Journal of Computational Physics 487 (2023) 112155

Contents lists available at ScienceDirect

Journal of Computational Physics

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

Bridging mean-field games and normalizing flows with trajectory regularization

Han Huang[a], Jiajia Yu[a], Jie Chen[b], Rongjie Lai[a,*]

[a] Rensselaer Polytechnic Institute, 110 8th St, Troy, NY 12180, USA
[b] MIT-IBM Watson AI Lab, IBM Research, 314 Main St, Cambridge, MA 02142, USA

ARTICLE INFO

Article history:
Received 14 January 2023
Received in revised form 12 April 2023
Accepted 14 April 2023
Available online 20 April 2023

Keywords:
Mean-field games
Optimal transport
Optimal control
Normalizing flows

ABSTRACT

Mean-field games (MFGs) are a modeling framework for systems with a large number of interacting agents. They have applications in economics, finance, and game theory. Normalizing flows (NFs) are a family of deep generative models that compute data likelihoods by using an invertible mapping typically parameterized by neural networks. They are useful for density modeling and data generation. While active research has been conducted on both models, few noted the relationship between the two. In this work, we unravel the connections between MFGs and NFs by contextualizing the training of an NF as solving the MFG. This is achieved by reformulating the MFG problem in terms of agent trajectories and parameterizing a discretization of the resulting MFG with flow architectures. With this connection, we explore two research directions. First, we employ expressive NF architectures to accurately solve high-dimensional MFGs, sidestepping the curse of dimensionality in traditional numerical methods. Compared with other deep learning approaches, our trajectory-based formulation encodes the continuity equation in the network architecture to better approximate population dynamics. Second, we regularize the training of NFs with transport costs and show the effectiveness on controlling the model's Lipschitz bound, resulting in better generalization performance. We demonstrate numerical results through comprehensive experiments on a variety of synthetic and real-life datasets.

1. Introduction

Mean-field games (MFGs) are a powerful modeling framework for systems with a large number of interacting agents, which arise naturally in the study of game theory [1,2], economics [3,4], finance [5–7], and industrial planning [8–10]. In general, the MFG formulation prescribes an identical objective for all agents and seeks the optimal strategy over a time interval. Instead of tracking individual strategies, mean-field approaches are used to model distributions of strategic profiles, and the optima of the MFG objective generalizes the Nash equilibria in the corresponding finite N-player game as $N \to \infty$. When the strategies concern spatial movement with a particular transport cost, the MFG is reduced to the optimal transport (OT) problem, which finds rich applications in signal processing and machine learning [11–14].

A key theoretical underpinning for MFGs is the characterization of its optimality condition in terms of the Hamilton-Jacobi-Bellman (HJB) equation. Obtaining the Nash equilibrium of an MFG is thus reduced to solving a system of nonlinear

* Corresponding author.
E-mail addresses: huangh14@rpi.edu (H. Huang), yuj12@rpi.edu (J. Yu), chenjie@us.ibm.com (J. Chen), lair@rpi.edu (R. Lai).

https://doi.org/10.1016/j.jcp.2023.112155
0021-9991/© 2023 Elsevier Inc. All rights reserved.

partial differential equations (PDEs), for which a plethora of numerical methods have been developed based on the Eulerian framework [15–21]. Nevertheless, the reliance on spatial discretization renders traditional approaches exponentially more expensive as the dimensionality grows. Hence, numerically solving MFGs in high dimensions remains a difficult problem due to the curse of dimensionality. Meanwhile, machine learning-based approaches emerged lately. A recent method [22] successfully solves MFGs in quite high dimensions using deep neural networks. This method uses a Lagrangian-based approach to approximate the value function. However, besides penalizing the HJB equation, this method has to additionally solve the continuity equation that governs the evolution of the densities under the learned dynamics.

Rather than approximating the value function in MFGs by using neural networks as discussed in [22], we propose to approximate the trajectories instead. To this end, we establish a mathematical connection between the MFG framework and a popularly used family of generative models in deep learning—normalizing flows (NFs). Composed of a sequence of invertible mappings, NFs allow for the exact computation of the data likelihood. They can be used to model complex data distributions for density estimation and latent sampling [23]. Starting from the MFG problem, we reformulate it in terms of agent trajectories and canonically parameterize the discretization as an NF. As a result, the standard negative log-likelihood training of an NF is equivalent to a special case of the MFG without transport and interaction costs.

This connection between the MFGs and the NFs motivates us to explore two promising directions. First, we approximate the trajectories of MFGs with expressive NF architectures to accurately and efficiently solve high-dimensional MFG problems. The key novelty of our approach lies in its encoding of the continuity equation into the neural network, which mitigates numerical errors incurred in the approximation and produces more accurate solutions of MFGs. We demonstrate the advantage of our approach in problems including dynamic OT, crowd motion, and multi-group path planning. Additionally, we conduct numerical analysis of a discretization scheme and explore the relationship between continuous and discretized MFGs. By characterizing the solution behaviors of discretized MFGs with the OT theory [24], we obtain an important insight, which suggests that direct linear interpolation of the discretized problems in the temporal direction solves MFGs without the interaction term. Moreover, we adapt the universality theory from [25] to argue the effectiveness of using a particular kind of NFs—affine coupling flows—in parameterizing MFG trajectories. The resultant push-forward measure in the NF converges in distribution, corroborating accurate modeling of the population dynamics.

Second, we improve NFs by introducing the transport cost to their training, resulting in *trajectory-regularized flows*. In general, there exists more than one flow that transforms a given probability density to another. Therefore, the NF training problem is ill-posed and proper regularization is needed to incentivize the solution towards a more meaningful one. The OT theory suggests that this can be done via using the kinetic energy as the transport cost, which induces an OT plan that traverses the geodesic in the space of measures with the Wasserstein-2 metric [24]. We show by using a variety of synthetic datasets that in the absence of the transport cost, one cannot expect existing NF models to approximate the OT trajectory. Additionally, we find that the intermediate flows incur significantly less distortion when trained with transport costs. Regulating the kinetic energy, therefore, serves as an effective way of controlling the Lipschitz bound, which is intimately related to the model's generalization performance as well as its ability to counter adversarial samples [26,27]. Traditionally, the control over the Lipschitz bound for a neural network is implemented through explicit regularization such as the l_2 weight decay. To the best of our knowledge, this is the first work regularizing NFs by using transport costs and relating the regularization to the Lipschitz bound of the trained neural network, which proves to be more robust and effective than weight decay. We demonstrate the effectiveness of transport-based regularization on a variety of synthetic and real-life datasets and show improvements over popularly used NF models, such as RealNVP and the neural spline flow [28,29].

2. Related works

Variational MFGs are a generalization of the dynamic formulation of OT [24], where the final density matching is encouraged but not enforced and transport costs other than the kinetic energy can be considered. Traditional methods for solving OT and variational MFG problems are well-studied in low dimensions [15–21,30]; and machine learning-based approaches emerged recently for solving high-dimensional MFGs. The work [22] is the first on solving high-dimensional deterministic MFGs by approximating the value function using deep neural networks. The work [31] solves high-dimensional MFGs in the stochastic setting using the primal-dual formulation of MFGs, where training is conducted similarly to that of generative adversarial networks [32,14].

Our proposed trajectory-regularized flow can be built on any discrete NF. In general, existing NF models impose a certain structure on the flow mapping to enable fast evaluation of the log-determinant of the Jacobian. For a non-exhaustive list, RealNVP and NICE combine affine coupling flows to build simple yet flexible mappings [28,33], while NSF and Flow++ use splines and a mixture of logistic CDFs as more sophisticated coupling functions [29,34]. MAF and IAF are autoregressive flows with affine coupling functions [35,36], while NAF and UMNN parameterize the coupling function with another neural network [37,38].

Not the focus of this work, continuous NF models consider the ODE limit of the discrete version, with significant changes to the computation of the parameter gradient and the change of variables. In these models, neural ODE outlines the theoretical framework [39] and FFJORD uses the Hutchinson estimator to approximate the trace term incurred in the density evaluation [40]. OT-Flow introduces the OT cost to continuous NFs, suggesting a parameterization of the HJB potential that allows for exact trace computations [41]. In addition, recent works found that the transport cost can enhance and stabilize the training of continuous NFs [42]. For example, the OT theory suggests that the optimal agent trajectories are straight;

this characterization allows the ODE solver in OT-Flow to integrate with larger time steps and reduce the computational cost [41].

3. Roadmap

The rest of the paper is organized as follows. We provide the mathematical background for MFGs and NFs in Section 4 and elaborate the connections between the two in Section 5, via a reformulation and discretization of the MFG objective. In Section 6, we perform theoretical analysis to relate the discretized and continuous MFG problems, as well as establishing universality theorems for using NFs to solve MFGs. In Section 7, we provide numerical results that demonstrate the effectiveness and accuracy of solving high-dimensional MFGs by using NFs. Meanwhile, we illustrate the effect of regularized NF training, stress the ability of acquiring a lower Lipschitz bound, and show its superiority to weight decay in real-life datasets.

4. Mathematical background

4.1. Mean-field games

The study of MFGs considers classic N-player games at the limit of $N \to \infty$ [43–45]. Assuming homogeneity of the objectives and anonymity of the players, we construct a MFG with a continuum of non-cooperative rational agents distributed in a spatial domain $\Omega \subseteq \mathbb{R}^d$ and evolving across a time interval $[0, T]$. For any fixed $t \in [0, T]$, we denote the population density of agents by $p(\cdot, t) \in \mathcal{P}(\Omega)$, where $\mathcal{P}(\Omega)$ is the space of all probability densities over Ω. For an agent starting at $\boldsymbol{x}_0 \in \Omega$, their position over time follows a trajectory $\boldsymbol{x} : [0, T] \to \Omega$ governed by

$$\begin{cases} \mathrm{d}\boldsymbol{x}(t) = \boldsymbol{v}(\boldsymbol{x}(t), t)\mathrm{d}t, \quad \forall t \in [0, T] \\ \boldsymbol{x}(0) = \boldsymbol{x}_0, \end{cases} \tag{1}$$

where $\boldsymbol{v} : \Omega \times [0, T] \to \Omega$ specifies an agent's action at a given time. For simplicity, we assume no stochastic terms in (1) in this work. Then, each agent's trajectory is completely determined by $\boldsymbol{v}(\boldsymbol{x}, t)$. To play the game over an interval $[t, T]$, each agent seeks to minimize their objective:

$$J_{\boldsymbol{x}_0,t}(\boldsymbol{v}, p) := \int_t^T [L(\boldsymbol{x}(s), \boldsymbol{v}(\boldsymbol{x}(s), s)) + I(\boldsymbol{x}(s), p(\boldsymbol{x}(s), s))]\mathrm{d}s + M(\boldsymbol{x}(T), p(\boldsymbol{x}(T), T)) \\ \text{s.t. (1) holds.} \tag{2}$$

Each term in the objective denotes a particular type of cost. The running cost $L : \Omega \times \Omega \to \mathbb{R}$ is incurred by each agent's own action. A commonly used example for L is the kinetic energy $L(\boldsymbol{x}, \boldsymbol{v}) = \|\boldsymbol{v}\|^2$, which accounts for the total amount of movement along the trajectory. The running cost $I : \Omega \times \mathcal{P}(\Omega) \to \mathbb{R}$ is accumulated through each agent interacting with one another or with the environment. For example, this term can be an entropy that discourages the agents from grouping together, or a penalty for colliding with an obstacle [22]. The terminal cost $M : \Omega \times \mathcal{P}(\Omega) \to \mathbb{R}$ is computed from the agents' final state, which typically measures a discrepancy between the final density $p(\cdot, T)$ and a desirable density $p_1 \in \mathcal{P}(\Omega)$.

To solve the MFG, we define the value function $u : \Omega \times [0, T] \to \mathbb{R}$ as

$$u(\boldsymbol{x}_0, t) := \inf_{\boldsymbol{v}} J_{\boldsymbol{x}_0,t}(\boldsymbol{v}, p), \qquad \text{s.t. (1) holds.} \tag{3}$$

One can show that $u(\boldsymbol{x}, t)$ and $p(\boldsymbol{x}, t)$ are solutions to the following HJB equation and continuity equations [43–45]

$$\begin{cases} -\partial_t u(\boldsymbol{x}, t) + H(\boldsymbol{x}, \nabla_{\boldsymbol{x}} u(\boldsymbol{x}, t)) = I(\boldsymbol{x}, p(\boldsymbol{x}, t)) \\ \partial_t p(\boldsymbol{x}, t) + \nabla_{\boldsymbol{x}} \cdot (p(\boldsymbol{x}, t)\boldsymbol{v}(\boldsymbol{x}, t)) = 0 \\ p(\boldsymbol{x}, 0) = p_0(\boldsymbol{x}), \quad \text{and} \quad u(\boldsymbol{x}, t) = M(\boldsymbol{x}, p(\boldsymbol{x}, t)), \end{cases} \tag{4}$$

where $H : \Omega \times \Omega \to \mathbb{R}$ is the Hamiltonian $H(\boldsymbol{x}, \boldsymbol{q}) := \sup_{\boldsymbol{v}} -\langle \boldsymbol{q}, \boldsymbol{v}\rangle - L(\boldsymbol{x}, \boldsymbol{v})$; $p_0 \in \mathcal{P}(\Omega)$ is the initial population density at $t = 0$; and $\boldsymbol{v}(\boldsymbol{x}, t)$ denotes the optimal strategy for an agent at position $\boldsymbol{x}$ and time t. It can be shown that the optimal strategy satisfies $\boldsymbol{v}(\boldsymbol{x}, t) = -\nabla_p H(\boldsymbol{x}, \nabla_{\boldsymbol{x}} u(\boldsymbol{x}, t))$. We note that the continuity equations in the last two lines of (4) are a special case of the Fokker-Planck equation with no diffusion terms.

The setup (1)–(2) concerns the strategy of an individual agent. Under suitable assumptions, the work [43] developed a macroscopic formulation of the MFG that models the collective strategy of all agents. Suppose there exist functionals $\mathcal{I}, \mathcal{M} : \mathcal{P}(\Omega) \to \mathbb{R}$ such that

$$I(\boldsymbol{x}, p) = \frac{\delta \mathcal{I}}{\delta p}(\boldsymbol{x}), \quad M(\boldsymbol{x}, p) = \frac{\delta \mathcal{M}}{\delta p}(\boldsymbol{x}),$$

where $\frac{\delta}{\delta p}$ is the variational derivative. Then, the functions $p(\boldsymbol{x},t)$ and $\boldsymbol{v}(\boldsymbol{x},t)$ satisfying (4) coincide with the optimizers of the following variational problem:

$$\begin{aligned}\inf_{p,\boldsymbol{v}} J(p,\boldsymbol{v}) &:= \int_0^T \int_\Omega L(\boldsymbol{x}, \boldsymbol{v}(\boldsymbol{x},t), p(\boldsymbol{x},t)) \mathrm{d}\boldsymbol{x}\mathrm{d}t + \int_0^T \mathcal{I}(p(\cdot,t))\mathrm{d}t + \mathcal{M}(p(\cdot,T)) \\ s.t. \quad & \partial_t p(\boldsymbol{x},t) + \nabla_{\boldsymbol{x}} \cdot (p(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t)) = 0, \quad \boldsymbol{x} \in \Omega, t \in [0,T] \\ & p(\boldsymbol{x},0) = p_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega.\end{aligned} \tag{5}$$

This formulation is termed the *variational MFG* and it serves as the main optimization problem this work solves. It is known that traditional numerical PDE methods that solve (4) are computationally intractable in dimensions higher than three, as the number of grid points grows exponentially. This challenge motivates us to solve the MFG problem with parameterizations of the agent trajectory based on NFs.

4.2. Normalizing flows

NFs are a family of invertible neural networks that find applications in density estimation, data generation, and variational inference [33,28,29,34–37]. In machine learning, NFs are considered deep generative models, which include variational autoencoders [46] and generative adversarial networks [32] as other well-known examples. The key advantage of NFs over these alternatives is the exact computation of the data likelihood, which is made possible by invertibility.

Mathematically, suppose we have a dataset $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^N \subseteq \mathbb{R}^d$ generated by an underlying data distribution P_1; that is, $\boldsymbol{x}_n \overset{\text{iid}}{\sim} P_1$. We define a *flow* to be an invertible function $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$ parameterized by $\theta \in \mathbb{R}^W$, and a *normalizing flow* to be the composition of a sequence of flows: $F_\theta = f_{\theta_K} \circ f_{\theta_{K-1}} \circ ... \circ f_{\theta_2} \circ f_{\theta_1}$ parameterized by $\theta = (\theta_1, \theta_2, ..., \theta_K)$. To model the complex data distribution P_1, the idea is to use an NF to gradually transform a simple base distribution P_0 to P_1. Formally, the transformation of the base distribution is conducted through the push-forward operation $F_{\theta *}P_0$, where $F_{\theta *}P(A) := P(F_\theta^{-1}(A))$ for all measurable sets $A \subset \Omega$. The aim is that the transformed distribution resembles the data distribution. Thus, a commonly used loss function for training the NF is the KL divergence:

$$\min_\theta \quad \mathcal{D}_{KL}(P_1 || F_{\theta *}P_0). \tag{6}$$

If the measure P admits density p with respect to the Lebesgue measure, the density $F_{\theta *}p$ associated with the push-forward measure $F_{\theta *}P$ satisfies the change-of-variable formula [23]:

$$F_{\theta *}p(\boldsymbol{x}) = p(F_\theta^{-1}(\boldsymbol{x}))|\det \nabla F_\theta^{-1}(\boldsymbol{x})|, \tag{7}$$

where ∇F_θ^{-1} is the Jacobian of F_θ^{-1}. This allows one to tractably compute the KL term in (6), which is equivalent to the negative log-likelihood of the data up to a constant (independent of θ):

$$\begin{aligned}\mathcal{D}_{KL}(P_1||F_{\theta *}P_0) &= -\mathbb{E}_{\boldsymbol{x}\sim P_1}[\log F_{\theta *}p_0(\boldsymbol{x})] + \text{const.} \\ &= -\mathbb{E}_{\boldsymbol{x}\sim P_1}\left[\log p_0(g_{\theta_1} \circ ... \circ ...g_{\theta_K}(\boldsymbol{x})) + \sum_{k=1}^K \log|\det \nabla g_{\theta_k}(\boldsymbol{y}_{K-k})|\right] + \text{const.},\end{aligned} \tag{8}$$

where $g_{\theta_i} := f_{\theta_i}^{-1}$ are the flow inverses, and $\boldsymbol{y}_j := g_{\theta_{K-j}} \circ ... \circ g_{\theta_K}(\boldsymbol{x}), \forall j = 1, 2, ..., K-1, \boldsymbol{y}_0 := \boldsymbol{x}$.

We remark that there is flexibility in choosing the base distribution P_0, as long as it admits a known density to evaluate on. In practice, a typical choice is the standard normal $\mathrm{N}(0, I)$. In addition, (8) requires the log-determinant of the Jacobians, which take $O(d^3)$ time to compute in general. Existing NF architectures sidestep this issue by meticulously using functions that follow a (block) triangular structure [23], so that the log-determinants can be computed in $O(d)$ time from the (block) diagonal elements.

One popular family of NF models is the coupling flow [23], which composes individual flows in the following form

$$f(\boldsymbol{x}_1, \boldsymbol{x}_2) = (h(\boldsymbol{x}_1; \theta(\boldsymbol{x}_2)), \boldsymbol{x}_2) := (\boldsymbol{y}_1, \boldsymbol{y}_2), \tag{9}$$

where $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{R}^a \times \mathbb{R}^{d-a}$, $\theta : \mathbb{R}^{d-a} \to \mathbb{R}^d$ is the called the *conditioner*, and $h : \mathbb{R}^d \to \mathbb{R}^d$ is the coupling function. Suppose $h(\cdot; \theta)$ is invertible for the given θ, then the inverse f^{-1} is:

$$(\boldsymbol{x}_1, \boldsymbol{x}_2) = (h^{-1}(\boldsymbol{y}_1; \theta(\boldsymbol{y}_2)), \boldsymbol{y}_2). \tag{10}$$

In common scenarios, h is defined element-wise, such that invertibility is ensured by strict monotonicity. For example, a simple yet expressive choice is the affine function $h(\boldsymbol{x}; \theta) = \theta_1 \boldsymbol{x} + \theta_2, \theta_1 \neq 0, \theta_2 \in \mathbb{R}$. This gives rise to the RealNVP flow [28]. Other popular coupling functions include splines, mixture of CDFs, and functions parameterized by neural networks, which are shown to be more expressive than the affine coupling function [29,47,48,34,49,37].

