsymbol{x}; \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^m = \sum_{i=1}^m w_i \cdot p(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \ with \ \sum_{i=1}^m w_i = 1. \tag{38}$$

Here $w_i > 0$ is the weight of the i-th Gaussian distribution. In the numerical experiments, we choose $m = 2$ and set $w_i = 0.5, i = 1, 2$. The following table shows the performance of DP models, diffusion, and two single-step diffusion models vs. dimensions. The *2-Wasserstein* distance is computed between the first two dimensions of the network output and the reference.

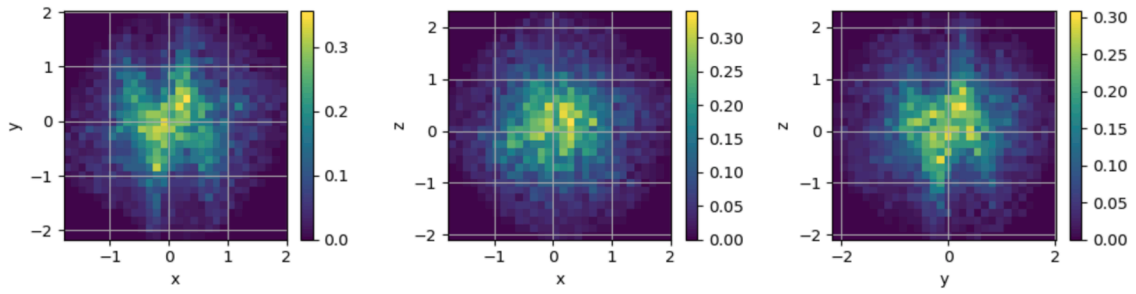


Fig. 10. Cross sectional views of Rectified flow in 3D Kolmogorov flow at $\sigma = 100$.

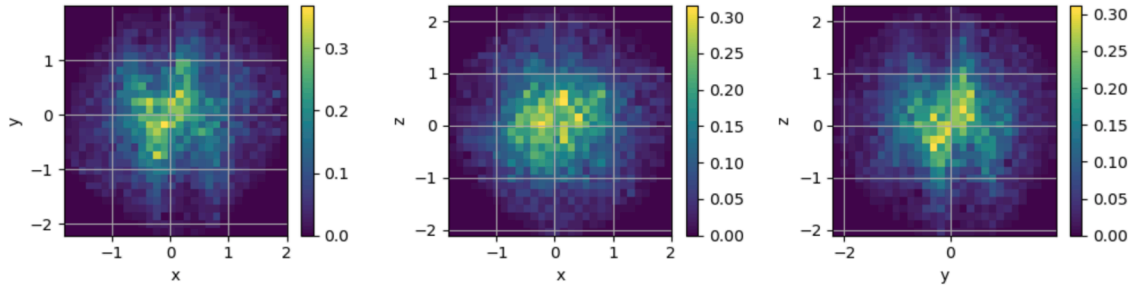


Fig. 11. Cross sectional views of Shortcut model in 3D Kolmogorov flow at $\sigma = 100$.

