Physica D 460 (2024) 134082

Contents lists available at ScienceDirect

Physica D

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

A DeepParticle method for learning and generating aggregation patterns in multi-dimensional Keller–Segel chemotaxis systems

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

[a] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, 21 Nanyang Link, Singapore 637371, Singapore

[b] Department of Mathematics, University of California at Irvine, Irvine, CA 92697, USA

[c] Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong Special Administrative Region of China

ARTICLE INFO

Communicated by V.M. Perez-Garcia

Keywords:
Keller–Segel system
Chemotaxis
Interacting particle approximation
Optimal transportation
Wasserstein distance
Deep neural networks

ABSTRACT

We study a regularized interacting particle method for computing aggregation patterns and near singular solutions of a Keller–Segel (KS) chemotaxis system in two and three space dimensions, then further develop the DeepParticle method to learn and generate solutions under variations of physical parameters. The KS solutions are approximated as empirical measures of particles that self-adapt to the high gradient part of solutions. We utilize the expressiveness of deep neural networks (DNNs) to represent the transform of samples from a given initial (source) distribution to a target distribution at a finite time T prior to blowup without assuming the invertibility of the transforms. In the training stage, we update the network weights by minimizing a discrete 2-Wasserstein distance between the input and target empirical measures. To reduce the computational cost, we develop an iterative divide-and-conquer algorithm to find the optimal transition matrix in the Wasserstein distance. We present numerical results of the DeepParticle framework for successful learning and generation of KS dynamics in the presence of laminar and chaotic flows. The physical parameter in this work is either the evolution time or the flow amplitude in the advection-dominated regime.

1. Introduction

Chemotaxis partial differential equations (PDEs) were introduced by Keller and Segel (KS [1]) to describe the aggregation of the slime mold amoeba Dictyostelium discoideum due to an attractive chemical substance. Related random walk model by Patlak was known earlier [2], see [3] for an analysis of basic taxis behaviors (aggregation, blowup, and collapse) based on reinforced random walks. Recall a common form of KS model [4] as follows:

$$\rho_t = \nabla\cdot(\mu\nabla\rho - \chi\,\rho\nabla c), \quad \epsilon\, c_t = \Delta c - k^2 c + \rho, \tag{1}$$

where χ, μ (ϵ, k) are positive (non-negative) constants. The model is called elliptic if $\epsilon = 0$ (when c evolves rapidly to a local equilibrium), and parabolic if $\epsilon > 0$. The ρ is the density of active particles (bacteria), and c is the concentration of chemo-attractant. The bacteria diffuse with mobility μ and drift in the direction of ∇c with velocity $\chi\nabla c$, where χ is called chemo-sensitivity.

In the simplest regime $(\epsilon, k) = \mathbf{0}$, the concentration equation becomes the Poisson equation $-\Delta c = \rho$. Subject to a suitable boundary condition of c, the classical integral representation $c = -\mathcal{K} * \rho$ holds, where $\mathcal{K}$ is the Green's function of the Laplacian, and $*$ denotes convolution. The KS system then reduces to a scalar non-local nonlinear advection–diffusion PDE governing the evolution of the density function ρ:

$$\rho_t = \mu\,\Delta\rho + \chi\,\nabla\cdot\big(\rho\,\nabla(\mathcal{K} * \rho)\big). \tag{2}$$

For modeling chemotaxis in a fluid environment such as ocean [5–9], people studied Eq. (2) with the advective Lie derivative ρ on the left hand:

$$\rho_t + \nabla\cdot(\rho\,\mathbf{v}) = \mu\,\Delta\rho + \chi\,\nabla\cdot\big(\rho\,\nabla(\mathcal{K} * \rho)\big). \tag{3}$$

The mixing mechanism of the flow field $\mathbf{v}$ is known to slow down or smooth out blowup or aggregation in (2), see analysis in [5,6,8,9] and references therein, and related convection induced smoothing in fluids [10,11]. Eq. (3) is the macroscopic limit (McKean–Vlasov equation) of the interacting particle system below as $J \uparrow \infty$:

$$dX^j = -\frac{\chi\mathcal{M}}{J}\nabla_{X^j}\sum_{i=1,i\neq j}^{J}\mathcal{K}(|X^j - X^i|)\,dt + \mathbf{v}(X^j)dt + \sqrt{2\mu}\,dW^j, \quad j = 1,\dots,J, \tag{4}$$

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

https://doi.org/10.1016/j.physd.2024.134082
Received 22 February 2023; Received in revised form 18 August 2023; Accepted 27 January 2024
Available online 3 February 2024

where M is the conserved total mass (integral of ρ), and W^j's are independent Brownian motions in $\mathbb{R}^d$.