5. Bridging MFGs and NFs

In this section, we reformulate the MFG problem as a generalization of the NF training loss. This reformulation relates the two models and opens opportunities to improve both. The contribution is two-fold. On the one hand, we solve high-dimensional MFG problems by using an NF parameterization. The NF model encodes the discretized trajectory of the agents in its network architecture, which allows for efficient optimization through evaluation of the log-determinants in $O(d)$ cost. We show that the expressitivity of the NF architecture allows one to tractably solve high-dimensional MFG problems with small error. On the other hand, under the MFG framework we introduce a model-agnostic regularization to improve the training of NFs. In the MFG language, existing NF models are trained by using only the terminal cost (i.e., the KL loss). With the introduction of the transport cost, the intermediate flows become better behaved and more robust, owing to a smaller Lipschitz bound. It turns out that the learned NF improves the matching of the densities, as evident in various synthetic and real-life examples.

5.1. Trajectory-based formulation of MFGs

Rather than solving for the density $p(\boldsymbol{x},t)$ and the action $\boldsymbol{v}(\boldsymbol{x},t)$, we reparameterize the problem (5) to directly work with agent trajectories, from which $p(\boldsymbol{x},t)$ and $\boldsymbol{v}(\boldsymbol{x},t)$ can be derived. Let $P(\cdot,t)$ be the measure that admits $p(\cdot,t)$ as its density for all $t\in[0,T]$. Define the agent trajectory as $F:\mathbb{R}^d\times\mathbb{R}\to\mathbb{R}^d$, where $F(\boldsymbol{x},t)$ is the position of the agent starting at $\boldsymbol{x}$ and having traveled for time t. It satisfies the following ordinary differential equation:

$$\begin{cases}\partial_t F(\boldsymbol{x},t)=\boldsymbol{v}(F(\boldsymbol{x},t),t), & \boldsymbol{x}\in\Omega, t\in[0,T]\\ F(\boldsymbol{x},0)=\boldsymbol{x}, & \boldsymbol{x}\in\Omega.\end{cases} \tag{11}$$

The evolution of the population density is determined by the movement of agents. Thus, $P(\cdot,t)$ is simply the push-forward of P_0 under $F(\cdot,t)$; namely, $P(\boldsymbol{x},t)=F(\cdot,t)_*P_0(\boldsymbol{x})$, whose associated density satisfies

$$p(\boldsymbol{x},t)=\mathrm{d}(F(\cdot,t)_*P_0)(\boldsymbol{x}), \tag{12}$$

where $\mathrm{d}(F(\cdot,t)_*p_0(\boldsymbol{x}))$ is a Radon-Nikodym derivative. Note that this definition of $p(\boldsymbol{x},t)$ automatically satisfies the continuity equation as well as the initial condition for $p(\boldsymbol{x},t)$ in (5). Unless specified otherwise, we assume $T=1$ and use the L_2 transport cost $L(\boldsymbol{x},\boldsymbol{v}(\boldsymbol{x},t),p(\boldsymbol{x},t))=\lambda_L p(\boldsymbol{x},t)\|\boldsymbol{v}(\boldsymbol{x},t)\|_2^2$ from now on. Here, $\lambda_L\geq 0$ is treated as a model (hyper-)parameter. Furthermore, we can apply a change of variables on (12) to rewrite the integral involving $L(\boldsymbol{x},\boldsymbol{v},p)$:

$$\begin{aligned}\int_0^1\int_\Omega L(\boldsymbol{x},\boldsymbol{v}(\boldsymbol{x},t),p(\boldsymbol{x},t))\mathrm{d}\boldsymbol{x}\mathrm{d}t&=\lambda_L\int_0^1\int_\Omega p(\boldsymbol{x},t)\|\boldsymbol{v}(\boldsymbol{x},t)\|_2^2\mathrm{d}\boldsymbol{x}\mathrm{d}t\\&=\lambda_L\int_0^1\int_\Omega\|\boldsymbol{v}(\boldsymbol{x},t)\|_2^2 d(F(\cdot,t)_*P_0)(\boldsymbol{x})\mathrm{d}t\\&=\lambda_L\int_0^1\int_\Omega p_0(\boldsymbol{x})\|\partial_t F(\boldsymbol{x},t)\|_2^2\mathrm{d}\boldsymbol{x}\mathrm{d}t.\end{aligned}$$

Similarly, we can rewrite the interaction cost $\mathcal{I}(p(\cdot,t))$ and the terminal cost $\mathcal{M}(p(\cdot,T))$ as:

$$\begin{aligned}\mathcal{I}(p(\cdot,t))&=\mathcal{I}(F(\cdot,t)_*p_0),\\ \mathcal{M}(p(\cdot,T))&=\mathcal{M}(F(\cdot,T)_*p_0).\end{aligned} \tag{13}$$

Therefore, (5) becomes:

$$\begin{aligned}\inf_F\quad \lambda_L\int_0^1\int_\Omega p_0(\boldsymbol{x})\|\partial_t F(\boldsymbol{x},t)\|_2^2\mathrm{d}\boldsymbol{x}\mathrm{d}t+\int_0^1\mathcal{I}(F(\cdot,t)_*p_0)\mathrm{d}t+\mathcal{M}(F(\cdot,1)_*p_0)&:=\mathcal{L}(F)\\ s.t.\quad F(\boldsymbol{x},0)=\boldsymbol{x}.&\end{aligned} \tag{14}$$

It is worth noting that (14) removes the continuity-equation constraint. The reformulated problem optimizes over agent trajectories F, which allows one to parameterize with an NF. In addition, the remaining constraint $F(\cdot,0)=\mathrm{Id}$ will be automatically integrated into the NF, such that it can be trained in an unconstrained manner.

5.2. Discretization of MFG trajectories with NFs

Starting from the base density p_0, each flow in the NF advances the density one step forward. Hence, the NF model specifies a discretized evolution of the agent trajectories. Formally, we discretize the trajectory $F(\boldsymbol{x}, t)$ in time with regular grid points $t_i := i \cdot \Delta t, \forall i = 0, 1, \ldots K$. Thus, the grid spacing is $\Delta t := \frac{1}{K}$. Denote $F(\boldsymbol{x}, t_i) := F_i(\boldsymbol{x})$. By approximating $\partial_t F(\boldsymbol{x}, t)$ with the forward difference, the transport cost becomes:

$$\begin{aligned}\lambda_L \int_0^1 \int_\Omega p_0(\boldsymbol{x}) \|\partial_t F(\boldsymbol{x}, t)\|_2^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t &= \lambda_L \int_0^1 \mathbb{E}_{z\sim P_0}[\|\partial_t F(\boldsymbol{z}, t)\|_2^2]\mathrm{d}t \\ &\sim \lambda_L K \cdot \mathbb{E}_{z\sim P_0}\left[\sum_{i=0}^{K-1} \|F_{i+1}(\boldsymbol{z}) - F_i(\boldsymbol{z})\|_2^2\right].\end{aligned} \tag{15}$$

Moreover, suppose that the terminal cost is computed as the KL divergence as in standard NFs:

$$\mathcal{M}(F(\cdot, T)_* p_0) = \lambda_{\mathcal{M}} D_{KL}(P_1 || F(\cdot, T)_* P_0) = \lambda_{\mathcal{M}} D_{KL}(P_1 || F_{K_*} P_0). \tag{16}$$

For simplicity of exposition, for now we omit the interaction term (that is, $\mathcal{I} \equiv 0$). We can discretize the interaction cost $\mathcal{I}(F(\cdot, t)_* P_0)$, similarly to (15), given its exact form. Subsequently, we will give numerical examples for the case $\mathcal{I} \not\equiv 0$ (see Sections 7.1.2 and 7.1.3).

For any F, the discretized objective value converges to the continuous version at $O(\frac{1}{K})$. However, we show later that the optimal discretized and continuous objective values in fact agree for any fixed number of grid points used. Hence, there introduce no additional errors by discretizing the original MFG problem, in the absence of interaction costs.

To solve the discretized MFG, we define the flow maps for $0 \le t_i \le t_j \le 1$ as $\Phi_{t_i}^{t_j}(\boldsymbol{x}_0) = \boldsymbol{x}(t_j)$, where

$$\begin{cases} \mathrm{d}\boldsymbol{x}(t) = \boldsymbol{v}(\boldsymbol{x}(t), t)\mathrm{d}t, & t \in [t_i, t_j] \\ \boldsymbol{x}(t_i) = \boldsymbol{x}_0. \end{cases} \tag{17}$$

Here, we leverage the semi-group property $\Phi_{t_b}^{t_c} \circ \Phi_{t_a}^{t_b} = \Phi_{t_a}^{t_c}$ to decompose the agent trajectories into a sequence of flow maps. It follows that $F_k = \Phi_{t_{k-1}}^{t_k} \circ \Phi_{t_{k-2}}^{t_{k-1}} \circ \ldots \circ \Phi_{t_0}^{t_1}, \forall k = 1, 2, \ldots K$, and we parameterize each $\Phi_{t_i-1}^{t_i} : \mathbb{R}^d \to \mathbb{R}^d$ by a flow model f_{θ_i}. Denote $F_{\theta^k} := f_{\theta_k} \circ \ldots \circ f_{\theta_1}$, $F_{\theta^0} := Id$, and $F_\theta := F_{\theta^K}$. The discretized MFG problem with flow-parameterized trajectories is:

$$\inf_\theta \quad \lambda_L K \cdot \mathbb{E}_{\boldsymbol{z}\sim P_0}\left[\sum_{k=0}^{K-1} \|F_{\theta^{k+1}}(\boldsymbol{z}) - F_{\theta^k}(\boldsymbol{z})\|_2^2\right] + \lambda_{\mathcal{M}} \mathcal{D}_{KL}(P_1 || F_{\theta_*} P_0), \tag{18}$$

where $\lambda_{\mathcal{M}}, \lambda_L \ge 0$ are weights.

By comparing (18) to the NF optimization problem (8), we see that training a standard NF is equivalent to solving a special MFG with no transport and interaction costs. In the past, many highly flexible flows were developed that are capable of solving this special MFG [37,29,28,50,34]. As a result, the trajectory-based formulation opens the door to employ expressive NFs to accurately solve MFGs. In addition, note that the density evolution can be directly obtained from the agent trajectories, which are the outputs of the individual flows in the NF model. Thus, this formulation sidesteps the additional effort required to solve the Fokker-Planck equation (4), resulting in a more efficient solution of the population dynamics.

We also note that the computation of (18) involves both a forward (normalizing) and an inverse (generating) pass of the NF: the forward pass computes $\mathcal{D}_{KL}(P_1 || F_{\theta_*} P_0)$ via (8) and the inverse pass yields the transport cost. For NFs that have comparable run time for passes in both directions (e.g. coupling flows), adding the transport cost only doubles the overall computation time. For NFs where the inverse pass is much slower than the forward pass (or vice versa), such as autoregressive flows [23], the additional computational effort associated with the transport cost may be prohibitive. To remedy this, we propose an alternative, equivalent formulation that incurs less overhead.

Alternative formulation Recall that the forward agent trajectory is defined in (11), such that $p(\boldsymbol{x}, t) = F(\cdot, t)_* p_0(\boldsymbol{x})$. Equivalently, we can consider the backward trajectory $G : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$:

$$\begin{aligned} &\partial_t G(\boldsymbol{x}, 1-t) = \boldsymbol{v}(G(\boldsymbol{x}, 1-t), t), \quad \boldsymbol{x} \in \Omega, t \in [0, T] \\ &\qquad G(\boldsymbol{x}, 0) = \boldsymbol{x}, \quad \boldsymbol{x} \in \Omega. \end{aligned} \tag{19}$$

Therefore, $G(\cdot, t)_* P_1 = P(\cdot, 1-t), \forall t \in [0, 1]$. As a result, the objective (18) becomes:

$$\inf_\theta \quad \lambda_L K \cdot \mathbb{E}_{\boldsymbol{x}\sim P_1}\left[\sum_{k=0}^{K-1} \|G_{\theta^k}(\boldsymbol{x}) - G_{\theta^{k+1}}(\boldsymbol{x})\|_2^2\right] + \lambda_{\mathcal{M}} \mathcal{D}_{KL}(P_1 || F_{\theta *} P_0), \tag{20}$$

where $G_{\theta^k}(\boldsymbol{x}) = g_{\theta_k} \circ g_{\theta_{k-1}} \circ \dots \circ g_{\theta_1}(\boldsymbol{x}), \forall k = 1, 2, \dots, K$ are the flow inverses, and $G_{\theta^0}(\boldsymbol{x}) := \boldsymbol{x}$. Not that the evaluation of $\mathcal{D}_{KL}(P_1 || F_{\theta *} P_0)$ requires the inverse of F_θ, which is expressed in the g_{θ_i}'s defined in (8). We defer the detailed derivation to Appendix B.

The transport cost in the equivalent formulation (20) can be computed via a single forward pass on the training data, which can be done together with the negative log-likelihood computation. As a result, the addition of the transport cost incurs minimal overhead compared to the standard NF training.

5.3. *Regularizing NF training with MFG costs*

The aforementioned method for solving high-dimensional MFGs using NFs leads to the view of NFs as a special type of MFG problems. Comparing the MFG formulation (18) with the standard NF training (8), one can interpret the MFG transport cost as a regularization term in the NF training with strength $\lambda_L/\lambda_{\mathcal{M}}$; we call this regularization a *trajectory regularization*. It is straightforward to implement it on top of any existing discrete NF models (not necessarily restricted to coupling flows). Since the transport cost aggregates squared differences between the input and the output of each flow, the regularization encourages smoothness in the intermediate transforms.

From the perspective of inverse problems, there exist many flows (transport plans) that can transform the base P_0 to P_1. Therefore, regularization helps ameliorate the inevitably ill-posed nature of NF training. Through OT theory, the L_2 regularized evolution of densities is characterized as the geodesic connecting P_0 and P_1 in the space of measures equipped with the Wasserstein-2 metric [51]. Since P_0 is always chosen to be absolutely continuous for NFs, the OT between the two densities will be a unique and bijective mapping known as the Monge map, $T(\boldsymbol{x})$. The same result holds for the discretized MFG problem, shown later in Theorem 6.4. Therefore, the NF training problem is no longer ill-posed. Furthermore, the optimal trajectory is then a linear interpolation between each point $\boldsymbol{x}$ and its destination $T(\boldsymbol{x})$: $F^*(\boldsymbol{x}, t) = (1-t)\boldsymbol{x} + tT(\boldsymbol{x})$.

From the perspective of machine learning, the transport cost provides an effective framework to control the flow's Lipschitz bound, which closely correlates with the robustness and the generalization capability [52,27]. Note that by enforcing smoothness in the time domain, the trajectory regularization also encourages smoothness in the spatial domain, owing to the triangle inequality:

$$\begin{aligned} \|F_K(\boldsymbol{x}) - F_K(\boldsymbol{y})\|_2^2 &= \left\| \sum_{i=0}^{K-1} (F_{i+1}(\boldsymbol{x}) - F_i(\boldsymbol{x})) - \sum_{i=0}^{K-1} (F_{i+1}(\boldsymbol{y}) - F_i(\boldsymbol{y})) + (\boldsymbol{x} - \boldsymbol{y}) \right\|^2 \\ &\le 2\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + K \sum_{i=0}^{K-1} \|F_{i+1}(\boldsymbol{x}) - F_i(\boldsymbol{x})\|_2^2 + K \sum_{i=0}^{K-1} \|F_{i+1}(\boldsymbol{y}) - F_i(\boldsymbol{y})\|_2^2, \end{aligned} \tag{21}$$

where $F_0 = Id, F_i = f_i \circ \dots f_1, i = 1, \dots, K$, and F_K is the NF model. By regularizing with the transport cost as in (18), the right-hand side can be controlled.

As observed by the authors of [26], explicit regularization methods such as l_2 weight decay has little impact on the model's generalization gap. In contrast, the trajectory regularization is implicit in that it imposes no direct penalty on the parameters. We find it to be much more effective than weight decay in reducing the NF's Lipschitz bound. We will demonstrate in the numerical experiments that a regularized evolution can reduce overfitting and improve density estimation.

6. Theoretical analysis

We first characterize the solutions of the MFG problem and its discretized analogue, when the interaction cost is absent and the transport cost is the kinetic energy. We prove that the discretized objective converges to the continuous version linearly, and that the minimum values agree between the two problems, for any number of discretized points. Through reduction to an OT problem, we show uniqueness of the optimizer for discretized MFG problems, thereby alleviating ill-posedness of the NF training objective.

In addition, we show that with mild assumptions on the MFG solution, certain classes of normalizing flows can be used to approximate the optimal trajectories of any MFG problem in L_2 locally. Our results build on the work [25], which establishes universality results for a broad class of coupling flows. In addition, we state that under the learned mapping, the evolution of the probability measure converges to the ground truth in distribution, at every discretized time step. We sketch the intuitions here and defer the proofs to the appendix.

6.1. *Relating continuous and discretized MFGs*

Our analysis focuses on the MFG problem without the interaction cost. We recall the discretized MFG problem with $\mathcal{I} \equiv 0$ here for convenience:

$$\begin{aligned}\inf_{F=\{F_i\}_{i=0}^K} \quad & \lambda_L K \cdot \mathbb{E}_{z\sim P_0}\left[\sum_{k=0}^{K-1} \|F_{i+1}(\boldsymbol{z}) - F_i(\boldsymbol{z})\|_2^2\right] + \mathcal{M}(F_{K*}p_0) := \mathcal{L}_K(F) \\ s.t. \quad & F_0(\boldsymbol{z}) = \boldsymbol{z}.\end{aligned} \tag{22}$$

First, we note that in the absence of the interaction cost, the discretized objective converges to the continuous analogue in $O(\frac{1}{K})$ when evaluated on any F.

Theorem 6.1. *Assume $\mathcal{I} \equiv 0$. The discretized MFG objective value converges to the continuous version at $O(\frac{1}{K})$ for any F, when both objectives are finite:*

$$|\mathcal{L}_K(F) - \mathcal{L}(F)| = O\left(\frac{1}{K}\right).$$

The proof is based on a straightforward Taylor expansion, similar to that used in finite difference error analysis. In general, one can use a higher order scheme to discretize (14) in time, yielding a higher order convergence rate. For example, the composite Simpson's rule is used in our numerical experiments.

Next, we consider the optimization and establish two lemmas that characterize the optimal trajectories of both the discretized and the continuous MFG problems.

Lemma 6.2. *Assume P_0 is absolutely continuous with density p_0. Let $F^*(\boldsymbol{z}, t)$ be an optimizer for (14) with $\mathcal{I} \equiv 0$. Define $F^*(\boldsymbol{z}, 1) := F_1(\boldsymbol{z})$. Then, $F^*(\boldsymbol{z}, t) = (1-t)\boldsymbol{z} + tF_1(\boldsymbol{z})$.*

Lemma 6.2 can be shown by recasting the MFG as a suitable OT problem. Then, the straight trajectories follow from the OT theory. A similar result holds for the discretized MFG problem.

Lemma 6.3. *Assume P_0 is absolutely continuous with density p_0. Let $\{F_i^*\}_{i=0}^K$ be an optimizer for (22) with $F_K^* := F_1$. Then, $F_i^* = (1-\frac{i}{K})\boldsymbol{z} + \frac{i}{K}F_1, i = 0, 1, ..., K$.*

Lemma 6.3 can be shown by recasting the problem in a similar fashion, followed by solving its KKT system. Combining Lemma 6.2 and Lemma 6.3, we can show that the optimal value of the discretized problem agrees with its continuous counterpart.

Theorem 6.4. *Assume $\mathcal{I} \equiv 0$. Both the continuous (14) and the discretized (22) MFG problems admit unique optimizers and their optimal values agree. In addition, the trajectory obtained from linearly interpolating the optimizers of the discretized MFG problem, $F(\boldsymbol{z}, t) = (1-t)\boldsymbol{z} + tF_K^*(\boldsymbol{z})$, is the optimizer of the continuous MFG problem.*

Theorem 6.4 has two important implications. First, when we use an NF to parameterize and solve the MFG problem with no interaction costs, the optimizer incurs no discretization error. Therefore, it is not necessary to use higher order discretization schemes for the problem. Second, the discretized MFG problem, which corresponds to the training objective (18) modulo parameterization, admits a unique solution. In other words, the regularized NF training is well-posed.