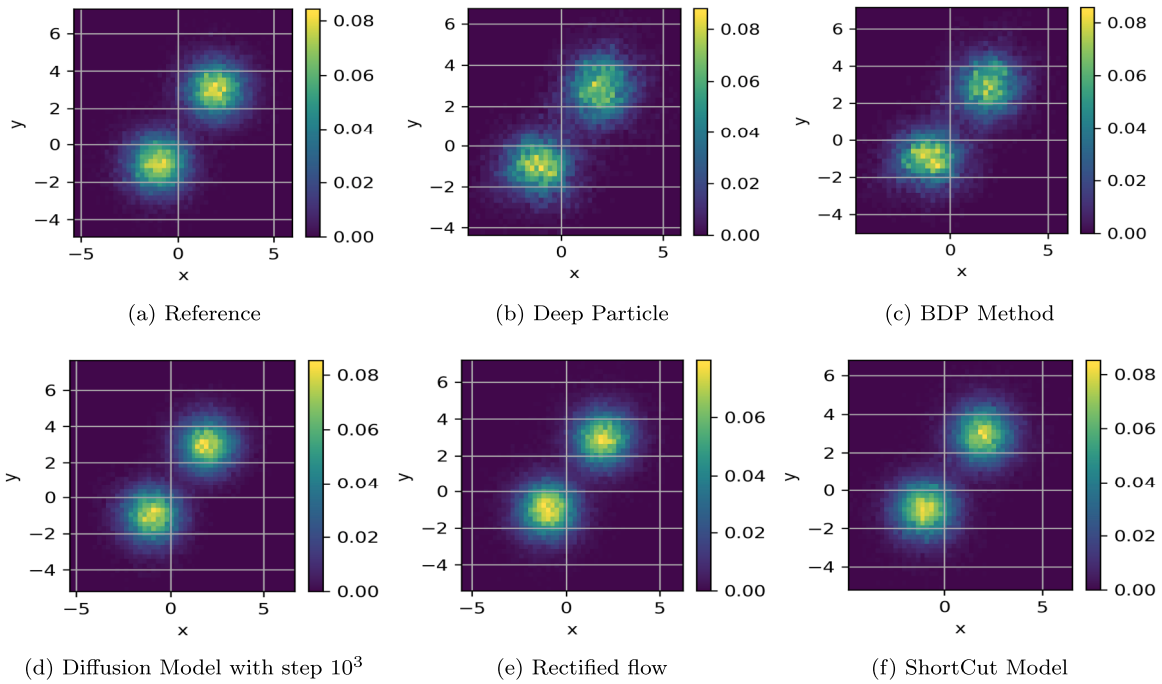


Fig. 12. Generation projected on the x-y plane by models vs. reference, targeting a $16D$ Gaussian mixture.

From Table 6, we observe that the two single-step models maintain performance better than DP as the dimension increases. Fig. 12 also illustrates the performance of each model under the 16D setting, from which we observe that the two single-step models achieve better performance than DP. This is due to the transition matrix computations in DP models which have $O(N^2)$ memory cost, N the mini-batch size. However, DM and flow matching (FM) models only need $O(N)$ memory for the cost related to batch size. As the dimension increases, the DP-based models have more information to learn in each round of learning, but the $O(N^2)$ memory cost growth and the memory limitation of the machine itself will make it challenging to provide more data samples in one epoch. In

Table 6
Model performance in mixture of two Gaussians on $\mathbb{R}^n$, bold/italics is best among all (one-step) methods.

$n \setminus$ Model	W_2 distance (parameter size $3k$)				
	DM	DM	DM	DP	DP
	Rectified flow	Shortcut		Bi-direction	
$n = 3$	0.0633	0.0812	0.0654	0.0635	**0.0534**
$n = 4$	**0.0603**	0.0840	*0.0632*	0.1141	0.1052
$n = 8$	**0.0657**	0.0826	*0.0688*	0.1317	0.1207
$n = 16$	**0.0700**	0.0896	*0.0739*	0.1598	0.1493
$n = 32$	**0.1035**	0.1369	*0.1124*	0.3507	0.2926

comparison, DM-based models are much less sensitive in this regard. In lower dimensions, such as 2-3 space dimensions, or when data volume is not large, the DP models can achieve better performance while remaining efficient, which is consistent with the above two examples of KS models.

4. Conclusion

In this work, we developed the BDP method, a deep learning approach for efficiently solving a class of physically parameterized transport map problems in low-dimensional spaces. The BDP method is based on the simultaneous optimization of forward and reverse mappings via the minimization of the 2-Wasserstein distance between distributions. Our method combines three key innovations: (1) a bidirectional architecture that improves the stability of the learned mapping, (2) the use of the mini-batch optimization technique in the training process with an analysis of the approximation error with respect to mini-batch size, and (3) the single-step generation to accelerate computation.

We present numerical experiments to demonstrate the performance of the BDP method and compare it with two single-step diffusion models, applied to the Keller-Segel system (under 3D laminar and Kolmogorov flows) and Gaussian mixture distributions. We identify a critical dimension phenomenon: while the BDP method achieves the fastest inference speed across all dimensions, its performance is superior in 2D/3D problems with moderate data volumes, whereas the one-step diffusion models scale more effectively in high-dimensional, data-rich regimes. This critical dimension phenomenon has important implications for scientific machine learning.

Our results establish the BDP method as the method of choice for low-dimensional transport problems. In future work, we will further study the critical phenomenon (the transition from BDP to diffusion models) by examining the transition dimension and developing methods to improve the scalability of the BDP method.

CRediT authorship contribution statement

Tan Zhang: Writing – original draft, Software, Methodology, Investigation; **Zhongjian Wang:** Writing – original draft, Methodology, Investigation, Conceptualization; **Jack Xin:** Writing – review & editing, Supervision, Methodology, Conceptualization; **Zhiwen Zhang:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

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.

Acknowledgement

ZZ was partially supported by the National Natural Science Foundation of China (Projects 92470103 and 12171406), the Hong Kong RGC grant (Projects 17304324 and 17300325), the Seed Funding Programme for Basic Research (HKU), and the Hong Kong RGC Research Fellow Scheme 2025. ZW was partly supported by NTU SUG-023162-00001, MOE AcRF Tier 1 Grant RG17/24. JX was partly supported by NSF grants DMS-2309520, DMS-2219904, and DMS-2151235. The computations were performed at computing facilities provided by Information Technology Services, The University of Hong Kong, and the Greenplanet Cluster at UC Irvine.