In this work, $\mathbf{v}$ is a prescribed divergence-free vector field. We shall approximate ρ of Eq. (3) numerically based on the associated interacting particle system in two and three spatial dimensions ($d = 2, 3$), and carry out a systematic deep learning study (a.k.a. DeepParticle [12]) on the $(\mu, \mathbf{v})$ dependency of solutions. As we are interested in studying near singular KS solutions, the main challenge for training data collection is to approximate the fields $\rho(\mathbf{x})$ and $c(\mathbf{x})$ reliably as they intensify. Due to the singular behavior of Green's function, as particles come close to each other, our approach is to regularize $\mathcal{K}$ in (4) for approximating $\rho(\mathbf{x})$ as particles aggregate, in a similar spirit to the vortex blob method for fluids ([13] and references therein) and [14].

Deep learning tools have been applied broadly for scientific computing in recent years, such as solving PDEs and their inverse problems; see [12,15] and references therein. DeepParticle [12] is based on a particle method for solving a time-dependent physically parameterized PDE, whose solution is approximated by the particle empirical measure (distribution). A deep neural network (DNN) with physical parameter dependence learns the mapping from the initial particle distribution to the particle distribution at time T with training data over sampled physical parameters provided by the particle method. The trained DNN then generates approximate solutions at time T for new physical parameters unseen in the training process. DeepParticle has been successfully designed and trained for learning and generating invariant measures of stochastic interacting particle systems (at $T = \infty$) arising in reaction–diffusion front speeds in three-dimensional chaotic flows [12]. In this paper, we further develop DeepParticle to learn and generate KS solutions exhibiting aggregation behavior at a finite time T before blow-up for a range of diffusivity μ and advection amplitude values.

The rest of the paper is organized as follows. In Section 2, we briefly review the blow-up behavior in the KS model and the regularized particle methods to solve the KS model. In Section 3, we present our DeepParticle method to learn the transport map from an input distribution to a target distribution. Moreover, we will discuss the details of the implementation of our method and how to learn the distributions in particle simulation of the KS model. In Section 4, we show numerical results to demonstrate the performance of our method, where both 2D and 3D KS chemotaxis systems will be studied. Finally, conclusions and future works will be discussed in Section 5.

2. Regularized interacting particle method of KS model

2.1. Blow up behavior in KS model

We start from the simplest KS model without advection, namely the Eq. (2) with $\mu = \chi = 1$ in two spatial dimensions ($d = 2$). This system has been extensively studied by many authors; see the survey article [16] and references therein. The conservation of mass holds:

$$\frac{d}{dt}\int_{\mathbb{R}^2} \rho(x,t)dx = 0, \tag{5}$$

which is also true for Eq. (3) when the advection field $\mathbf{v}$ is divergence-free. If we set the total mass $M := \int_{\mathbb{R}^2} \rho(x,0)dx$, the second moment has a fixed time derivative, i.e.,

$$\frac{d}{dt}\int_{\mathbb{R}^2} |x|^2\rho(x,t)\,dx = \frac{M}{2\pi}(8\pi - M), \tag{6}$$

where 8π is called the critical mass of the KS system. Accordingly, it is well-known that: (1) if $M > 8\pi$, the system has no global smooth solutions; (2) if $M = 8\pi$, the system has a global smooth solution, which blows up as $t \to \infty$; and (3) if $M < 8\pi$, the system has a global smooth solution.