6.2. Universality of NFs for solving MFGs

The universality results from [25] indicate that NFs built with affine coupling transforms and sufficiently expressive conditioners are universal approximators for $\mathcal{D}^2$ locally in L_p, $p \in [1, \infty)$. Here, $\mathcal{D}^2 = \{f : Dom(f) \to Im(f) \subseteq \mathbb{R}^d\}$, where f is C^2 diffeomorphic, and $Dom(f) \subseteq \mathbb{R}^d$ is open and diffeomorphic to $\mathbb{R}^d$. As a result, affine coupling flows are well-suited for approximating the optimal MFG trajectories. To be more precise, we consider the following sets introduced in [25].

Definition 6.5. Define the set of single coordinate affine coupling flows with conditioner class $\mathcal{H}$ as $\mathcal{H}$**-ACF** $= \{\Psi_{d-1,s,t} : s, t \in \mathcal{H}\}$, where $\Psi_{k,s,t}(\boldsymbol{x}_{\leq k}, \boldsymbol{x}_{>k}) = (\boldsymbol{x}_{\leq k}, \boldsymbol{x}_{>k} \odot \exp(s(\boldsymbol{x}_{\leq k})) + t(\boldsymbol{x}_{\leq k}))$ is an affine coupling flow. Further, define the set of invertible neural networks $\textbf{INN}_{\mathcal{H}\textbf{-ACF}} = \{W_1 \circ g_1 \circ \ldots \circ W_n \circ g_n : n \in \mathbb{N}, g_i \in \mathcal{H}\textbf{-ACF}, W_i(\boldsymbol{x}) = A_i x + b_i, A_i \text{ invertible}\}$, which is a family of normalizing flows with flow transforms from the class of invertible functions $\mathcal{H}$**-ACF**.

Note that the affine coupling flows used in our numerical experiments belong to $\textbf{INN}_{\mathcal{H}\textbf{-ACF}}$, where $\mathcal{H}$ is the class of ReLU networks. It turns out that the universality of $\textbf{INN}_{\mathcal{H}\textbf{-ACF}}$ provides an ideal characterization for the approximate solutions of (14).

Theorem 6.6. *Let $\mathcal{H}$ be sup-universal for $C_c^\infty(\mathbb{R}^{d-1})$ and contain piecewise C^1 functions. For a fixed $K \in \mathbb{N}$, let the optimizer of* (14) *be $F^*(\boldsymbol{z},t)$ and its evaluation on the grid points $t_i = i\Delta t$ be $F^*(\boldsymbol{z},t_i) = \Phi_{t_{i-1}}^{t_i} \circ \ldots \Phi_{t_0}^{t_1}$, $i = 1, \ldots, K$. Assume $F^*(\boldsymbol{z},t_i) \in \mathcal{D}^2$, $i = 1, \ldots, K$. Then, for all $\epsilon > 0$, $p \in [1,\infty)$, and compact sets A, there exists $\{F_{\theta^i}\}_{i=1}^K \subset \mathbf{INN}_{\mathcal{H}\text{-}\mathbf{ACF}}$ such that*

$$\|F^*(\cdot,t_i) - F_{\theta^i}\|_{p,A} < \epsilon, \quad i = 1, \ldots, K.$$

This result can be directly obtained from the conclusion in [25]. We remark that the conditions on $\mathcal{H}$ can be satisfied by fully connected networks equipped with the ReLU activation. In addition, we show that the push-forward measures from the approximated optimal trajectories converge to the ground truth in distribution. This ensures the validity of the learned evolution of the agent population.

Theorem 6.7. *Let P_0 be absolutely continuous with a bounded density p_0, and $F(\boldsymbol{z},t)$ be the optimizer for* (14) *such that $F(\boldsymbol{z},t_i) \in \mathcal{D}^2$, $i = 1, \ldots, K$. Then, for each $k = 1, 2, \ldots, K$, there exists a sequence of normalizing flows $\{F_{\theta_k}^n\}_{n=1}^\infty := \{f_{\theta_k}^n \circ \cdots \circ f_{\theta_1}^n\}_{n=1}^\infty \subset \mathbf{INN}_{\mathcal{H}\text{-}\mathbf{ACF}}$ such that*

$$F_{\theta_k *}^n P_0 \xrightarrow{d} F(\cdot,t_k)_* P_0, \quad \text{as } n \to \infty,$$

where $\xrightarrow{d}$ denotes convergence in distribution.

The proof is based on the Lipschitz-bounded definition of convergence in distribution. See the proof in the appendix.

7. Numerical experiments

Within this section, we organize the numerical results into two parts. The first part showcases NFs as an effective parameterization for solving high-dimensional MFG problems. The second part leverages MFG transport costs to improve the robustness of NF models for generative modeling.

7.1. Solving MFGs with NFs

In the first part, we follow the settings in [22] to construct two MFG problems. They are designed so that the behavior of the MFG solutions is invariant to dimensionality, when projected onto the first two components. This property allows for qualitative evaluation through visualization. As the dimensionality increases from 2 to 100, the problem becomes more challenging in terms of the neural network's capacity and the computational complexity. We further extend our numerical experiments to multi-group path planning. In these experiments, we use spline flows as it is commonly believed that they are more expressive than coupling flows.

7.1.1. OT in high dimensional spaces

In this experiment, we devise a dynamically formulated OT problem that can be solved efficiently with classic numerical methods in $d \le 3$ [15–21,30]. The optimization problem is the following:

$$\begin{aligned} &\inf_{p,\boldsymbol{v}} \int_0^1 \int_\Omega p(\boldsymbol{x},t)\|\boldsymbol{v}(\boldsymbol{x},t)\|_2^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \\ &s.t. \quad \partial_t p(\boldsymbol{x},t) + \nabla_{\boldsymbol{x}} \cdot (p(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t)) = 0, \quad \boldsymbol{x} \in \Omega, t \in [0,T] \\ &\qquad p(\boldsymbol{x},0) = p_0(\boldsymbol{x}), \boldsymbol{x} \in \Omega; \quad p(\boldsymbol{x},1) = p_1(\boldsymbol{x}), \boldsymbol{x} \in \Omega. \end{aligned} \tag{23}$$

Let $\mathrm{N}(\boldsymbol{x};\mu,\Sigma)$ denote the density function of a multivariate Gaussian with mean $\mu \in \mathbb{R}^d$ and covariance $\Sigma \in \mathbb{R}^{d\times d}$. We set the initial density to be $p_0(\boldsymbol{x}) = \mathrm{N}(\boldsymbol{x};0,0.3I)$ and the terminal density to be $p_1(\boldsymbol{x}) = \frac{1}{8}\sum_{i=1}^8 \mathrm{N}(\boldsymbol{x};\mu_i,0.3I)$, where $\mu_i = 4\cos(\frac{\pi}{4}i)e_1 + 4\sin(\frac{\pi}{4}i)e_2, \forall i = 1, \ldots, 8$, and e_1, e_2 are the first two standard basis vectors.

Contrary to MFGs, the OT problem imposes a strict constraint on the matching of the terminal density. To handle this, we cast the OT as a variational MFG problem by penalizing the mismatch of the terminal density using the KL divergence, and we parameterize the MFG with the coupling version of the neural spline flow (NSF-CL) [29], which employs flexible rational-quadratic coupling functions to learn expressive transformations. Overall, we find NSF-CL to be a robust architecture for solving different kinds of MFGs.

An illustration of the agent trajectories obtained from our method is given in Fig. 1 and the evolution of the population density is displayed in Fig. 2. We see that the NF learns to equally partition the initial Gaussian density and transport each partition to a separate Gaussian in an approximately straight trajectory. Moreover, it does so consistently as the problem dimension grows. More experimental results can be found in the appendix.

We compute the MFG cost term and compare our results to a traditional Eulerian method [21] as well as the MFGNet [22], another deep learning approach based on continuous NFs. The Eulerian method is provably convergent, but

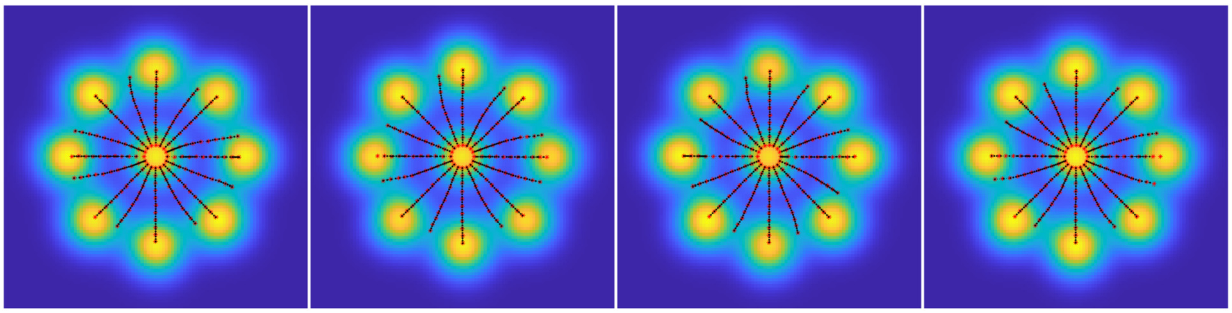


Fig. 1. Left to right: samples of 16 computed trajectories in 2D, 10D, 50D, and 100D.

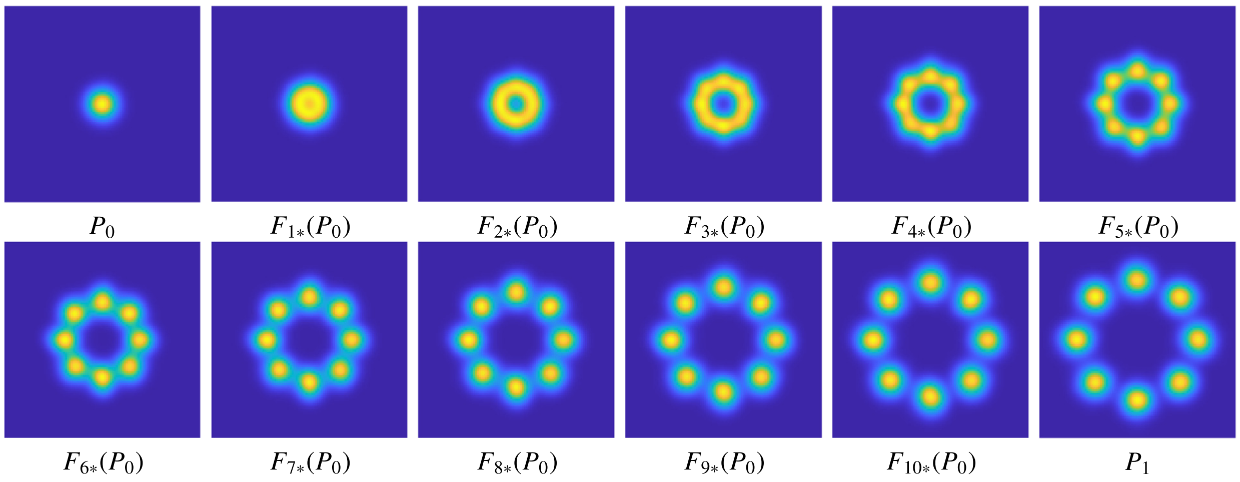


Fig. 2. Evolution of the density for the OT problem (23) in 50D.

Table 1
Comparison of methods for solving the OT problem (23). L: transport cost; $\mathcal{M} = D_{KL}$: terminal cost. *The first 500 iterations are trained with less data for a warm start.

Method	Dimension	L	$\mathcal{M}$	Time/iter	Number of iters
Eulerian	2	9.8	0.1725	-	-
MFGNet	2	9.9	0.1402	2.038	1e3*
Ours	2	10.1	0.0663	0.549	1e5
MFGNet	10	10.1	0.1616	8.256	1e3*
Ours	10	10.1	0.0641	0.561	1e5
MFGNet	50	10.1	0.1396	81.764	1e3*
Ours	50	10.1	0.0641	0.584	1e5
MFGNet	100	10.1	0.1616	301.043	1e3*
Ours	100	10.1	0.0622	0.625	1e5

its reliance on a discrete grid renders it infeasible for dimensions higher than three. For a fair comparison, we report the unweighted cost and treat the KL divergence $D_{KL}(F(\cdot, T)_* P_0 || P_1)$ as the terminal cost.

Table 1 demonstrates that the NF-based parameterization can solve MFG problems up to 100 dimensions while maintaining a consistent transport cost. Compared to the MFGNet, our approach yields a similar transport cost but a much lower terminal cost, entailing a more accurate density matching at the final time. We attribute the improvement in terminal matching to the trajectory-based parameterization of the MFG. By encoding the continuity equation (4) in the flow architecture, our approach sidesteps the errors incurred in the numerical schemes used in the MFGNet, when approximating the solution.

We remark that our model is optimized with ADAM [53] while the MFGNet is trained with BFGS. The superlinear convergence of BFGS allows the MFGNet to converge with fewer iterations, at the price of higher costs per update. In addition, our implementation is based on Pytorch, which effectively leverages GPU parallelism to achieve an optimal runtime scaling, up to 100 dimensions. While being slower than MFGNet in lower dimensions, our approach uses only one third of the time in $d = 100$, since its runtime is nearly constant in different dimensions.

7.1.2. Crowd motion

In this experiment, we consider an MFG problem with nonzero interaction costs. Following the general setting in (14), we use a two-part interaction cost $\mathcal{I}$:

$$\begin{aligned}\mathcal{I}_P(F(\cdot,t)_*p_0) &:= \lambda_P \int_\Omega Q(\boldsymbol{x})\mathrm{d}(F(\cdot,t)_*P_0)(\boldsymbol{x}) = \lambda_P \int_\Omega Q(F(\boldsymbol{x},t))p_0(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\\ \mathcal{I}_E(F(\cdot,t)_*p_0) &:= \lambda_E \int_\Omega (\log p_0(\boldsymbol{x}) - \log|\det \nabla F(\boldsymbol{x},t)|)p_0(\boldsymbol{x})\mathrm{d}x\\ \mathcal{I}(F(\cdot,t)_*p_0) &:= \lambda_{\mathcal{I}}(\mathcal{I}_P(F(\cdot,t)_*p_0) + \mathcal{I}_E(F(\cdot,t)_*P_0)).\end{aligned} \tag{24}$$

Here, $Q(\boldsymbol{x})$ acts as an obstacle that incurs a cost for an agent that passes through it. It is set to be the density of a bivariate Gaussian centered at the origin, with a magnitude of 50:

$$Q(\boldsymbol{x}) := 50 \cdot \mathrm{N}(\boldsymbol{x}; 0, \mathrm{diag}(1, 0.5)). \tag{25}$$

In dimensions higher than two, we evaluate the first two components of $F(\boldsymbol{x},t)$ on Q to compute $\mathcal{I}_P$. The initial and the terminal densities are chosen as Gaussians: $p_0(\boldsymbol{x}) = \mathrm{N}(\boldsymbol{x}; 3e_2, 0.3I)$, $p_1(\boldsymbol{x}) = \mathrm{N}(\boldsymbol{x}; -3e_2, 0.3I)$; and the terminal cost is identical to that in the OT experiment (23). Intuitively, this problem aims at transporting an initial Gaussian density to a different location, while avoiding an obstacle placed at the origin. Moreover, the optimal trajectories are invariant to the dimensionality on the first two components, allowing us to visualize the dynamics in high dimensions.

Similar to the transport cost L, we perform a discretization on (24), based on the NF parameterization. Note that $\mathcal{I}$ is integrated in time in the MFG problem (14). Thus, discretizing the integral with the right-point rule yields

$$\int_0^1 \mathcal{I}(F(\cdot,t)_*p_0)\mathrm{d}t = \mathbb{E}_{\boldsymbol{x}\sim P_0}\left\{\lambda_{\mathcal{I}}\int_0^1 [\lambda_P Q(F(\boldsymbol{x},t)) + \lambda_E(\log p_0(\boldsymbol{x}) - \log|\det\nabla F(\boldsymbol{x},t)|]\mathrm{d}t\right\} \tag{26}$$

$$\approx \frac{\lambda_{\mathcal{I}}}{K}\mathbb{E}_{\boldsymbol{x}\sim P_0}\left\{\sum_{i=1}^K [\lambda_P Q(F_i(\boldsymbol{x})) + \lambda_E(\log p_0(\boldsymbol{x}) - \log|\det\nabla F_i(\boldsymbol{x}_{i-1})|)]\right\}, \tag{27}$$

where $F_i = f_i \circ f_{i-1} \circ \ldots \circ f_1, \forall i = 1, \ldots, K$ and $\boldsymbol{x}_0 := \boldsymbol{x}, \boldsymbol{x}_j := f_j(\boldsymbol{x}_{j-1}), \forall j = 1, \ldots, K-1$. Similar to before, our discretization is consistent with the $O(\frac{1}{K})$ global error. In practice, we use fourth-order forward difference for the $\partial_t F(\boldsymbol{z},t)$ term in L and approximate all integrals with Simpson's rule. The overall numerical scheme converges to the continuous MFG with $O(\frac{1}{K^4})$ global error.

Similar to the OT experiment, we use the same NSF-CL flow [29] to solve the crowd motion problem from 2 to 100 dimensions and compare the computed cost terms to those of MFGNet in Table 2. Compared to the baseline, our method yields comparable transport cost but lower interaction and terminal costs, suggesting a more accurate characterization of the agents' behavior in avoiding the conflicts with the obstacle, as well as a more authentic matching of the final population distribution. Similar to the MFGNet, our computed $L, \mathcal{M}$ costs are approximately invariant with respect to the dimensionality, as further validated by the visualizations of the sampled trajectories shown in Fig. 3. The agents plan their paths to circumvent the obstacle; moreover, trajectories of symmetric initial points are symmetric as well, which is expected from the problem setup. In Fig. 4, we also show the optimal trajectories at different levels of aversion preference, further validating the robustness of the method. From left to right, the agents steer farther away from the obstacle while maintaining approximately symmetric trajectories and accurate terminal matching. The associated costs for each scenario are given in the appendix, together with more numerical results.

7.1.3. Multi-group path planning

In the above crowd motion experiment, we demonstrated that the interaction cost can be used to encourage a single population to avoid obstacles in the environment. Here, we take one step further and investigate path planning when multiple populations are moving simultaneously. Therein, the interaction cost is used to discourage conflicts among different populations.

We consider a generalized MFG problem that accommodates multiple densities; this problem is also explored in [54,55]. We consider N_p populations $\{p^i(\boldsymbol{x},t)\}_{i=1}^{N_p}$ and their associated trajectories $\{F^i(\boldsymbol{x},t)\}_{i=1}^{N_p}$, each of which is parameterized by an NF. Every population maintains an independent transport cost $\lambda_L \int_0^1 \int_\Omega p_0(\boldsymbol{x})\|\partial_t F^i(\boldsymbol{x},t)\|_2^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t$ and an independent terminal cost $\mathcal{M}_i = \lambda_{\mathcal{M}}\mathcal{D}_{KL}(P_1^i||F^i(\cdot,1)_*P_0^i)$, where P_1^i, P_0^i are the initial and terminal distributions for the i-th population, respectively.

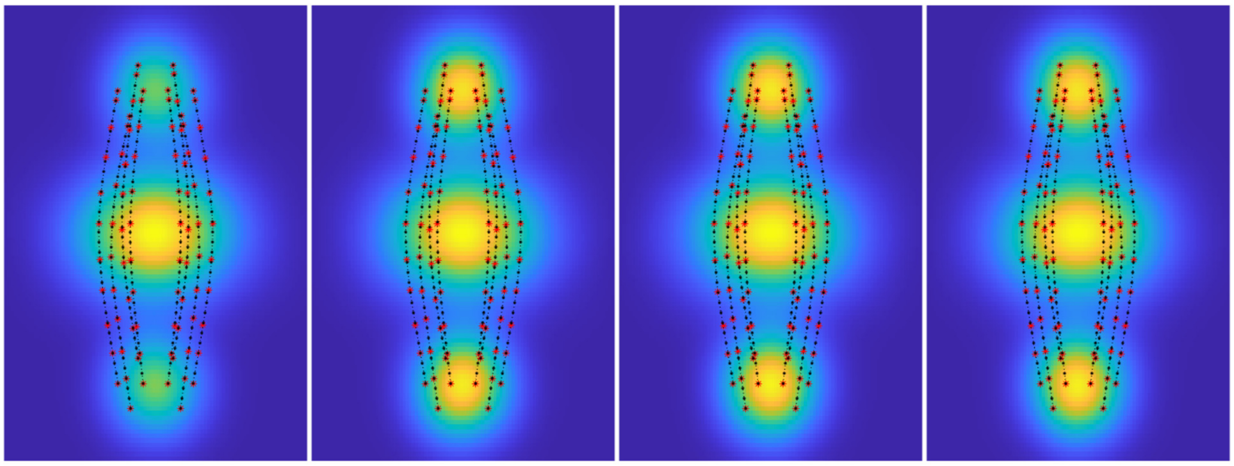


Fig. 3. Left to right: samples of the computed trajectories in 2D, 10D, 50D, and 100D. The top and bottom densities are p_0 and p_1, respectively, while the middle density denotes the obstacle.