References

[1] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, MIT Press, 2014, pp. 2672–2680.
[2] D.P. Kingma, M. Welling, Auto-encoding variational bayes, in: International Conference on Learning Representations, 2013.
[3] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using real NVP, in: International Conference on Learning Representations, 2017.
[4] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2223–2232.
[5] N. Courty, R. Flamary, D. Tuia, A. Rakotomamonjy, Optimal transport for domain adaptation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (9) (2016) 1853–1865.
[6] G. Trigila, E.G. Tabak, Data-driven optimal transport, Commun. Pure Appl. Math. 69 (4) (2016) 613–648.
[7] G. Peyré, M. Cuturi, et al., Computational optimal transport: with applications to data science, Found. Trends textregistered Mach. Learn. 11 (5-6) (2019) 355–607.
[8] L. Yang, C. Daskalakis, G.E. Karniadakis, Generative ensemble regression: learning particle dynamics from observations of ensembles with physics-informed deep generative models, SIAM J. Sci. Comput. 44 (1) (2022) B80–B99.
[9] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, Adv. Neural Inf. Process. Syst. 31 (2018).
[10] G. Papamakarios, E. Nalisnick, D.J. Rezende, S. Mohamed, B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference, J. Mach. Learn. Res. 22 (57) (2021) 1–64.
[11] Y. Song, J. Sohl-Dickstein, D.P. Kingma, A. Kumar, S. Ermon, B. Poole, Score-Based generative modeling through stochastic differential equations, in: International Conference on Learning Representations, 2021.
[12] X. Liu, C. Gong, et al., Flow straight and fast: learning to generate and transfer data with rectified flow, in: International Conference on Learning Representations, 2023.
[13] K. Frans, D. Hafner, S. Levine, P. Abbeel, One step diffusion via shortcut models, In Int. Conf. Learn. Represent. (2025), arXiv:2410.12557.
[14] R. Li, X. Ye, H. Zhou, H. Zha, Learning to match via inverse optimal transport, J. Mach. Learn. Res. 20 (80) (2019) 1–37.
[15] W. Gangbo, W. Li, S. Osher, M. Puthawala, Unnormalized optimal transport, J. Comput. Phys. 399 (2019) 108940.
[16] C. Villani, Topics in optimal transportation, volume 58, American Mathematical Soc., 2021.
[17] L. Ambrosio, E. Brué, D. Semola, et al., Lectures on optimal transport, volume 130, Springer, 2021.
[18] A. Figalli, F. Glaudo, An invitation to optimal transport, Wasserstein distances, and gradient flows, 2021.
[19] Z. Wang, J. Xin, Z. Zhang, DeepParticle: learning invariant measure by a deep neural network minimizing Wasserstein distance on data generated by an interacting particle method, J. Comput. Phys. 464 (2022) 111309.
[20] Z. Wang, J. Xin, Z. Zhang, A DeepParticle method for learning and generating aggregation patterns in multi-dimensional Keller-Segel chemotaxis systems, Phys. D Nonlinear Phenom. 460 (2024) 134082.
[21] E.F. Keller, L.A. Segel, Initiation of slime mold aggregation viewed as an instability, J. Theor. Biol. 26 (3) (1970) 399–415.
[22] K. Fatras, Y. Zine, S. Majewski, R. Flamary, R. Gribonval, N. Courty, Minibatch optimal transport distances; analysis and applications, (2021) arXiv:2101.01792.
[23] M. Sommerfeld, J. Schrieber, Y. Zemel, A. Munk, Optimal transport: fast probabilistic approximation with exact solvers, J. Mach. Learn. Res. 20 (105) (2019) 1–23.
[24] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Adv. Neural Inf. Process. Syst. 33 (2020) 6840–6851.
[25] J. Song, C. Meng, S. Ermon, Denoising diffusion implicit models, in: International Conference on Learning Representations, 2021.
[26] R. Sinkhorn, A relationship between arbitrary positive matrices and doubly stochastic matrices, Ann. Math. Stat. 35 (2) (1964) 876–879.
[27] J. Altschuler, J. Niles-Weed, P. Rigollet, Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration, Adv. Neural Inf. Process. Syst. 30 (2017).
[28] N. Fournier, A. Guillin, On the rate of convergence in Wasserstein distance of the empirical measure, Probab. Theory Relat. Fields 162 (3) (2015) 707–738.