In [5–7], an extra advection term is introduced to KS-type equations, in order to model organism movement in prescribed fluid flows. Then, the second-moment identity in Eq. (6) and the subsequent blowup vs. global evolution results are no longer valid. By comparison principle, [7] shows that if the total mass is smaller than the critical mass, Eq. (3) has a global smooth solution with smooth initial data. In the cases with supercritical mass, there are only numerical experiments suggesting that the advection, if sufficiently large, prevents the solutions from blowing up. Later, [9] shows that when the flow exerts a 'stretching' effect, the advection field $\mathbf{v}$ indeed suppresses the growth or the concentration of chemo-attractant and hence the solution has global regularity and exists for all time. Examples of stretching flows include hyperbolic flows where $\mathbf{v}(\mathbf{x}) = \left(x_1, -\frac{1}{d-1}\mathbf{x}_-\right)$ and laminar flows where $\mathbf{v}(\mathbf{x}) = \left(v\left(\mathbf{x}_-\right), \mathbf{0}_-\right)$, with $\mathbf{x}_- = \left(x_2, \ldots, x_d\right)$ and v is periodic. However, the case of chaotic flows, or when the amplitude of advection $\mathbf{v}$ is not sufficiently large, remains open.

2.2. Regularized interacting particle methods

The singular behavior of the governing PDE and the associated Green's function that cause trouble for particle methods is a well-known problem. The vortex blob method [17] provides a regularization approach to extend vortex sheet motion past the singularity formation time into the physically important roll-up regime, in which the points representing the vortex sheet are replaced by vortices of prescribed and fixed shape. Numerical calculations show regular motion for the centers of the blobs even after the time when a curvature singularity on a vortex sheet is formed. Later, a special form of the vortex blob method [18] is used to calculate the roll-up of a periodic vortex sheet resulting from the classical Kelvin–Helmholtz instability. The singular solutions are closely related to the ill-posedness of vortex sheet motion [19]. Nonetheless, as the regularization parameter approaches zero, the regularized vortex sheet solution converges to a weak solution of the Euler equation [20]. As in vortex blob methods, we formulate a regularized interacting particle method (denoted as IPM from here on) to solve KS chemotaxis systems.

Let us approximate the density function (solution of Eq. (2) or Eq. (3)) with empirical distributions of particle positions. In SDE (4) on particle positions, the chemo-attractant term $\frac{\chi\mathcal{M}}{J}\nabla_{X^j}\sum_{i=1,i\neq j}^{J}\mathcal{K}(|X^j - X^i|)\,dt$ causes numerical instabilities when particles tend to concentrate. To overcome this difficulty, we replace the singular kernel $\mathcal{K}(\cdot)$ in (4) by a smoothed approximation $K_\delta(\cdot)$, such that $K_\delta(z) \to K(z)$ as $\delta \to 0$, where $\delta > 0$ is a regularization parameter. For example, we define

$$K_\delta(z) = K(z)\frac{|z|^2}{|z|^2 + \delta^2}. \tag{7}$$

Equipped with the kernel $K_\delta(\cdot)$, we obtain a system of regularized SDEs for the particles as follows:

$$\begin{aligned} dX^j &= -\frac{\chi\mathcal{M}}{J}\nabla_{X^j}\sum_{i=1,i\neq j}^{J} K_\delta(|X^j - X^i|)\,dt + \mathbf{v}(X^j)dt + \sqrt{2\mu}\,dW^j, \\ &\quad j = 1, 2, \ldots, J, \end{aligned} \tag{8}$$

where J is the number of particles, $X^j \in \mathbb{R}^d$ is the position of the jth particle, and dW^j's are mutually independent d-dimensional Brownian motions. The convergence of a random particle blob method, similar to (8) yet for KS without advection field $\mathbf{v}$, is analyzed in [21]. For a stochastic interacting particle method using heuristic collision and splitting rules to bypass the singular behavior of Green's function, see [22].

Representing PDE solutions by particles belongs to the Lagrangian framework. The Lagrangian methods have several advantages: (1) easy to implement; (2) spatially mesh-free and self-adaptive; and (3) computational costs scale linearly with the dimension of spatial variables in the underlying stochastic dynamical systems. If we discretize KS system (3) on mesh grids with grid-based methods, e.g. finite element [23] and spectral [24] methods, the number of mesh grids depends exponentially on the spatial dimension. The key benefit of the Lagrangian framework

in computing KS models is its natural capability to follow the KS solution when a singular behavior is emerging.