Table 2
Comparison of methods for solving the crowd motion problem (24). L: transport cost; $\mathcal{I}$: interaction cost; $\mathcal{M}$: terminal cost. *The first 500 iterations are trained with less data for a warm start.

Method	Dimension	L	$\mathcal{I}$	$\mathcal{M}$	Time/iter	Number of iters
Eulerian	2	31.8	2.27	0.1190	-	-
MFGNet	2	33.0	2.29	0.1562	4.122	1e3*
Ours	2	32.8	2.19	0.0417	0.559	1e5
MFGNet	10	33.0	2.29	0.1502	17.205	1e3*
Ours	10	32.9	2.13	0.0436	0.568	1e5
MFGNet	50	33.0	1.91	0.1440	134.938	1e3*
Ours	50	32.9	1.82	0.0381	0.581	1e5
MFGNet	100	33.0	1.49	0.2000	241.727	1e3*
Ours	100	33.0	1.36	0.0464	0.625	1e5

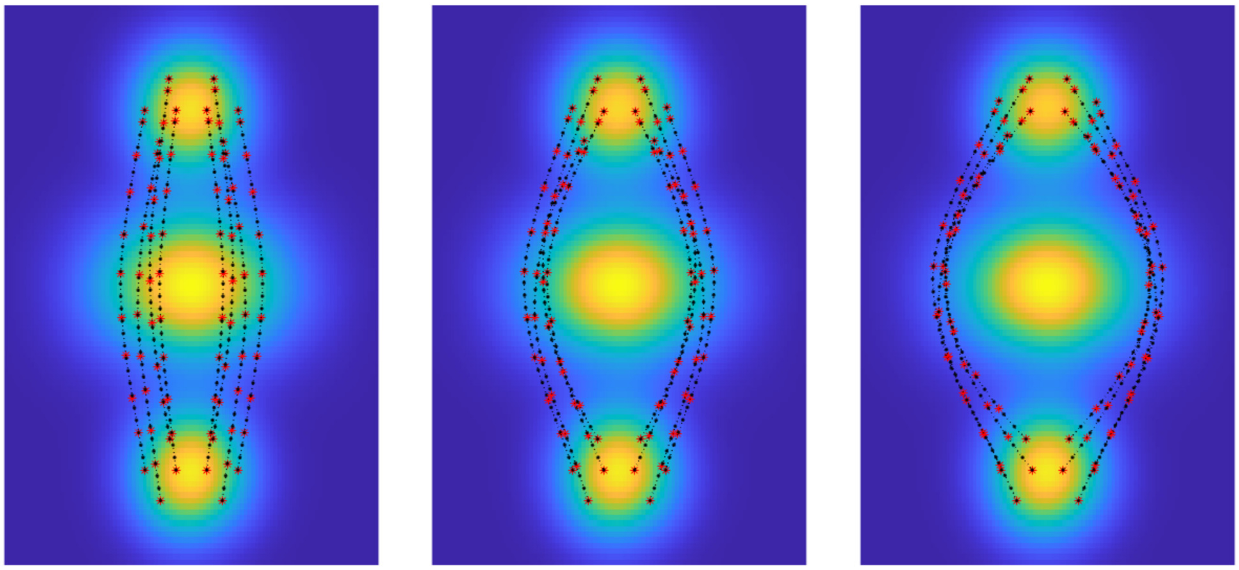


Fig. 4. Computed trajectories for the crowd motion problem (24) in 10D, projected onto the first two dimensions. From left to right: minor ($\lambda_{\mathcal{I}} = 0.2$), moderate ($\lambda_{\mathcal{I}} = 0.5$), and strong ($\lambda_{\mathcal{I}} = 1$) penalty on conflicts with the obstacle. The left figure corresponds to the case reported in Table 2.

Table 3
Results of solving the multi-group path planning problem by using different interaction weights. $\lambda_{\mathcal{I}}$: weight for $\mathcal{I}$; L: transport cost; $\mathcal{I}$: interaction cost; $\mathcal{M}$: terminal cost.

2D, 2 POPULATIONS				3D, 2 POPULATIONS				2D, 8 POPULATIONS			
$\lambda_{\mathcal{I}}$	L	$\mathcal{I}$	$\mathcal{M}$	$\lambda_{\mathcal{I}}$	L	$\mathcal{I}$	$\mathcal{M}$	$\lambda_{\mathcal{I}}$	L	$\mathcal{I}$	$\mathcal{M}$
0	3.9755	0.8992	0.0038	0	5.9899	0.8897	0.0023	0	120.8807	12.5090	0.0067
3	5.5874	0.6556	0.0034	3	7.2835	0.6574	0.0017	1	125.7670	10.6740	0.0086
5	9.1048	0.4690	0.0099	5	10.8889	0.4677	0.0047	3	164.5073	6.0951	0.0089

Table 4
Results of solving the multi-group path planning problem by using different inter-group collision weights. λ_{inter}: weight for the inter-group collision cost; L: transport cost; $\mathcal{I}_i nter$: inter-group collision cost. $\mathcal{I}_{obs}$: obstacle collision cost. $\mathcal{M}$: terminal cost.

λ_{inter}	L	$\mathcal{I}_{inter}$	$\mathcal{I}_{obs}$	$\mathcal{M}$
0	1.8850	0.6011	0.2403	0.0013
0.1	6.0718	0.7436	0.0369	0.0457
2.0	6.5895	0.5894	0.0381	0.0457

To keep the setting simple yet illustrative, we consider an inter-group interaction cost that encourages different populations to avoid each other in their paths. In a real-life scenario, this can be thought of as path planning for multiple groups of drones, and the desired outcome is for them to arrive at their desired locations through short paths while minimizing collisions. Formally, the interaction cost is modeled by a Gaussian kernel $\mathcal{I} = \sum_{j\neq i} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} e^{-\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2} \mathrm{d}P^i(\boldsymbol{x},t)\mathrm{d}P^j(\boldsymbol{y},t)$, which decays exponentially as the radial distance between two populations increases.

In Fig. 5, we show results for problems in two and three dimensions. For the 2D experiment with two populations, we choose $p_0^1(\boldsymbol{x},t) = \mathrm{N}((0,0),0.01I)$, $p_1^1(\boldsymbol{x},t) = \mathrm{N}((1,1),0.01I)$ as the initial and terminal density for the first population, and $p_0^2(\boldsymbol{x},t) = \mathrm{N}((1,0),0.01I)$, $p_1^2(\boldsymbol{x},t) = \mathrm{N}((0,1),0.01I)$ for the second population. For the 3D experiment with two populations, we have $P_0^1(\boldsymbol{x},t) = \mathrm{N}((0,0,0),0.01I)$, $p_1^1(\boldsymbol{x},t) = \mathrm{N}((1,1,1),0.01I)$, $p_0^2(\boldsymbol{x},t) = \mathrm{N}((1,0,0),0.01I)$, $p_2^1(\boldsymbol{x},t) = \mathrm{N}((0,1,1),0.01I)$. For the 2D experiment with eight populations, the densities are $p_i(\boldsymbol{x}) = \mathrm{N}(\boldsymbol{x};\mu_i,0.05I)$, where $\mu_i = 4\cos(\frac{\pi}{4}i)e_1 + 4\sin(\frac{\pi}{4}i)e_2, \forall i = 1,...,8$. Each population uses the density mirrored across the origin as its destination.

In these problems, the initial and the terminal densities are configured to create a full collision if the interaction term is absent. As the weights on the interaction cost increase, the populations coordinate a delicate dance to avert each other during their courses of travel. Note that the computed trajectories are approximately symmetric, which is consistent with the symmetric nature of the problem setup. We summarize the computed unweighted costs in Table 3. As higher weights are placed on the interaction cost, the populations spend more effort maneuvering and hence the transport cost increases.

Lastly, we provide a simple yet illustrative example for multi-group motion planning in the presence of obstacles. Imagine two lanes of traffic traveling towards their respective destinations. When there are no obstacles, each group stays in its lane and reaches its destination without having to interact with the other groups. With the addition of an obstacle in one of the lanes, however, the blocked group temporarily enters the other group's lane to avoid collision with the obstacle, and the other group has to then adjust its course to avoid the intruding group. This dynamics is captured in Fig. 6, where the populations are $p_0^1(\boldsymbol{x},t) = \mathrm{N}((0,0),0.01I)$, $p_1^1(\boldsymbol{x},t) = \mathrm{N}((0,1),0.01I)$; $p_0^2(\boldsymbol{x},t) = \mathrm{N}((1,0),0.01I)$, $p_1^2(\boldsymbol{x},t) = \mathrm{N}((1,1),0.01I)$, and the obstacle is $Q(\boldsymbol{x}) = 50 \cdot \mathrm{N}(\boldsymbol{x};(0.5,2),\mathrm{diag}(0.4,0.03))$. See also Table 4.

7.2. Improving NF training with MFGs

From the MFG point of view, the standard training of NFs only considers minimizing the KL divergence, which corresponds to the terminal cost, while neglecting the transport and interaction costs. Therefore, intermediate distributions may be meaningless artifacts of the flow parameterization and they often come with enormous distortions. By viewing NFs as a parameterization of the discretized MFG, it is natural to introduce the transport cost into NF training. We showed in Theorem 6.4 that the training problem admits a unique mapping as its solution. In other words, the problem is no longer ill-posed.

More interestingly, the unique transport-efficient mapping tends to be well-behaved, in terms of the Lipschitz bound illustrated in (21). This behavior is closely related to the model's complexity and the robustness of the neural network [26]. As a result, the weight on the trajectory regularization serves as an effective means of controlling the Lipschitz bound. With appropriate choices of the weight, we observe in a variety of tabular and image datasets that the trained model has a better generalization performance.

7.2.1. Synthetic datasets

In this experiment, we compare the intermediate flows trained on 2D synthetic datasets, with and without the transport cost.

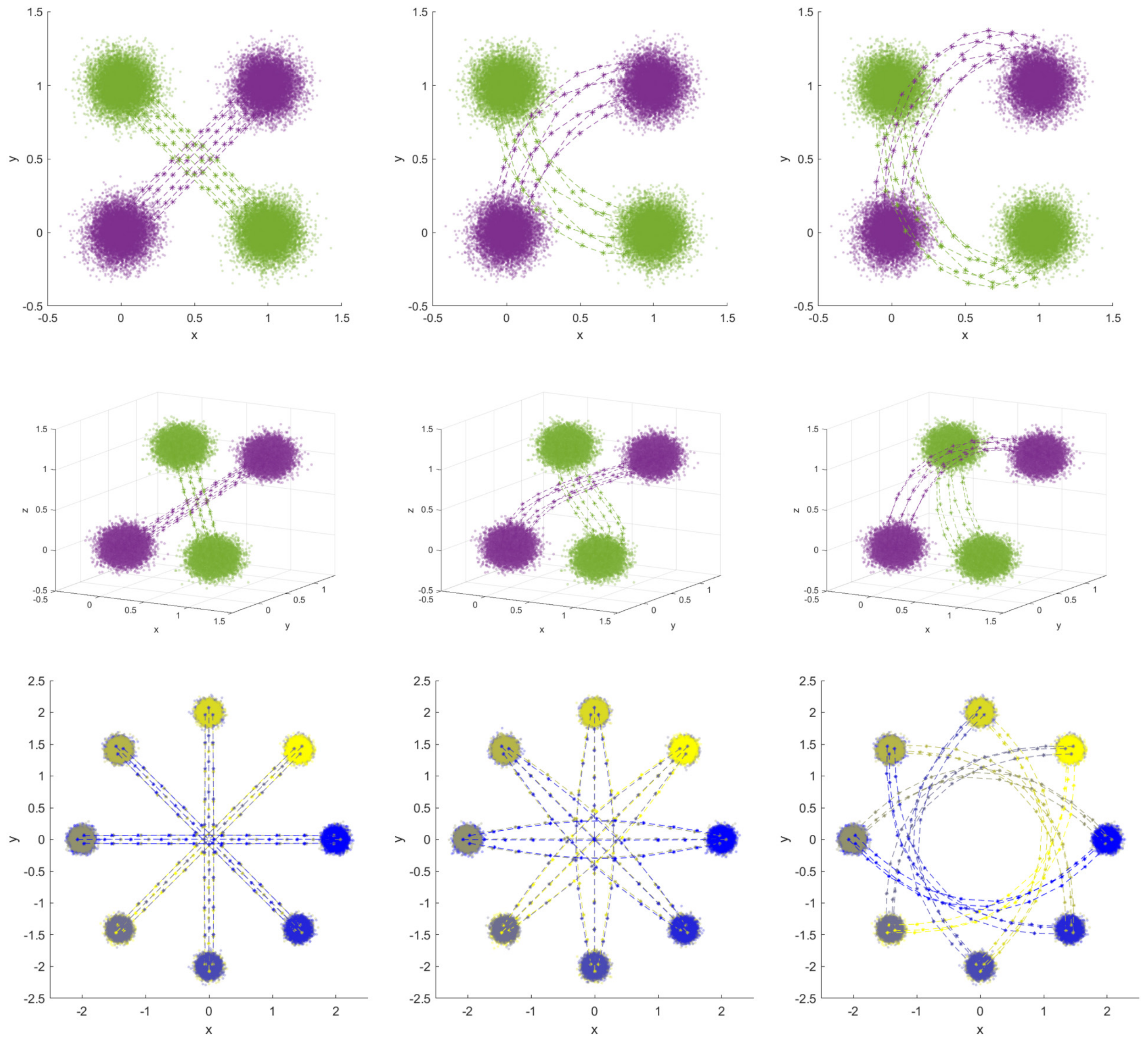


Fig. 5. Computed trajectories for the multi-group path planning problem. From left to right: zero, moderate, and strong penalty on conflicts between populations. Colors indicate different populations. From top to bottom: two populations in 2D, two populations in 3D, and eight populations in 2D. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

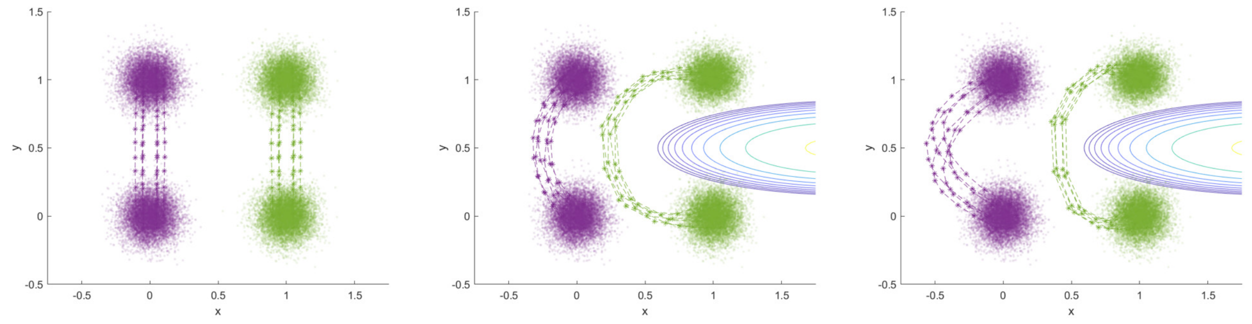


Fig. 6. Computed trajectories for multi-group path planning with obstacles. Colors indicate different populations. Left: two lanes of traffic traveling to their destination independently. Middle: in the presence of obstacles (shown as level curves), the green population turns left to avoid collision with the obstacle, forcing the purple population to adjust its course to avoid collision with the green population. Right: Same setting as the middle, but the populations are more defensive against inter-group collisions.

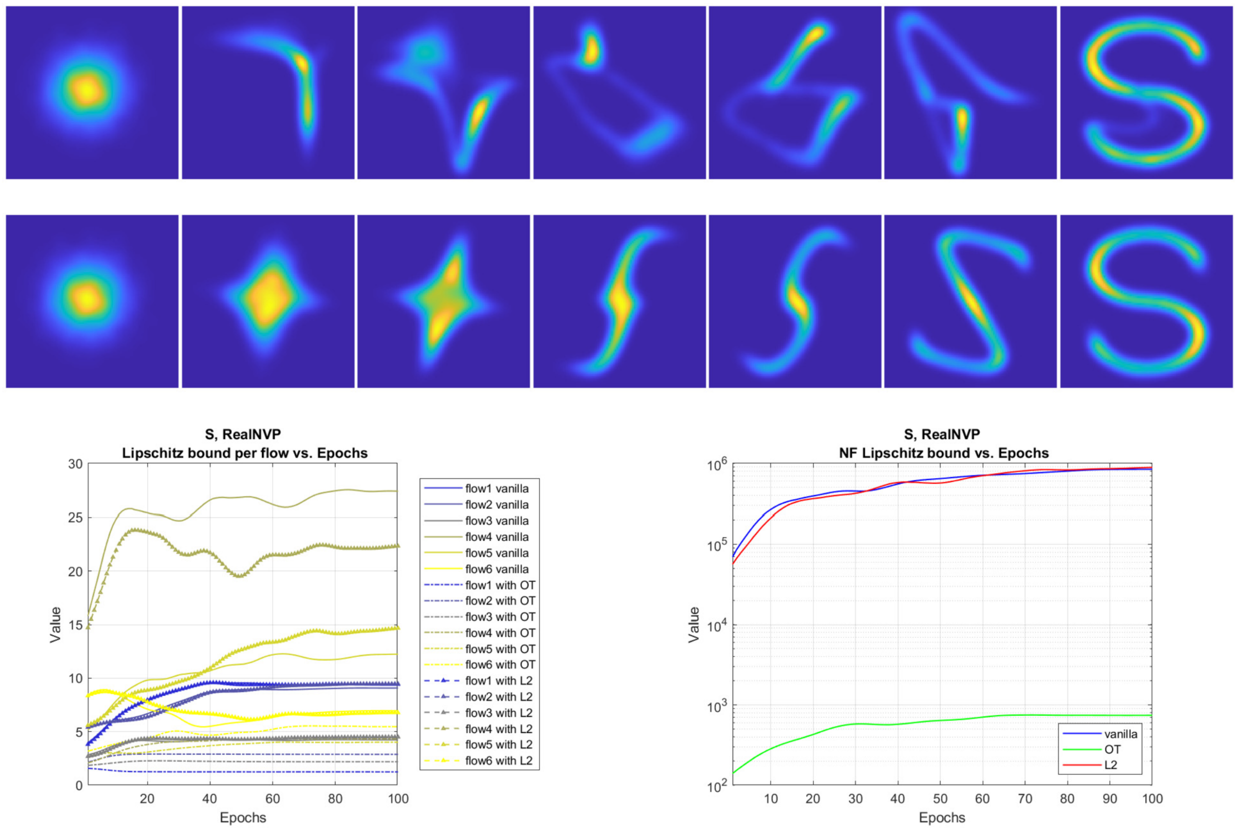


Fig. 7. Top row: output of each intermediate RealNVP flow between a single Gaussian and the S-shape density, trained without using the transport cost. Middle row: same but trained with using the transport cost. Bottom left: the Lipschitz bound for each flow over the training epochs. Bottom right: the Lipschitz bound for the entire flow. The weights of the transport regularization are chosen so that the negative log-likelihood is not severely obstructed.

The introduction of the transport cost encourages NFs to evolve the data to their desired destinations with the least amount of distortion. As a result, the intermediate flows are more sensible and visually appealing, as illustrated in Fig. 7 (see more examples in the appendix; Figs. E.18, E.19, E.20). By comparing the NFs trained with and without the transport cost, we can see that in the absence of the regularization, existing NF models cannot be expected to learn trajectories that are approximately optimal under the L_2 cost. Instead, the learned trajectories are artifacts of the flow parameterization and they are sensitive to initialization.

Visually, NFs trained with the transport cost incur significantly less distortion, so it is not surprising for the model's Lipschitz bound to be much smaller. In Fig. 7, we compare the Lipschitz bounds between the standard NF, the NF trained with l_2 weight decay, and the NF trained with trajectory regularization. It is evident that while weight decay has a small impact on the Lipschitz bound, the trajectory regularization yields more noticeable improvement.

7.2.2. *Tabular datasets*

To demonstrate the benefit of training with transport costs, we replicate the density estimation experiments on popular benchmarks from the UCI repository and BSDS300 [56,57], using the same setup as in [35]. Our approach requires no changes to the baseline architecture and hence any existing discrete NF is applicable. Here, we use two popular NF architectures, RealNVP and NSF-AR, which are representatives of coupling flows and autoregressive flows, respectively [28,29]. The former uses simple (affine) coupling functions while the latter uses sophisticated (spline) coupling functions.

Table 5 indicates that NF models trained with OT (trajectory regularization) are able to achieve better performance on the test set across all tabular datasets. The Lipschitz bounds for RealNVP trained on the datasets Power, Gas, and BSDS300 are visualized in Fig. 8 (for other datasets, see the appendix). Clearly, while l_2 regularization has a mixed impact on the model's Lipschitz bound, the trajectory regularization reduces the bound effectively and improves the model's generalization performance.

We can interpret the discretized points on the OT trajectories as landmarks that should be visited by each flow in the interim. Our training setup is thus analogous to DSN [58], which guides the learning of intermediate layers with intermediate classifiers. Consistent with their observation, the transport cost provides a natural mechanism for distributing losses throughout the network to improve convergence of the initial flows.

Table 5
The log-likelihoods obtained by using different models. Bolded results highlight the improvements of the OT-based models over their counterparts (e.g., RNVP + OT versus RNVP). The other NF models are included for reference. The numbers after the ± symbol are twice the standard errors.

Model	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
MAF (10)	0.24 ± 0.01	10.08 ± 0.02	-17.73 ± 0.02	-12.24 ± 0.45	154.93 ± 0.28
MAF MoG	0.30 ± 0.01	9.59 ± 0.02	-17.39 ± 0.02	-11.68 ± 0.44	156.36 ± 0.28
NAF	0.60 ± 0.02	11.96 ± 0.33	-15.32 ± 0.23	-9.01 ± 0.01	157.43 ± 0.30
SOS	0.60 ± 0.01	11.99 ± 0.41	-15.15 ± 0.10	-8.90 ± 0.11	157.48 ± 0.41
Quad. Spline	0.64 ± 0.01	12.80 ± 0.02	-15.35 ± 0.02	-9.35 ± 0.44	157.65 ± 0.28
RNVP	0.335 ± 0.013	11.017 ± 0.022	-17.983 ± 0.021	-11.540 ± 0.469	153.398 ± 0.283
NSF-AR	0.650 ± 0.013	13.001 ± 0.018	-14.145 ± 0.025	-9.662 ± 0.501	157.546 ± 0.282
RNVP + OT	**0.374 ± 0.013**	**11.063 ± 0.022**	**-17.879 ± 0.0215**	**-11.167 ± 0.466**	**153.406 ± 0.283**
NSF-AR + OT	**0.654 ± 0.012**	**13.018 ± 0.018**	**-13.989 ± 0.026**	**-9.395 ± 0.494**	**157.729 ± 0.280**

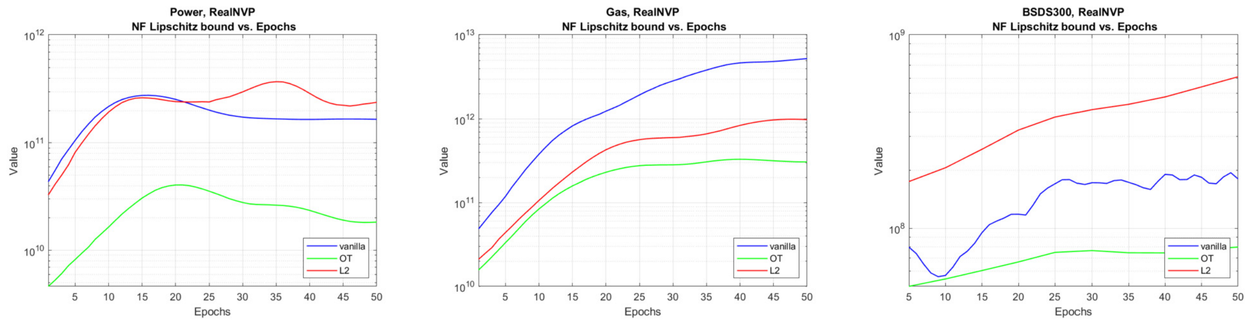


Fig. 8. The Lipschitz bound over training epochs, for different tabular datasets (left to right: Power, Gas, and BSDS300). The "L2" label denotes NF trained with weight decay, which is shown to be consistently worse in controlling NF's Lipschitz constant compared to the proposed OT regularization.

Table 6
The bits per dimension obtained by using different models (Glow versus Glow with transport cost). The numbers after the ± symbol are twice the standard errors.

Model	MNIST	FMNIST	CIFAR-10	SVHN	EUROSAT
Glow	1.182 ± 0.005	2.944 ± 0.017	3.489 ± 0.012	2.156 ± 0.005	2.670 ± 0.006
Glow + OT	**1.142 ± 0.005**	**2.901 ± 0.017**	**3.465 ± 0.012**	**2.146 ± 0.005**	**2.658 ± 0.006**

7.2.3. Image datasets

To further demonstrate the usefulness of our transport-efficient model, we add transport costs to Glow [50], a prominent NF architecture well-suited for generating high-quality images. Similar to tabular data, we observe improvements in density estimation via training Glow with the transport cost, in a number of datasets.

In Table 6, we select five popular image datasets to compare the density estimation results. The selected datasets include both greyscale and color images, with sizes varying between 28×28 and 64×64. The Glow implementation is down-scaled to fit on a single GPU; and the baseline results are similar to those reported by other works that follow this approach; e.g. [59]. In addition, we omit the 1x1 convolution transforms [50] in the transport cost, as they play a role similar to permutations. Overall, we observe improvements in the estimated density across all five image datasets, with similar regularization weights.

8. Conclusion

In this work, we unravel connections between the training of an NF and the solving of an MFG. Based on this insight, we explore two research directions. First, we employ expressive flow transforms to accurately and efficiently solve high-dimensional MFG problems. We also approximation analysis for the proposed method. Compared to existing deep learning approaches, our approach encodes the continuity equation into the NF architecture and achieves a more accurate terminal matching. In addition, our implementation makes effective use of GPU parallelism to achieve fast computations in high dimensions. Second, we introduce transport costs into the standard NF training and demonstrate the effectiveness in controlling the network's Lipschitz bound. As a result, NFs trained with appropriate trajectory regularization are shown to achieve better generalization performance in density modeling across tabular and image datasets.

CRediT authorship contribution statement

Han Huang: Conceptualization, Formal analysis, Methodology, Validation, Visualization, Writing – original draft. **Jiajia Yu:** Conceptualization, Methodology. **Jie Chen:** Conceptualization, Methodology. **Rongjie Lai:** Conceptualization, Methodology, Supervision, Writing – original draft.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Rongjie Lai reports financial support was provided by National Science Foundation. Bill Huang reports financial support was provided by National Science Foundation. Jiajia Yu reports financial support was provided by National Science Foundation. Rongjie Lai reports a relationship with National Science Foundation that includes: funding grants.

Data availability

Data will be made available on request.

Acknowledgements

This work is supported in part by an NSF Career Award (DMS-1752934) and NSF DMS-2134168.

Appendix A. Derivations regarding $F(\boldsymbol{x}, t)$ and $\Phi_{t_i}^{t_{i+1}}(\boldsymbol{x})$

Here, we provide additional details for some statements made in section 5, regarding the agent trajectories $F(\boldsymbol{x}, t)$ and $\Phi_{t_i}^{t_{i+1}}(\boldsymbol{x})$. First, we will derive the ODE system defining $F(\boldsymbol{x}, t)$ that is central to our MFG reformulation. Second, we will show how $F_k(\boldsymbol{x}) := F(\boldsymbol{x}, t_k)$ can be decomposed into a series of flow maps $\Phi_{t_i}^{t_{i+1}}(\boldsymbol{x})$ that we subsequently parameterize with neural networks.

At the base level, we have in (1) the trajectory $\boldsymbol{x} : [0, T] \to \Omega$ for a single agent starting at $\boldsymbol{x}_0 \in \Omega$. From there, we can define the trajectories for all agents $F : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ as $F(\boldsymbol{x}, t) = \boldsymbol{x}(t)$, where

$$\begin{aligned} &\mathrm{d}\boldsymbol{x}(t) = \boldsymbol{v}(\boldsymbol{x}(t), t)\mathrm{d}t, \quad \forall t \in [0, T] \\ &\boldsymbol{x}(0) = \boldsymbol{x}. \end{aligned} \tag{A.1}$$

We see that $\partial_t F(\boldsymbol{x}, t) = \mathrm{d}\boldsymbol{x}(t) = \boldsymbol{v}(\boldsymbol{x}(t), t) = v(F(\boldsymbol{x}, t), t), \forall t \in [0, T]$, and $F(\boldsymbol{x}, 0) = \boldsymbol{x}(0) = \boldsymbol{x}$. This gives the ODE system governing $F(\boldsymbol{x}, t)$.

Next, we prove the additive property of $\Phi_{t_i}^{t_j}$.

Proposition A.1. *For any $0 \le t_a \le t_b \le t_c \le T$, we have $\Phi_{t_b}^{t_c} \circ \Phi_{t_a}^{t_b} = \Phi_{t_a}^{t_c}$.*

Proof. By directly integrating the ODE governing $\boldsymbol{x}(t)$,

$$\begin{aligned} &\Phi_{t_a}^{t_b}(\boldsymbol{x}_0) = X(t_b) = \boldsymbol{x}_0 + \int_{t_a}^{t_b} \boldsymbol{v}(\boldsymbol{x}(s), s)\mathrm{d}s \\ \Longrightarrow\ &\Phi_{t_b}^{t_c}(\Phi_{t_a}^{t_b}(\boldsymbol{x}_0)) = (\boldsymbol{x}_0 + \int_{t_a}^{t_b} \boldsymbol{v}(\boldsymbol{x}(s), s)\mathrm{d}s) + \int_{t_b}^{t_c} \boldsymbol{v}(\boldsymbol{x}(s), s)\mathrm{d}s \\ &\qquad = \boldsymbol{x}_0 + \int_{t_a}^{t_c} \boldsymbol{v}(\boldsymbol{x}(s), s)\mathrm{d}s \\ &\qquad = \Phi_{t_a}^{t_c}(\boldsymbol{x}_0). \quad \square \end{aligned}$$

Now, recalling the definition of $\Phi_{t_i}^{t_j}(\boldsymbol{x})$ in (17), we see that $F_k(\boldsymbol{x}) := F(x, t_k) = \Phi_0^{t_k}(\boldsymbol{x}) = \Phi_{t_0}^{t_k}(\boldsymbol{x})$, which by the additive property can be decomposed iteratively: $F_k = \Phi_{t_0}^{t_k} = \Phi_{t_{k-1}}^{t_k} \circ \Phi_{t_0}^{t_{k-1}} = \Phi_{t_{k-1}}^{t_k} \circ \Phi_{t_{k-2}}^{t_{k-1}} \circ \cdots \circ \Phi_{t_0}^{t_1}$.

Appendix B. Derivation of the alternative formulation

Consider reverse agent trajectories $\mathcal{G} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ satisfying:

$$\begin{aligned} \partial_t \mathcal{G}(x, 1-t) &= v(\mathcal{G}(x, 1-t), t), \quad \boldsymbol{x} \in \Omega, t \in [0, T] \\ \mathcal{G}(x, 0) &= x, \quad \boldsymbol{x} \in \Omega \\ \mathcal{G}(\cdot, t)_* P_1 &= P(\cdot, 1-t), \forall t \in [0, 1]. \end{aligned} \tag{B.1}$$

Applying the change of measure, the transport cost becomes:

$$\begin{aligned} \int_0^1 \int_\Omega \langle \boldsymbol{v}(\boldsymbol{x}, t), \boldsymbol{v}(\boldsymbol{x}, t) \rangle p(\boldsymbol{x}, t) \mathrm{d}\boldsymbol{x} \mathrm{d}t &= \int_0^1 \int_\Omega \langle v(\mathcal{G}(x, 1-t), t), v(\mathcal{G}(x, 1-t), t) \rangle p_1(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \mathrm{d}t \\ &= \int_0^1 \int_\Omega \langle \partial_t \mathcal{G}(x, 1-t), \partial \mathcal{G}(x, 1-t) \rangle p_1(\boldsymbol{x}) \mathrm{d}x \mathrm{d}t \\ &= \mathbb{E}_{x \sim P_1} \left[\int_0^1 \| \partial_t \mathcal{G}(x, 1-t) \|_2^2 \right] \mathrm{d}t. \end{aligned}$$

Defining a regular grid in time: $t_k := k \cdot \Delta t, \forall k = 0, 1, \ldots K,\ \Delta t := \frac{1}{K}$, we obtain the approximation:

$$\partial_t \mathcal{G}(x, 1-t_k) \approx \frac{G_{K-k}(\boldsymbol{x}) - G_{K-k-1}(\boldsymbol{x})}{\Delta t}, \forall k = 0, 1, .., K-1.$$

Therefore, the discretized transport cost is:

$$\begin{aligned} \mathbb{E}_{x \sim P_1} \left[\int_0^1 \| \partial_t \mathcal{G}(x, 1-t) \|_2^2 \right] \mathrm{d}t &= \mathbb{E}_{x \sim P_1} \left[\sum_{k=0}^{K-1} \left\| \frac{G_{K-k}(\boldsymbol{x}) - G_{K-k-1}(\boldsymbol{x})}{\Delta t} \right\|_2^2 \Delta t \right] \\ &= K \cdot \mathbb{E}_{x \sim P_1} \left[\sum_{k=0}^{K-1} \| G_{k+1}(\boldsymbol{x}) - G_k(\boldsymbol{x}) \|_2^2 \right]. \end{aligned}$$

Appendix C. Proofs

In this section, we provide the proofs of the statements mentioned in Section 6.

C.1. Theorem 6.1

The proof essentially follows the error analysis used in finite difference. Define a regular grid in time with grid points $t_i := i \cdot \Delta t, \forall i = 0, 1, \ldots K$ and grid spacing $\Delta t := \frac{1}{K}$. Let $\hat{L}(t) = \int_{\mathbb{R}^d} \| \partial_t F(\boldsymbol{z}, t) \|_2^2 p_0(\boldsymbol{z}) \mathrm{d}\boldsymbol{z} := \| \partial_t F(\cdot, t) \|_{P_0}^2$. We start with the continuous objective

$$\begin{aligned} \int_0^1 \int_{\mathbb{R}^d} \| \partial_t F(\boldsymbol{z}, t) \|_2^2 p_0(\boldsymbol{z}) \det \boldsymbol{z} \mathrm{d}t &:= \int_0^1 \| \partial_t F(\cdot, t) \|_{P_0}^2 \mathrm{d}t \\ &= \sum_{i=0}^{K-1} \int_{t_i}^{t_{i+1}} \hat{L}(t) \mathrm{d}t \\ &= \sum_{i=0}^{K-1} \int_{t_i}^{t_{i+1}} \left[\hat{L}(t_i) + (t - t_i) \partial_t \hat{L}(t_i) + \frac{1}{2} (t - t_i)^2 \partial_{tt} \hat{L}(t_i) + \cdots \right] \mathrm{d}t \\ &= \sum_{i=0}^{K-1} \hat{L}(t_i) \Delta t + \partial_t \hat{L}(t_i) \Delta t^2 + \frac{1}{6} \partial_{tt} \hat{L}(t_i) \Delta t^3 + O(\Delta t^4). \end{aligned} \tag{C.1}$$

To obtain $\hat{L}(t_i)$, we use the forward difference:

$$\begin{aligned}
\partial_t F(x,t_i) &= \frac{F_{i+1}(\boldsymbol{x}) - F_i(\boldsymbol{x})}{\Delta t} - \frac{1}{2}\Delta t \partial_{tt} F_i(\boldsymbol{x}) + O(\Delta t^2) \\
\Longrightarrow \hat{L}(t_i) &= \left\| \frac{F_{i+1} - F_i}{\Delta t} - \frac{1}{2}\Delta t \partial_{tt} F_i + O(\Delta t^2) \right\|_{P_0}^2 \\
&= \left\| \frac{F_{i+1} - F_i}{\Delta t} \right\|_{P_0}^2 - \left\langle \frac{F_{i+1} - F_i}{\Delta t}, \partial_{tt} F_i \right\rangle_{P_0} \Delta t + O(\Delta t^2),
\end{aligned}$$

where $F_i(\boldsymbol{x}) := F(x, t_i)$. Therefore, if we approximate $\hat{L}(t_i) \approx \| \frac{F_{i+1}-F_i}{\Delta t} \|_{P_0}^2$, the error of the integral is:

$$\begin{aligned}
&\sum_{i=0}^{K-1} \left[\left\| \frac{F_{i+1} - F_i}{\Delta t} \right\|_{P_0}^2 - \left\| \frac{F_{i+1} - F_i}{\Delta t} \right\|_{P_0}^2 + \left\langle \frac{F_{i+1} - F_i}{\Delta t}, \partial_{tt} F_i \right\rangle_{P_0} \Delta t + O(\Delta t^2) \right] \Delta t + \partial_t \hat{L}(t_i) \Delta t^2 + O(\Delta t^3) \\
&= \sum_{i=0}^{K-1} \left[\left\langle \frac{F_{i+1} - F_i}{\Delta t}, \partial_{tt} F_i \right\rangle_{P_0} + \partial_t \hat{L}(t_i) \right] \Delta t^2 + O(\Delta t^3) \\
&= \sum_{i=0}^{K-1} [\langle \partial_t F_i + O(\Delta t), \partial_{tt} F_i \rangle_{P_0} + \partial_t \hat{L}(t_i)] \Delta t^2 + O(\Delta t^3) \\
&= \sum_{i=0}^{K-1} [\langle \partial_t F_i, \partial_{tt} F_i \rangle_{P_0} + \partial_t \hat{L}(t_i)] \Delta t^2 + O(\Delta t^3) \\
&= O(\Delta t),
\end{aligned} \tag{C.2}$$

where the last equality is a result of summing over K grid points.

C.2. Lemma 6.2

First, note that the MFG problem (14) admits the same optimizer and objective value if we add $F(\boldsymbol{z}, 1) = F_1(\boldsymbol{z})$ as an additional constraint:

$$\begin{aligned}
&\inf_F \quad \lambda_L \int_0^1 \int_{\mathbb{R}^n} p_0(\boldsymbol{z}) \|\partial_t F(\boldsymbol{z}, t)\|_2^2 \mathrm{d}z\mathrm{d}t + \mathcal{M}(F(\cdot, 1)_* p_0) \\
&s.t. \quad F(\boldsymbol{z}, 0) = x,\ F(\boldsymbol{z}, 1) = F_1(\boldsymbol{z}).
\end{aligned} \tag{C.3}$$

Note that $\mathcal{M}(F(\cdot, 1)_* p_0)$ is a constant with respect to F. Therefore, the above problem has the same solution as:

$$\begin{aligned}
&\inf_F \quad \lambda_L \int_0^1 \int_{\mathbb{R}^n} p_0(\boldsymbol{z}) \|\partial_t F(\boldsymbol{z}, t)\|_2^2 \mathrm{d}z\mathrm{d}t \\
&s.t. \quad F(\boldsymbol{z}, 0) = x,\ F(\boldsymbol{z}, 1) = F_1(\boldsymbol{z}),
\end{aligned} \tag{C.4}$$

which in turn is equivalent to the following OT problem:

$$\begin{aligned}
&\inf_{p, \boldsymbol{v}} \quad \lambda_L \int_0^1 \int_{\mathbb{R}^n} p(\boldsymbol{x}, t) \|\boldsymbol{v}(\boldsymbol{x}, t)\|_2^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \\
&s.t. \quad p(\boldsymbol{x}, 0) = p_0(\boldsymbol{x}),\ p(\boldsymbol{x}, 1) = F_{1*} p_0(\boldsymbol{x}),
\end{aligned} \tag{C.5}$$

where $F_{1*} p_0(\boldsymbol{x}) = p_0(F_1^{-1}(\boldsymbol{x}))|\det \nabla F_1^{-1}(\boldsymbol{x})|$. By the OT theory, we know that there exists a unique Monge map $T(\boldsymbol{x})$ such that $p(\boldsymbol{x}, 1) = T_* p_0(\boldsymbol{x})$ [24]. Therefore, $T = F_1$ and the optimizer F^* of (C.4) satisfies $F^*(\boldsymbol{z}, t) = (1-t)z + tT(\boldsymbol{z}) = (1-t)\boldsymbol{z} + tF_1(\boldsymbol{z})$. As noted before, F^* minimizes the continuous MFG problem (14) as well.