As we shall see, the stochastic particle method based on (8) reproduces the well-known aggregation behavior and captures the KS dynamics during the potential blow-up stage of evolution. This is another step forward in our program of computing high gradient solutions in the Lagrangian framework, which has shown encouraging results for a range of multi-dimensional singularly-perturbed advection–diffusion PDEs. We refer interested readers to our recent progress in developing Lagrangian methods to compute effective diffusivities in chaotic or random flows [25–28] and KPP front speeds in chaotic flows [29] and random flows [30]. There are also deterministic particle methods ([14, 31] and references therein) for a class of KS and degenerate diffusion equations that fall in our DeepParticle framework (see Section 3).

Though the IPM in our study here is mesh-free and self-adaptive for solving multi-dimensional KS chemotaxis systems, the computational costs remain high if we want to study the systems under a variation of parameters (e.g. the evolution time T and the amplitude of advection A). Also, the number of particles J cannot be too large as the chemo-attractant term has $O(J^2)$ complexity in each time step evolution. The total complexity is then $O(J^2\frac{T}{\Delta t})$, where Δt is the time step of discretization for Eq. (8). On the other hand, our numerical simulation of Eq. (8) shows that the distribution at finite time T, starting from the same initial distribution, may have continuous dependence on the physical parameters. Therefore in the next section, we will introduce a deep learning algorithm that can learn the continuous dependence of the solutions on parameters of the KS models and generate approximated samples with $O(J)$ complexity.

3. Deep particle method

In this section, we introduce a DeepParticle algorithm to learn the features of the transport map from a trivial (input) distribution to a target (output) distribution. The mapping error is measured by the 2-Wasserstein distance.

3.1. Discrete Wasserstein distance

Given distributions μ and ν defined on metric spaces X and Y, let us construct a transport map $f_*^0 : X \to Y$ such that $f_*^0(\mu) = \nu$, where star denotes the push forward of the map. For any function $f_* : X \to Y$, the 2-Wasserstein distance between $f_*(\mu)$ and ν is defined by:

$$W_2(f_*(\mu),\nu) := \left(\inf_{\gamma\in\Gamma(f_*(\mu),\nu)}\int_{Y\times Y}\mathrm{dist}(y',y)^2\,\mathrm{d}\gamma(y',y)\right)^{1/2}, \tag{9}$$

where $\Gamma(f_*(\mu),\nu)$ denotes the collection of all measures on $Y\times Y$ with marginals $f_*(\mu)$ and ν on the first and second factors respectively, and $\mathrm{dist}(\cdot,\cdot)$ denotes the metric (distance) on Y. A straightforward derivation yields:

$$W_2(f_*(\mu),\nu) = \left(\inf_{\gamma\in\Gamma(\mu,\nu)}\int_{X\times Y}\mathrm{dist}(f(x),y)^2\,\mathrm{d}\gamma(x,y)\right)^{1/2}, \tag{10}$$

where $\Gamma(\mu,\nu)$ denotes the collection of all measures on $X\times Y$ with marginals μ and ν on the first and second factors respectively, and still $\mathrm{dist}(\cdot,\cdot)$ denotes the metric (distance) on Y. To design computational methods, we approximate distributions μ and ν by empirical distribution functions: $\mu = \frac{1}{N}\sum_{i=1}^N \delta_{x_i}$ and $\nu = \frac{1}{N}\sum_{j=1}^N \delta_{y_j}$, where N is the number of samples in the empirical distributions. Under the setting of learning distribution from interacting particle methods, we take $N < J$ sub-samples from the terminal time position of system (4) to represent the distribution. We will re-sample them every 1000 steps of training. The preceding technique is usually referred to as *mini-batch* in the deep learning literature.

Table 1
Shape and size of network parameters to learn in case of $d = 3$ and $p = 1$.