C.3. Lemma 6.3

Applying a similar logic to the proof of Lemma 6.2, we know that the optimizer of the discretized MFG problem (22) coincides with that of the following problem

$$\begin{aligned}
&\inf_{\{F_i\}_{i=0}^K} \quad \lambda_L \sum_{i=0}^{K-1} \mathbb{E}_{z\sim P_0}[\|F_{i+1}(\boldsymbol{z}) - F_i(\boldsymbol{z})\|_2^2] := \lambda_L \sum_{i=0}^{K-1} \|F_{i+1} - F_i\|_{P_0}^2 \\
&s.t. \quad F_0(\boldsymbol{z}) = \boldsymbol{z},\, F_K(\boldsymbol{z}) = F_1(\boldsymbol{z}),
\end{aligned}$$

where F_1 is an optimizer of the discretized MFG problem for F_K, and $\|f\|_{P_0}^2 := \int_{\mathbb{R}^d} f(\boldsymbol{x})^2 p_0(\boldsymbol{x})\mathrm{d}x$. Applying the KKT conditions on the above problem, we obtain:

$$\begin{aligned}
&\partial_{F_i}\lambda_L \sum_{i=0}^{K-1} \|F_{i+1} - F_i\|_{P_0}^2 = 0 \\
\implies & F_i^* = \frac{1}{2}(F_{i+1}^* + F_{i-1}^*), \quad \forall i = 1, 2, ..., K-1.
\end{aligned}$$

Along with $F_0^* = Id$, $F_K^* = F_1$, we see that the solution to this recurrence relation is $F_i^* = z(1-\frac{i}{K}) + F_1\frac{i}{K}, i = 0, 1, ..., K$.

C.4. Theorem 6.4

From Lemma 6.2, we know the optimizer of the continuous MFG problem (14) has the form

$$F^*(\boldsymbol{z}, t) = z(1-t) + T(\boldsymbol{x})t.$$

We can substitute this into the continuous MFG problem and optimize over T instead of F, yielding

$$\begin{aligned}
&\inf_T \quad \lambda_L \int_0^1 \int_{\mathbb{R}^d} p_0(\boldsymbol{z})\|\partial_t[z(1-t) + T(\boldsymbol{z})t)]\|_2^2 \mathrm{d}z\mathrm{d}t + \mathcal{M}(T_* p_0) \\
=&\inf_T \quad \lambda_L \int_0^1 \int_{\mathbb{R}^d} p_0(\boldsymbol{z})\|T(\boldsymbol{z}) - z\|_2^2 \mathrm{d}z\mathrm{d}t + \mathcal{M}(T_* p_0) \\
=&\inf_T \quad \lambda_L \int_{\mathbb{R}^d} p_0(\boldsymbol{z})\|T(\boldsymbol{z}) - z\|_2^2 \mathrm{d}z + \mathcal{M}(T_* p_0).
\end{aligned}$$

Second, we know from (6.3) the optimizer $\{F_i^*\}_{i=0}^K$ of the discretized MFG problem with K grid points has the form $F_i^* = z(1-\frac{i}{K}) + T_K\frac{i}{K}, i = 0, 1, ..., K$. As a result, we can rewrite the discretized MFG problem as

$$\begin{aligned}
&\inf_{T_K} \quad \lambda_L K \cdot \mathbb{E}_{z\sim P_0}\left[\sum_{k=0}^{K-1} \left\| z\left(1 - \frac{k+1}{K}\right) + T_K(\boldsymbol{z})\frac{k+1}{K} - z\left(1-\frac{k}{K}\right) - T_K(\boldsymbol{z})\frac{k}{K}\right\|_2^2\right] + \mathcal{M}(T_{K_*}p_0) \\
=&\inf_{T_K} \quad \lambda_L K \cdot \sum_{k=0}^{K-1} \int_{\mathbb{R}^d} p_0(\boldsymbol{z}) \left\| -\frac{1}{K}z + \frac{1}{K}T_K(\boldsymbol{z})\right\|_2^2 \mathrm{d}z + \mathcal{M}(T_{K_*}p_0) \\
=&\inf_{T_K} \quad \lambda_L \cdot \int_{\mathbb{R}^d} p_0(\boldsymbol{z})\|T_K(\boldsymbol{z}) - z\|_2^2 \mathrm{d}z + \mathcal{M}(T_{K_*}p_0),
\end{aligned}$$

which is identical to the problem over T for the continuous MFG. Therefore, their objective values agree; furthermore, the minimizers of the two transformed problems are also the same: $T = T_K$. As a result, the mapping at the terminal time agrees between the continuous and the discretized problem: $F^*(\boldsymbol{z}, 1) = F_K^*(\boldsymbol{z})$. It then follows that the linear interpolation between the initial and terminal mappings in the discretized problem, $z(1-t) + F_K^*(\boldsymbol{z})t$, is precisely the unique optimizer for the continuous problem.

C.5. *Theorem 6.7*

The proof can be obtained by combining Theorem 6.6 with the following lemma.

Lemma C.1. *Let P_0 be absolutely continuous with a bounded density p_0. Suppose for any compact $A \subset \mathbb{R}^{\mathbb{d}}$, there is a sequence of mappings $\{F_A^n\}_{n=1}^\infty$ such that $\|F_A^n - F\|_{1,A} \to 0$. Then, there exists $\{F^n\}_{n=1}^\infty$ such that $F_*^n P_0 \xrightarrow{d} F_* P_0$, where $\xrightarrow{d}$ denotes convergence in distribution.*

Proof. Define

$$\mathbf{BL_1} = \{\eta : \eta \text{ bounded and Lipschitz with } Lip(\eta) + \|\eta\|_\infty \le 1\}.$$

According to [60], to show $F_*^n P_0 \xrightarrow{d} F_* P_0$, it suffices to prove that:

$$\sup_{\eta \in \mathbf{BL_1}} |\mathbb{E}_{F_*^n P_0}[\eta] - \mathbb{E}_{F_* P_0}[\eta]| < \frac{1}{n}, \quad \forall n \in \mathbb{N}.$$

For each $n \in \mathbb{N}$, pick a compact $A_n \subset \mathbb{R}^d$ such that $P_0(A_n^c) = P_0(\mathbb{R}^d) - P_0(A_n) = 1 - P_0(A_n) < \frac{1}{2n}$. By assumption, we can find a sequence $\{F_n^k\}_{k=1}^\infty$ satisfying $\|F_n^k - F\|_{1,A_n} < \dfrac{1}{k\|p_0\|_{\infty,A_n}}$. Construct the sequence $\{F^n\}_{n=1}^\infty := \{F_n^n\}_{n=1}^\infty$, which satisfies $\|F^n - F\|_{1,A_n} < \dfrac{1}{n\|p_0\|_{\infty,A_n}}, \forall n \in \mathbb{N}$.

For any $\eta \in \mathbf{BL_1}$, we have:

$$\begin{aligned}
|\mathbb{E}_{F_*^n P_0}[\eta] - \mathbb{E}_{F_* P_0}[\eta]| &= \left| \int_{\mathbb{R}^d} \eta \mathrm{d}F_*^n P_0 - \eta \mathrm{d}F_* P_0 \right| \\
&= \left| \int_{\mathbb{R}^d} \eta \circ F^n \mathrm{d}P_0 - \eta \circ F \mathrm{d}P_0 \right| \\
&= \left| \int_{A_n^c} \eta \circ F^n \mathrm{d}P_0 + \int_{A_n} \eta \circ F^n \mathrm{d}P_0 - \int_{A_n^c} \eta \circ F \mathrm{d}P_0 - \int_{A_n} \eta \circ F \mathrm{d}P_0 \right| \\
&\le \left| \int_{A_n^c} \eta \circ F^n \mathrm{d}P_0 \right| + \left| \int_{A_n^c} \eta \circ F \mathrm{d}P_0 \right| + \left| \int_{A_n} \eta \circ F^n \mathrm{d}P_0 - \eta \circ F \mathrm{d}P_0 \right| \\
&\le \int_{A_n^c} |\eta \circ F^n| \mathrm{d}P_0 + \int_{A_n^c} |\eta \circ F| \mathrm{d}P_0 + \int_{A_n} |\eta(F^n(\boldsymbol{x})) - \eta(F(\boldsymbol{x}))| \mathrm{d}P_0 \\
&\le \|\eta\|_\infty P_0(A_n^c) + \|\eta\|_\infty P_0(A_n^c) + Lip(\eta) \int_A \|F^n(\boldsymbol{x}) - F(\boldsymbol{x})\|_1 \mathrm{d}P_0 \\
&\le 2\|\eta\|_\infty P_0(A_n^c) + Lip(\eta)\|p_0\|_{\infty,A_n} \int_{A_n} \|F^n(\boldsymbol{x}) - F(\boldsymbol{x})\|_1 \mathrm{d}x \\
&< \|\eta\|_\infty \frac{1}{n} + Lip(\eta)\frac{1}{n} \\
&= (\|\eta\|_\infty + Lip(\eta))\frac{1}{n} \le \frac{1}{n}. \quad \square
\end{aligned}$$

Appendix D. MFG experiments

We use NSF-CL with identical hyperparameters to solve both the OT and the crowd motion problems across different dimensions. The NF model contains 10 flows, each consisting of an alternating block with rational-quadratic splines as the coupling function, followed by a linear transformation. The conditioner has 256 hidden features, 8 bins, 2 transform blocks, ReLU activation, and a dropout probability of 0.25.

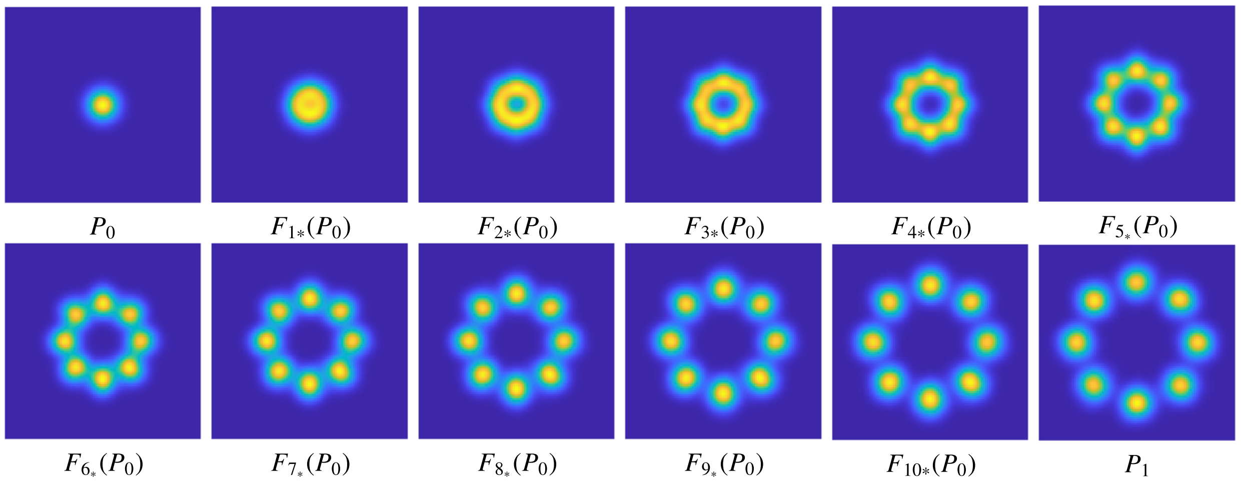


Fig. D.9. Evolution of the density for the OT problem (23) in 10D.

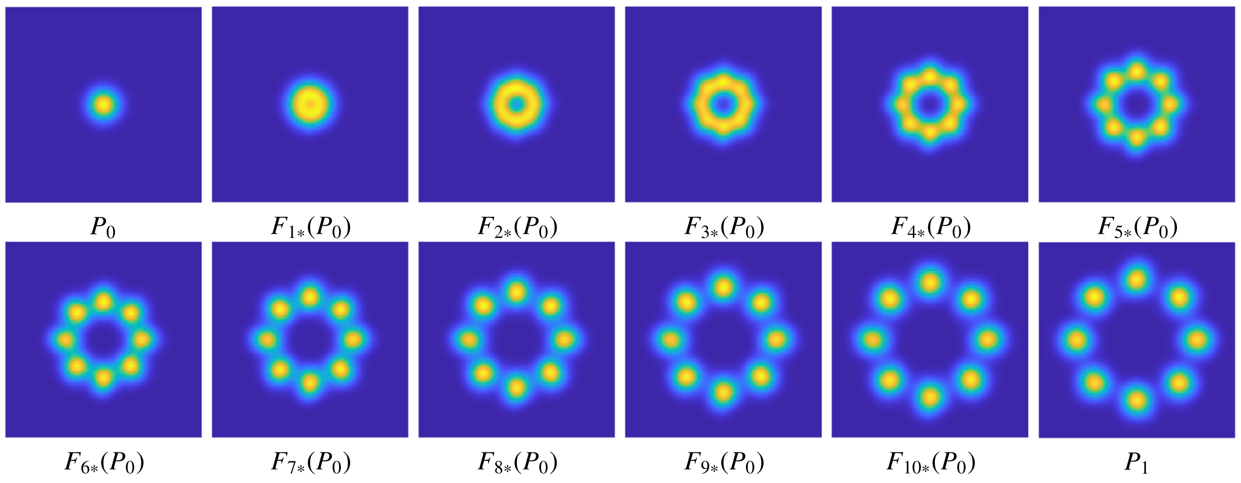


Fig. D.10. Evolution of the density for the OT problem (23) in 100D.

Table D.7
Results for crowd motion at different weights assigned to $\mathcal{I}$. $\lambda_{\mathcal{I}}$: weight for $\mathcal{I}$; L: transport cost; $\mathcal{I}$: interaction cost; $\mathcal{M}$: terminal cost.

$\lambda_{\mathcal{I}}$	L	$\mathcal{I}$	$\mathcal{M}$
0.2	32.8511 ± 0.0060	2.1274 ± 0.0114	0.0436 ± 0.0059
0.5	35.5121 ± 0.0134	1.3463 ± 0.0096	0.0424 ± 0.0058
1.0	39.9230 ± 0.0223	0.7076 ± 0.0053	0.0430 ± 0.0058

In both the OT and the crowd motion experiments, we use Jeffery's divergence for terminal matching, which is the symmetrized KL divergence: $\mathcal{M}(F(\cdot,T)_*P_0) = D_{KL}(P_1||F(\cdot,T)_*P_0) + D_{KL}(F(\cdot,T)_*P_0||P_1)$. This terminal cost is different from the typical negative log-likelihood loss in NF training, but it is still tractable because p_0, p_1 are known. The choice of the terminal cost is not unique. For example, one can use either forward or backward KL divergence. However, we find empirically that Jeffery's divergence yields the best results.

In the Monte Carlo approximation of the expectation, we resample a batch of data from the initial and the terminal distributions to compute the various loss terms at each iteration. Although [22] remarked that this approach yields slightly worse performance than resampling every 20 iterations, we find it to be good enough for our method.

The computed dynamics for the OT and the crowd motion experiments are approximately identical with respect to the dimensionality, so we select only one $d \in \{2, 10, 50, 100\}$ to show in the main text. For completeness, we include the remaining plots here.

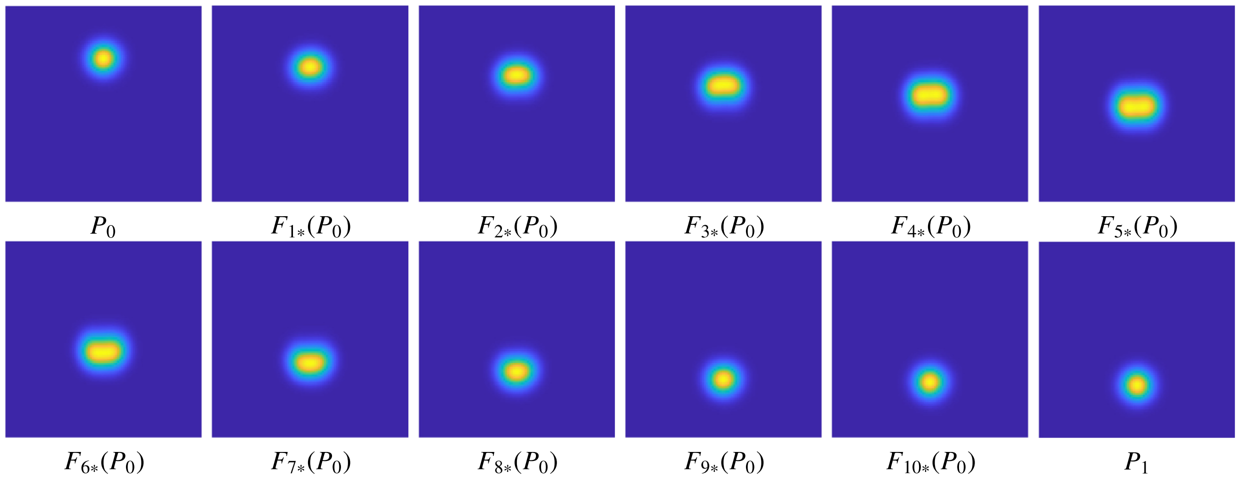


Fig. D.11. Evolution of the density for the crowd motion problem (24) in 2D.

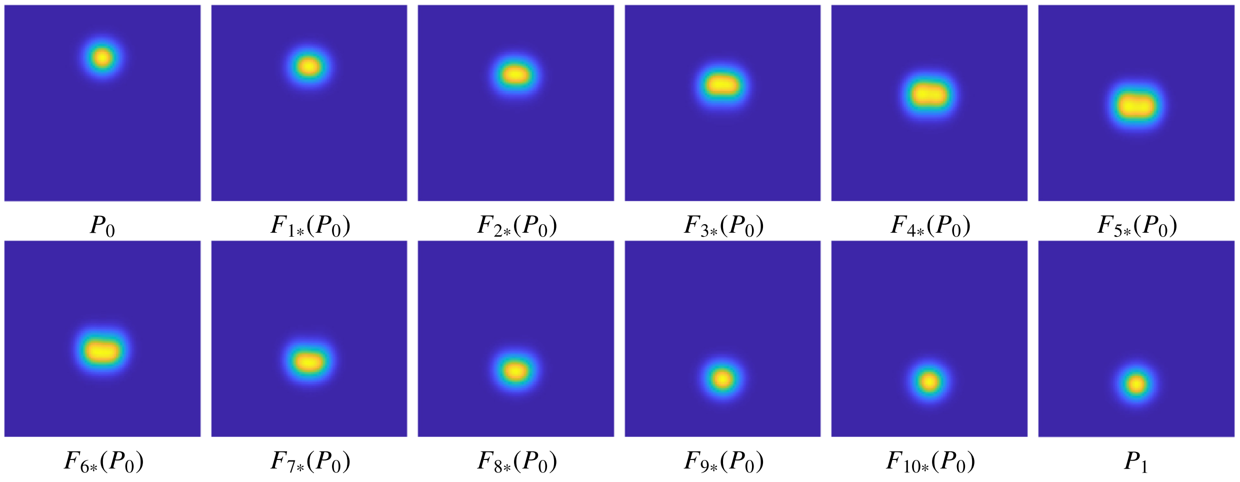


Fig. D.12. Evolution of the density for the crowd motion problem (24) in 10D.

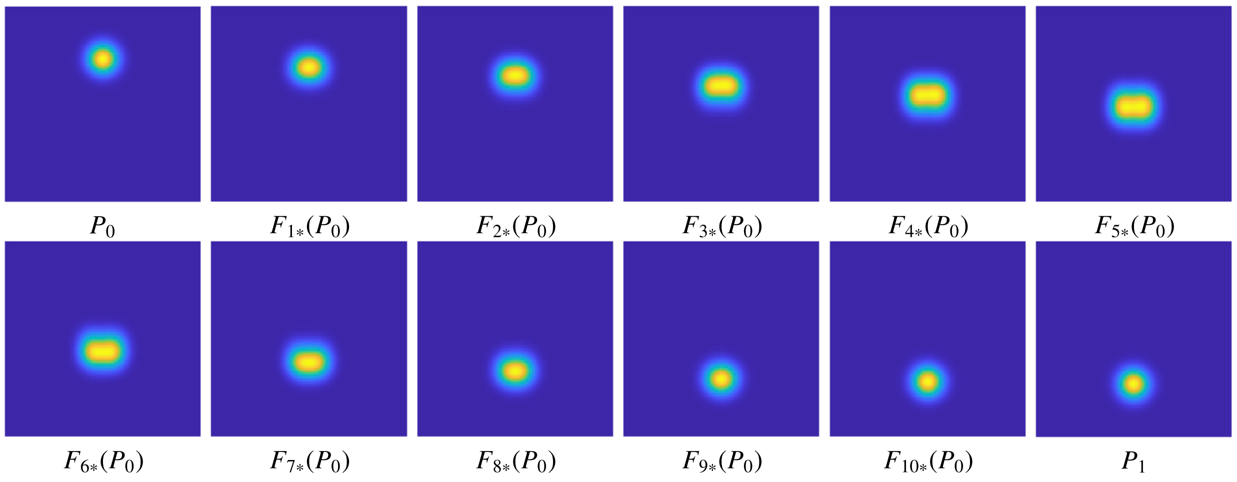


Fig. D.13. Evolution of the density for the crowd motion problem (24) in 50D.

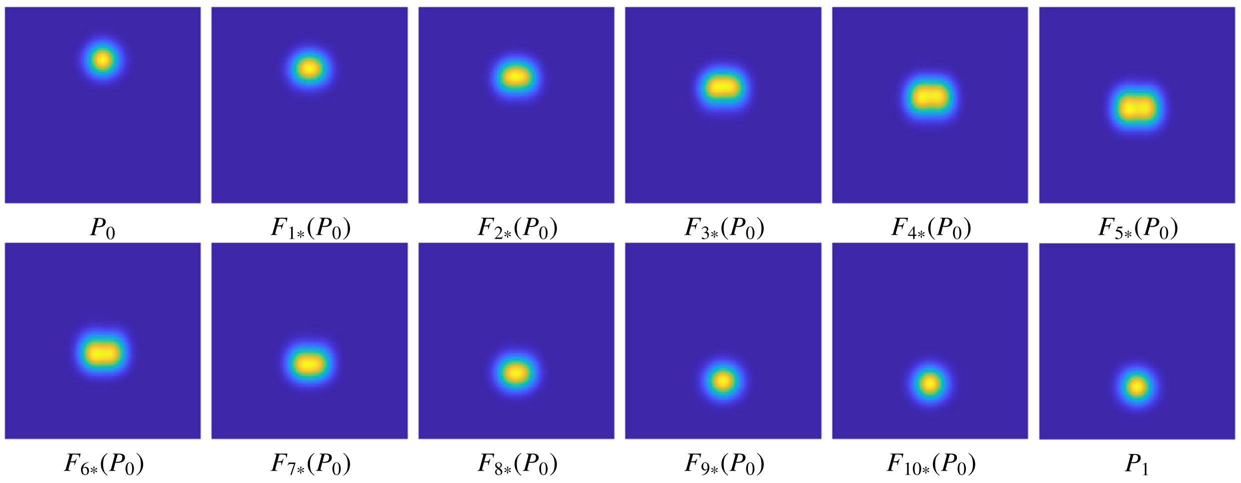


Fig. D.14. Evolution of the density for the crowd motion problem (24) in 100D.

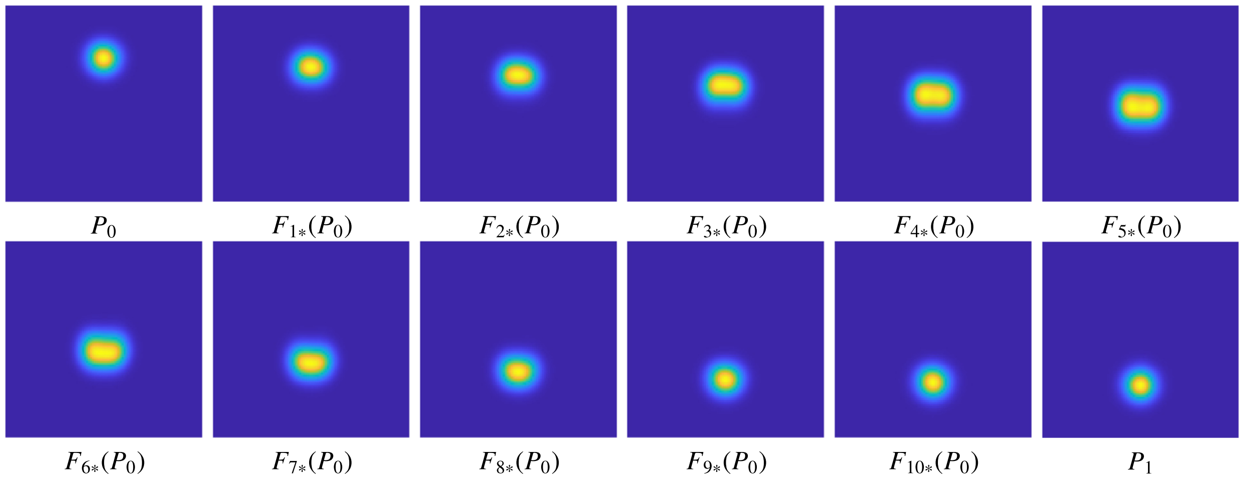


Fig. D.15. Evolution of the density for minor ($\lambda_{\mathcal{I}} = 0.2$) penalty on conflicts with the obstacle.

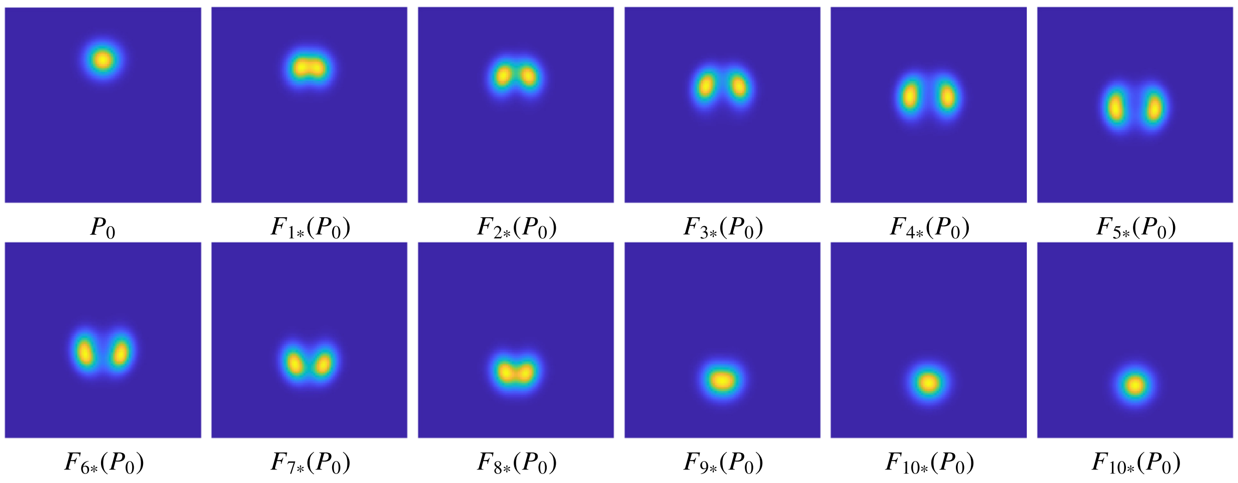


Fig. D.16. Evolution of the density for moderate ($\lambda_{\mathcal{I}} = 0.5$) penalty on conflicts with the obstacle.

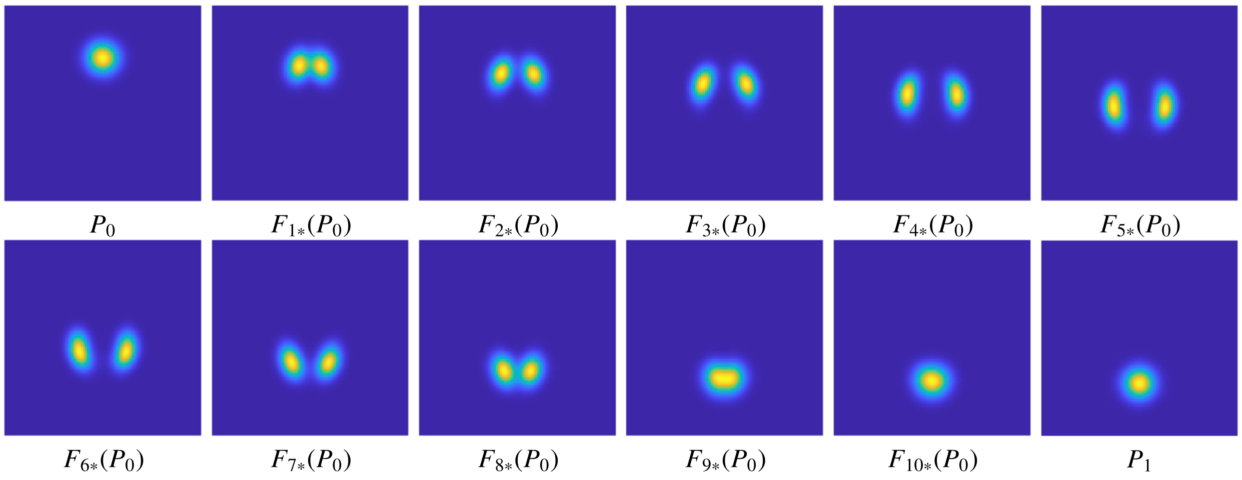


Fig. D.17. Evolution of the density for strong ($\lambda_{\mathcal{I}} = 1$) penalty on conflicts with the obstacle.

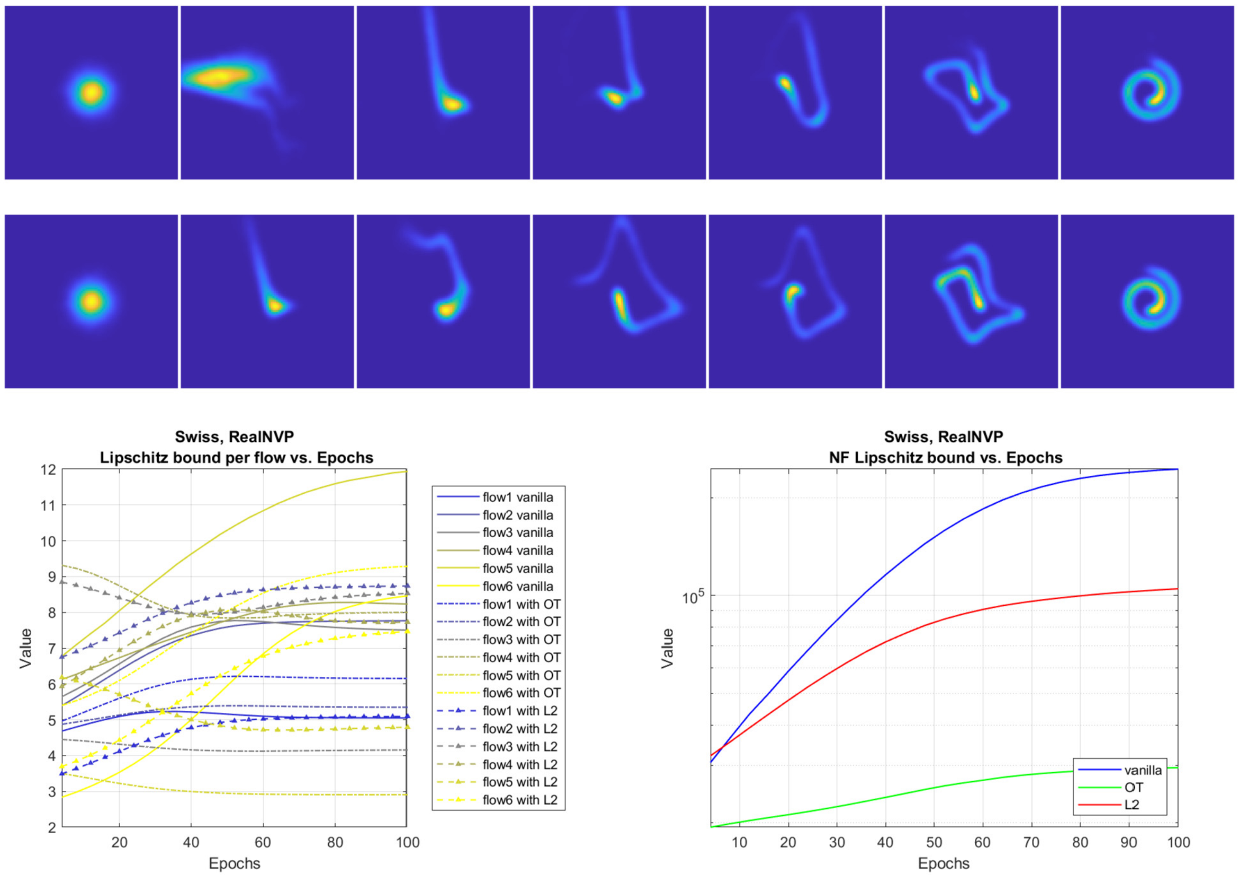


Fig. E.18. Top row: output of each intermediate RealNVP flow between a single Gaussian and the Swiss roll-shape density, trained without using the transport cost. Middle row: same but trained with using the transport cost. Bottom left: the Lipschitz bound for each flow over the training epochs. Bottom right: the Lipschitz bound for the entire flow. The weights of the transport regularization are chosen so that the negative log-likelihood is not severely obstructed.

D.1. Optimal transport

We show the density evolution and the sample trajectories for $d \in \{10, 100\}$ here (see Figs. D.9,D.10); for the case $d = 50$, see Fig. 2 in the main text. Visually, the evolutions are approximately invariant with respect to the dimensionality.

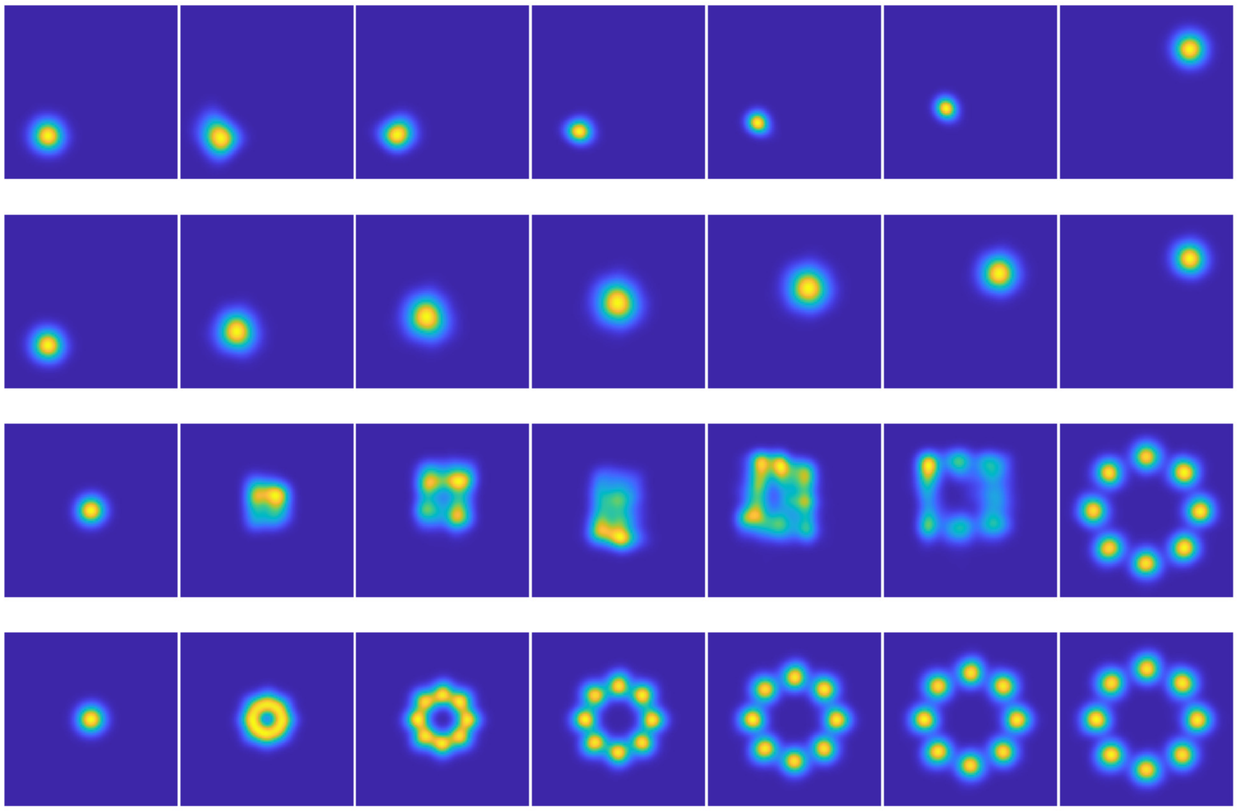


Fig. E.19. Rows 1 and 2: output of each intermediate RealNVP flow between two Gaussians, trained without and with using the transport cost, respectively. Rows 3 and 4: output of each intermediate NSF-CL flow between a Gaussian and a Gaussian mixture, trained without and with using the transport cost, respectively.

D.2. Crowd motion

D.2.1. Density evolution in different dimensions

We show the density evolution for $d \in \{2, 10, 50, 100\}$ in Figs. D.11–D.14, which further supports our method's robustness with respect to dimensionality.

D.2.2. Different levels of penalty on conflicts with the obstacle

In addition to the sampled trajectories provided in Fig. 4, we show the density evolution as well as the computed cost values for different aversion preferences here. All experiments are conducted at $d = 10$ with identical hyperparameter settings as the one used in Table 2, except for the weights on the cost terms. The level of aversion is manifested through the outward-curving behavior when the densities are transported near the origin. See Table D.7 and Figs. D.15–D.17.

Appendix E. NF experiments

E.1. More synthetic data

See Figs. E.18–E.20.

E.2. Tabular datasets

For the RealNVP flows, we used a reimplementation provided by [61], which achieved better log-likelihoods than the results reported in the original paper [28]. All RealNVP models use 6 flows with hidden dimensions of 256 in the s, t networks. For NSF, we used the original implementation [29] with the reported hyperparameters in their paper. The likelihood results for the other models are taken from Table 3 of [23]. Our Lipschitz bound for a flow f is defined as $\max_{\boldsymbol{x} \in P_1} \|\nabla f(\boldsymbol{x})\|_2$, the estimated maximum gradient spectral norm over the training set P_1. The Lipschitz bound for the entire NF is the product of the bounds of the individual flows. See Fig. E.21.