Parameter	Shape	Size
W_0	30×4	120
b_0	30×1	30
$W_{1:4}$	30×30	900×4
$b_{1:4}$	30×1	30×4
W_5	3×30	90
b_5	3×1	3
Total		3963

Any joint distribution in $\Gamma(\mu,\nu)$ can be approximated by an $N\times N$ doubly stochastic matrix [32], denoted as transition matrix, $\gamma = (\gamma_{ij})_{i,j}$ satisfying:

$$\gamma_{ij}\ge 0;\qquad \forall j,\ \sum_{i=1}^N \gamma_{ij} = 1;\qquad \forall i,\ \sum_{j=1}^N \gamma_{ij} = 1. \tag{11}$$

Then (10) becomes

$$\hat{W}(f) := \left(\inf_{\gamma\in\Gamma^N}\frac{1}{N}\sum_{i,j=1}^N \mathrm{dist}(f(x_i),y_j)^2\gamma_{ij}\right)^{1/2}. \tag{12}$$

$\hat{W}(f)$ in (12) has a simple intuitive interpretation: given a $\gamma\in\Gamma(\mu,\nu)$ and any pair of locations (x,y), the value of $\gamma(x,y)$ tells us what proportion of $f_*(\mu)$ mass at $f(x)$ should be transferred to y, in order to reconfigure $f_*(\mu)$ into ν. Computing the effort of moving a unit of mass from $f(x)$ to y by $\mathrm{dist}(f(x),y)^2$ yields the interpretation of $\hat{W}$ as the minimal effort (optimal transportation [33]) to reconfigure $f_*(\mu)$ mass distribution into that of ν.

3.2. Training data and neural network configuration

Note that given any fixed set of $\{x_i\}_{i=1}^N \subset \mathbb{R}^d$ and $\{y_j\}_{j=1}^N \subset \mathbb{R}^d$ (training data), we have derived Eq. (12) to be minimized by gradient descent. In addition, we aim to find a network that can represent the change of target distribution over some physical parameters. In such a scenario, more than one set of data ($\{x_i\}$ and $\{y_j\}$ consists of one set of data) should be assimilated. More precisely, let the total number of data set be denoted as N_{dict}. Then we have N_{dict} pairs of i.i.d. samples of input and output distribution, denoted by $\{x_{i,r}\}$ and $\{y_{j,r}\}$ for $r = 1\cdots N_{dict}$. Associated with the rth data set ($\{x_{i,r}\}$ and $\{y_{j,r}\}$), we assume that there is a physical parameter $\eta_r\in\mathbb{R}^p$. To represent this in the network, we encode η_r to each data in the set, i.e., the input of the network is $\{(x_{i,r},\eta_r)\}_i \subset \mathbb{R}^{d+p}$. This procedure is also called *padding* in the literature. The output of the network is then denoted by $f_\theta(x;\eta)$ where θ's are all trainable parameters of the network, i.e., the weights in the neural network.

We want to emphasize that our neural network is very compact. Between the input layer (l_0) and output layer (l_6), there are 5 latent layers, where each layer is 30 in width. The adjacent layers l_i and l_{i+1} are fully connected, i.e.,

$$l_{i+1} = \tanh(W_i(l_i) + b_i)\quad i = 0,\dots,4, \tag{13}$$

where W_i is (weight) matrix width of l_{i+1} and l_i, b_i is the (bias) vector with the same dimension as l_i, and $\tanh(\cdot)$ is the activation function. For the output layer, the formula is similar except we do not apply the activation function. In the case of $d = 2$ and $p = 1$ (e.g. the first numerical example in computing blowup behavior of KS without advection in Section 4.1), there are 3902 parameters (weight and bias) to learn. In 3D cases, we find that the network performs well even without altering the width of latent layers, where the parameter number is 3963; see Table 1 for the detailed shape and size.

Remark 3.1. Compared with the par-net approach developed in [12], we found that *padding* achieves similar performance when learning distribution in particle simulation of the KS model.