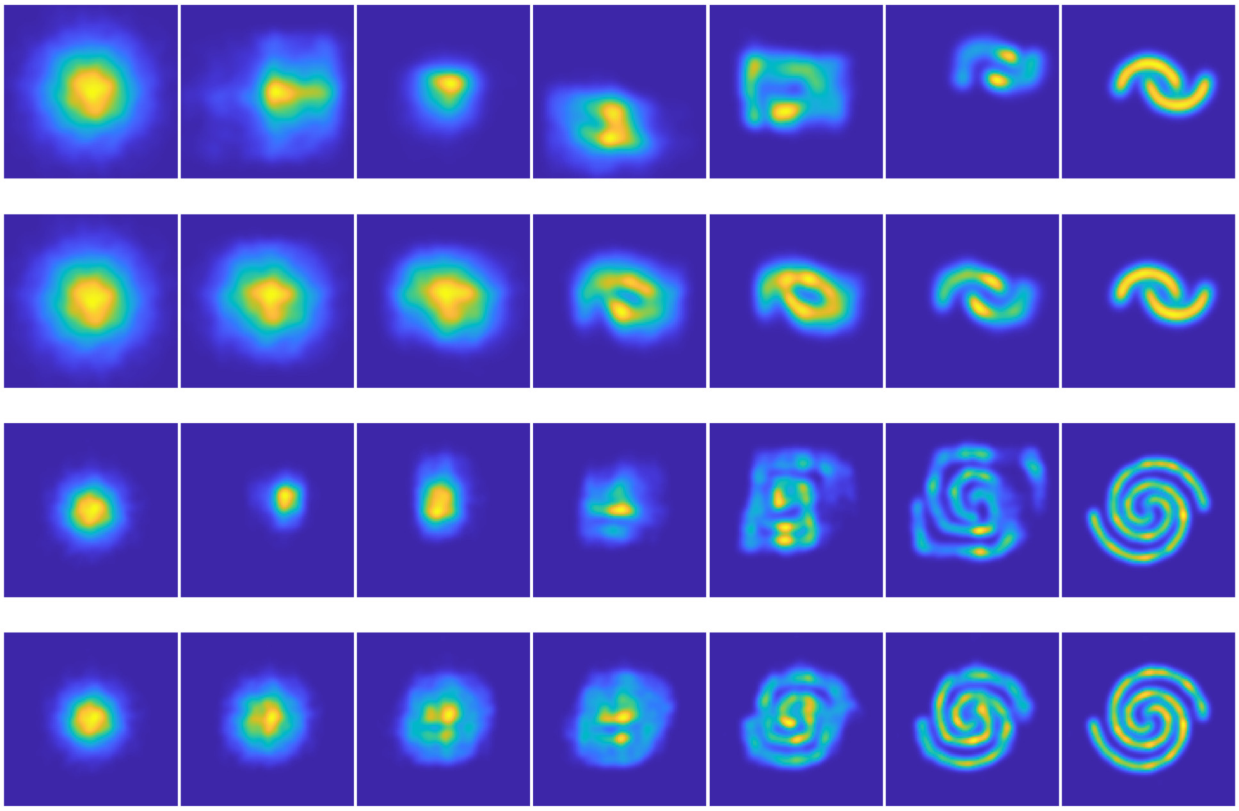
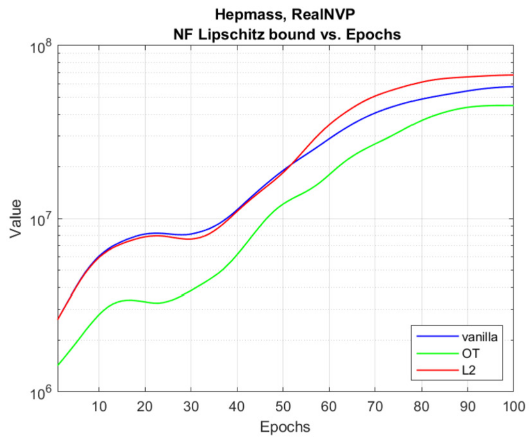


Fig. E.20. Rows 1 and 2: output of each intermediate RealNVP flow between two Gaussians, trained without and with using the transport cost, respectively. Rows 3 and 4: output of each intermediate NSF-CL flow between a Gaussian and the spiral density, trained without and with using the transport cost, respectively.



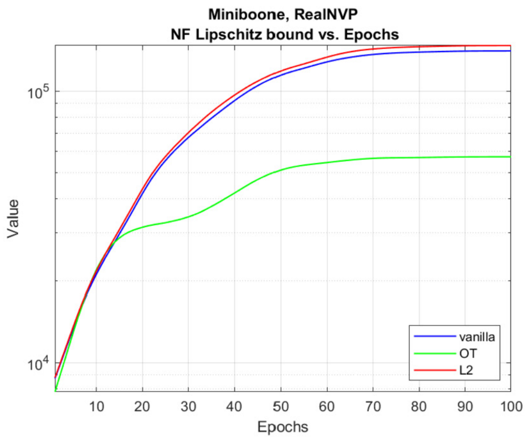


Fig. E.21. The Lipschitz bound over training epochs, for different tabular datasets (left to right: Hepmass and Minibone).

For a controlled study, all hyperparameters for the standard RealNVP (resp. NSF) and the RealNVP (resp. NSF) trained with transport costs are identical. The normalized weights $\frac{\lambda_L}{\lambda_{\mathcal{M}}}$ for the transport costs used to produce the results in Table 5 are summarized in Table E.8.

The dimension-wise permutations have shown to improve the flexibility of flow transforms both theoretically [25] and empirically [50]. When transport costs are computed, we omit the costs of permutation transforms, since the underlying object stays the same.

In the main text, we showed the marginal distributions on the first two dimensions for RealNVP trained on Miniboone. The remaining dimensions follow a similar trend. We additionally show those for dimensions 3,4 and for dimensions 5,6 here. See Figs. E.22 and E.23.

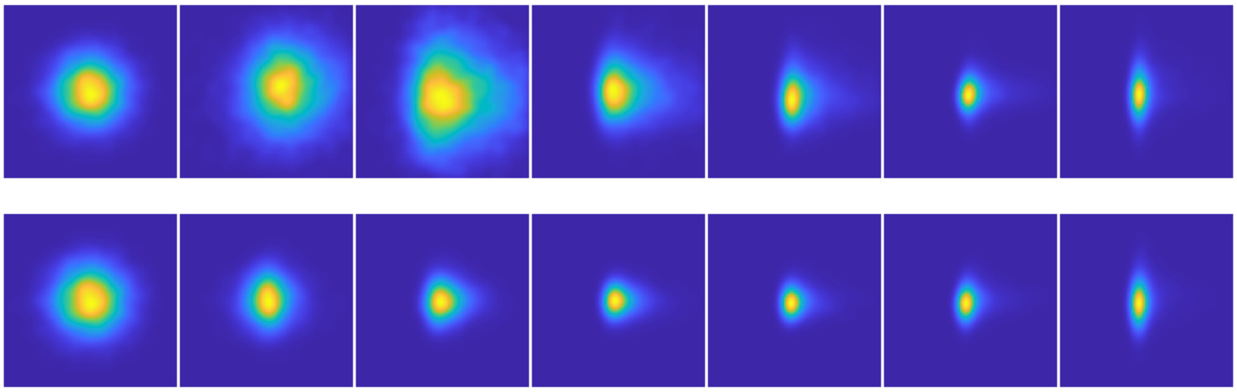


Fig. E.22. Output of each intermediate RealNVP flow, projected onto dimensions 3 and 4, between a Gaussian and the Miniboone dataset. Top: trained without using the transport cost; bottom: with the cost.

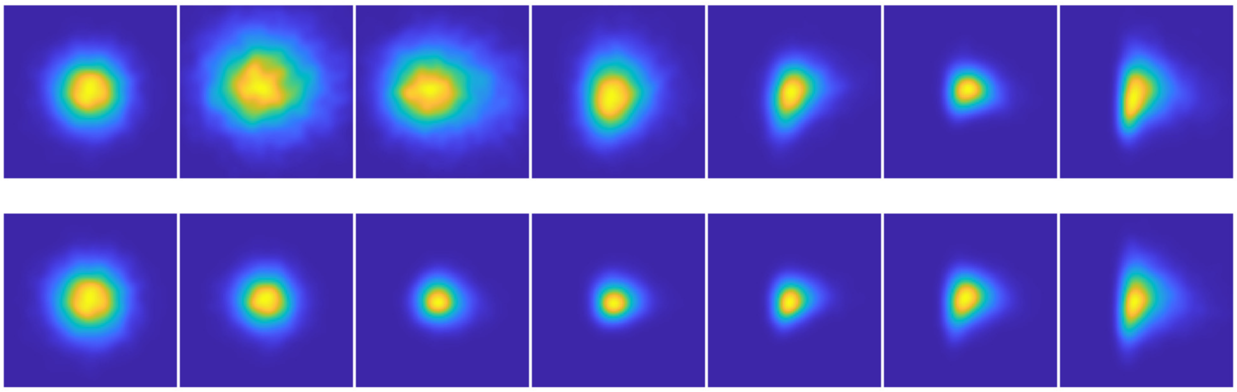


Fig. E.23. Output of each intermediate RealNVP flow, projected onto dimensions 5 and 6, between a Gaussian and the Miniboone dataset. Top: trained without using the transport cost; bottom: with the cost.

Table E.8
Normalized weights associated with the transport cost ($\frac{\lambda_L}{\lambda_{\mathcal{M}}}$) used to produce results in Table 5.

	Power	Gas	Hepmass	Miniboone	BSDS300
RNVP OT Weight	5e-6	1e-6	5e-5	5e-3	1e-5
NSF OT Weight	1e-5	5e-5	5e-5	5e-3	5e-2

Table E.9
Meta-data for the image datasets used in Table 6.

	MNIST	FMNIST	CIFAR-10	SVHN	EUROSAT
Number of Images	60000	60000	50000	73257	27000
Image Dimensions	28×28	28×28	32×32	32×32	64×64
Number of Channels	1	1	3	3	3

Table E.10
Hyperparameters used to produce the results in Table 6.

	MNIST	FMNIST	CIFAR-10	SVHN	EUROSAT
Glow OT Weight	1e-6	1e-6	5e-7	3e-7	3e-7
Number of levels	2	2	4	4	4

E.3. Image datasets

Here, we document the details of the image datasets as well as the Glow model. The SVHN dataset consists of color images of house numbers [62], and EuroSAT is composed of satellite images for various landscapes [63]. In Table E.9, we provide the meta-data for the image datasets used.

The Glow implementation we used is adapted from [64], which uses a simple CNN with 512 channels interlaced with actnorm as the conditioner in its affine coupling transforms. Each flow in Glow consists of an actnorm layer, an affine coupling, and a 1x1 convolution [50]. The full model employs 32 flow transforms per level, and the number of levels depends on the image dataset specified in Table E.10. At the end of each level, the images are squeezed from the shape $c \times n \times n$ to $4c \times \frac{n}{2} \times \frac{n}{2}$, where c, n are the number of channels and the side dimension, respectively. Afterwards, half of the channels go through an additional affine transformation and are not transformed further in the remaining flows. We optimize the models with Adam [53] at the highest learning rate where training does not diverge. Further information about the Glow model is summarized in Table E.10.

References

[1] O. Guéant, J.-M. Lasry, P.-L. Lions, Mean field games and applications, in: Paris-Princeton Lectures on Mathematical Finance 2010, Springer, 2011, pp. 205–266.
[2] X. Guo, A. Hu, R. Xu, J. Zhang, Learning mean-field games, Adv. Neural Inf. Process. Syst. 32 (2019).
[3] Y. Achdou, F.J. Buera, J.-M. Lasry, P.-L. Lions, B. Moll, Partial differential equation models in macroeconomics, Philos. Trans. R. Soc. A, Math. Phys. Eng. Sci. 372 (2014) 20130397.
[4] Y. Achdou, J. Han, J.-M. Lasry, P.-L. Lions, B. Moll, Income and wealth distribution in macroeconomics: a continuous-time approach, Rev. Econ. Stud. 89 (2022) 45–86.
[5] A. Lachapelle, J.-M. Lasry, C.-A. Lehalle, P.-L. Lions, Efficiency of the price formation process in presence of high frequency participants: a mean field game analysis, Math. Financ. Econ. 10 (2016) 223–262.
[6] P. Cardaliaguet, C.-A. Lehalle, Mean field game of controls and an application to trade crowding, Math. Financ. Econ. 12 (2018) 335–363.
[7] D. Firoozi, P.E. Caines, An optimal execution problem in finance targeting the market trading speed: an mfg formulation, in: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), IEEE, 2017, pp. 7–14.
[8] Z. Liu, B. Wu, H. Lin, A mean field game approach to swarming robots control, in: 2018 Annual American Control Conference (ACC), IEEE, 2018, pp. 4293–4298.
[9] Y. Jiang, Y. Hu, M. Bennis, F.-C. Zheng, X. You, A mean field game-based distributed edge caching in fog radio access networks, IEEE Trans. Commun. 68 (2019) 1567–1580.
[10] Y. Zhang, H. Zhang, K. Long, Energy efficient resource allocation in cache based terahertz vehicular networks: a mean-field game approach, IEEE Trans. Veh. Technol. 70 (2021) 5275–5285.
[11] A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, Int. J. Comput. Vis. 42 (2001) 145–175.
[12] S. Kolouri, S.R. Park, M. Thorpe, D. Slepcev, G.K. Rohde, Optimal mass transport: signal processing and machine-learning applications, IEEE Signal Process. Mag. 34 (2017) 43–59.
[13] M. Kusner, Y. Sun, N. Kolkin, K. Weinberger, From word embeddings to document distances, in: International Conference on Machine Learning, PMLR, 2015, pp. 957–966.
[14] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 214–223.
[15] Y. Achdou, I. Capuzzo-Dolcetta, Mean field games: numerical methods, SIAM J. Numer. Anal. 48 (2010) 1136–1162.
[16] J.-D. Benamou, G. Carlier, Augmented lagrangian methods for transport optimization, mean-field games and degenerate pdes, J. Optim. Theory Appl. 167 (2015) 1–26.
[17] J.-D. Benamou, G. Carlier, F. Santambrogio, Variational mean field games, in: Active Particles, vol. 1, Springer, 2017, pp. 141–171.
[18] J.-D. Benamou, Y. Brenier, A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem, Numer. Math. 84 (2000) 375–393.
[19] M. Jacobs, F. Léger, W. Li, S. Osher, Solving large-scale optimization problems with a convergence rate independent of grid size, SIAM J. Numer. Anal. 57 (2019) 1100–1123.
[20] N. Papadakis, G. Peyré, E. Oudet, Optimal transport with proximal splitting, SIAM J. Imaging Sci. 7 (2014) 212–238.
[21] J. Yu, R. Lai, W. Li, S. Osher, A fast proximal gradient method and convergence analysis for dynamic mean field planning, arXiv preprint, arXiv:2102.13260, 2021.
[22] L. Ruthotto, S.J. Osher, W. Li, L. Nurbekyan, S.W. Fung, A machine learning framework for solving high-dimensional mean field game and mean field control problems, Proc. Natl. Acad. Sci. 117 (2020) 9183–9193.
[23] I. Kobyzev, S. Prince, M. Brubaker, Normalizing flows: an introduction and review of current methods, IEEE Trans. Pattern Anal. Mach. Intell. (2020).
[24] G. Peyré, M. Cuturi, Computational optimal transport: with applications to data science, Found. Trends Mach. Learn. 11 (2019) 355–607.
[25] T. Teshima, I. Ishikawa, K. Tojo, K. Oono, M. Ikeda, M. Sugiyama, Coupling-based invertible neural networks are universal diffeomorphism approximators, Adv. Neural Inf. Process. Syst. 33 (2020) 3362–3373.
[26] P.L. Bartlett, D.J. Foster, M.J. Telgarsky, Spectrally-normalized margin bounds for neural networks, Adv. Neural Inf. Process. Syst. 30 (2017).
[27] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, N. Usunier, Parseval networks: improving robustness to adversarial examples, in: International Conference on Machine Learning, PMLR, 2017, pp. 854–863.
[28] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using real nvp, arXiv preprint, arXiv:1605.08803, 2016.
[29] C. Durkan, A. Bekasov, I. Murray, G. Papamakarios, Neural spline flows, Adv. Neural Inf. Process. Syst. 32 (2019) 7511–7522.
[30] M. Cuturi, Sinkhorn distances: lightspeed computation of optimal transport, Adv. Neural Inf. Process. Syst. 26 (2013) 2292–2300.
[31] A.T. Lin, S.W. Fung, W. Li, L. Nurbekyan, S.J. Osher, Alternating the population and control neural networks to solve high-dimensional stochastic mean-field games, Proc. Natl. Acad. Sci. 118 (2021) e2024713118.
[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. 27 (2014).
[33] L. Dinh, D. Krueger, Y. Bengio, Nice: non-linear independent components estimation, arXiv preprint, arXiv:1410.8516, 2014.
[34] J. Ho, X. Chen, A. Srinivas, Y. Duan, P. Abbeel, Flow++: improving flow-based generative models with variational dequantization and architecture design, in: International Conference on Machine Learning, PMLR, 2019, pp. 2722–2730.

[35] G. Papamakarios, T. Pavlakou, I. Murray, Masked autoregressive flow for density estimation, arXiv preprint, arXiv:1705.07057, 2017.
[36] D.P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, M. Welling, Improved variational inference with inverse autoregressive flow, Adv. Neural Inf. Process. Syst. 29 (2016) 4743–4751.
[37] C.-W. Huang, D. Krueger, A. Lacoste, A. Courville, Neural autoregressive flows, in: International Conference on Machine Learning, PMLR, 2018, pp. 2078–2087.
[38] A. Wehenkel, G. Louppe, Unconstrained monotonic neural networks, Adv. Neural Inf. Process. Syst. 32 (2019) 1545–1555.
[39] R.T. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, arXiv preprint, arXiv:1806.07366, 2018.
[40] W. Grathwohl, R.T. Chen, J. Bettencourt, I. Sutskever, D. Duvenaud, Ffjord: free-form continuous dynamics for scalable reversible generative models, arXiv preprint, arXiv:1810.01367, 2018.
[41] D. Onken, S. Wu Fung, X. Li, L. Ruthotto, Ot-flow: fast and accurate continuous normalizing flows via optimal transport, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021.
[42] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, A. Oberman, How to train your neural ode: the world of jacobian and kinetic regularization, in: International Conference on Machine Learning, PMLR, 2020, pp. 3154–3164.
[43] J.-M. Lasry, P.-L. Lions, Mean field games, Jpn. J. Math. 2 (2007) 229–260.
[44] M. Huang, R.P. Malhamé, P.E. Caines, et al., Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle, Commun. Inf. Syst. 6 (2006) 221–252.
[45] M. Huang, P.E. Caines, R.P. Malhamé, Large-population cost-coupled lqg problems with nonuniform agents: individual-mass behavior and decentralized ε-Nash equilibria, IEEE Trans. Autom. Control 52 (2007) 1560–1571.
[46] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, arXiv preprint, arXiv:1312.6114, 2013.
[47] T. Müller, B. McWilliams, F. Rousselle, M. Gross, J. Novák, Neural importance sampling, ACM Trans. Graph. 38 (2019) 1–19.
[48] C. Durkan, A. Bekasov, I. Murray, G. Papamakarios, Cubic-spline flows, arXiv preprint, arXiv:1906.02145, 2019.
[49] P. Jaini, K.A. Selby, Y. Yu, Sum-of-squares polynomial flow, in: International Conference on Machine Learning, PMLR, 2019, pp. 3009–3018.
[50] D.P. Kingma, P. Dhariwal, Glow: generative flow with invertible 1x1 convolutions, Adv. Neural Inf. Process. Syst. 31 (2018).
[51] L. Ambrosio, N. Gigli, G. Savaré, Gradient Flows: in Metric Spaces and in the Space of Probability Measures, Springer Science & Business Media, 2008.
[52] B. Neyshabur, R. Tomioka, N. Srebro, Norm-based capacity control in neural networks, in: Conference on Learning Theory, PMLR, 2015, pp. 1376–1401.
[53] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[54] C. Barilla, G. Carlier, J.-M. Lasry, A mean field game model for the evolution of cities, J. Dyn. Games 8 (2021) 299.
[55] G. Wang, W. Yao, X. Zhang, Z. Niu, Coupled alternating neural networks for solving multi-population high-dimensional mean-field games with stochasticity, 2022, TechRxiv.
[56] D. Dua, C. Graff, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2017.
[57] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, vol. 2, IEEE, 2001, pp. 416–423.
[58] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, Z. Tu, Deeply-supervised nets, in: Artificial Intelligence and Statistics, PMLR, 2015, pp. 562–570.
[59] E. Nalisnick, A. Matsukawa, Y.W. Teh, D. Gorur, B. Lakshminarayanan, Do deep generative models know what they don't know?, arXiv preprint, arXiv:1810.09136, 2018.
[60] R.M. Dudley, Real Analysis and Probability, 2 ed., Cambridge Studies in Advanced Mathematics, Cambridge University Press, 2002.
[61] T. Duan, Normalizing-flows, https://github.com/tonyduan/normalizing-flows, 2019.
[62] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in: NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.
[63] P. Helber, B. Bischke, A. Dengel, D. Borth, Eurosat: a novel dataset and deep learning benchmark for land use and land cover classification, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 12 (2019) 2217–2226.
[64] K. Bliznashki, Normalizing_flows, https://github.com/kamenbliznashki/normalizing_flows, 2020.