In our computation, by equipping $\mathbb{R}^d$ with Euclidean metric, we can get the training loss function as follows:

$$\hat{W}^2(f_\theta) := \frac{1}{Nn_\eta}\sum_{r=1}^{n_\eta}\left(\inf_{\gamma_r\in\Gamma^N}\sum_{i,j=1}^{N}|f_\theta(x_{i,r};\eta_r)-y_{j,r}|^2\gamma_{ij,r}\right). \tag{14}$$

The goal of DeepParticle algorithm is then to find $(\theta,\{\gamma_r\})$ through minimizing

$$P(\theta,\{\gamma_r\}) := \sum_{r=1}^{n_\eta}\sum_{i,j=1}^{N}\left(|f_\theta(x_{i,r};\eta_r)-y_{j,r}|^2\gamma_{ij,r}\right). \tag{15}$$

3.3. Iterative method in finding transition matrix γ

Notice that, with fixed θ, finding the transition matrix γ to calculate discrete Wasserstein distance in (14) is a linear programming problem with a degree of freedom N^2. When the number of particles N becomes large, it is expensive to go by a conventional algorithm, e.g. interior point algorithm or simplex algorithm. We now present a mini-batch linear programming algorithm to find the best γ for each inner sum of (14), while suppressing η_r dependence in f_θ for notational simplicity.

The problem (14) is a linear programming problem on the bounded convex set Γ^N of vector space of real $N\times N$ matrices. By Choquet's theorem, this problem admits solutions that are extremal points of Γ^N. The set of all doubly stochastic matrix Γ^N is viewed as Birkhoff polytope. The Birkhoff–von Neumann theorem [34] states that such polytope is the convex hull of all permutation matrices, i.e., those matrices such that $\gamma_{ij}=\delta_{j,\pi(i)}$ for some permutation π of $\{1,\dots,N\}$, where δ_{jk} is the Kronecker symbol.

Our algorithm is defined iteratively. We start from a permutation matrix, e.g., $\gamma_{ij}=\delta_{ij}$. In each iteration, we randomly select columns and corresponding rows such that the submatrix is a permutation matrix. Then, the entries of the submatrix consist of a linear programming problem under the constraint that maintains column-wise and row-wise sums equal to one. To be precise, we randomly choose $\{i_k\}_{k=1}^M$, $(M\ll N)$ from $\{1,2,\dots,N\}$ without replacement. Then, we select the indices j_k's such that $\gamma_{i_k,j_k}=1$. The cost function of the sub-problem is

$$C(\gamma^*) := \sum_{k,l=1}^{M}|f_\theta(x_{i_k})-y_{j_l}|^2\gamma^*_{i_kj_l} \tag{16}$$

subject to

$$\begin{cases}\displaystyle\sum_{k=1}^{M}\gamma^*_{i_k,j_l}=1 \quad \forall l=1,\dots,M\\ \displaystyle\sum_{l=1}^{M}\gamma^*_{i_k,j_l}=1 \quad \forall k=1,\dots,M\\ \gamma^*_{i_kj_l}\geq 0 \quad \forall k,l=1,\dots,M.\end{cases} \tag{17}$$

Then, the minimizer γ^* is again a permutation matrix. The linear programming sub-problem of much smaller size is solved by the interior point method [35]. In addition, as the goal is to find a permutation matrix, we set the tolerance of the interior point to be relatively large and project the resulting approximation to a permutation matrix. In our approach, we terminate the iteration of the sub-problem until $\min_k(\max_l\gamma^*_{i_k,j_l})>0.5$. This is to ensure that there is a unique large-value entry in each column. As a projection, we update γ by,

$$\gamma_{ij}=\begin{cases}1 & \text{if } j=\arg\max_l\gamma^*_{il},\\ 0 & \text{otherwise.}\end{cases} \tag{18}$$

We observe that the global minimizer of γ in (14) is also the solution of sub-problems (16) with arbitrarily selected rows and columns, subject to the row and column partial sum values of the global minimum. The selection of rows can be one's own choice. In our approach, in each step after gradient descent, we randomly select rows to solve the optimization problem iteratively.

Compared with the Wasserstein Generative Adversarial Network (WGAN) proposed in [36], (15) is a Min-Min optimization problem. Both Adam gradient descent of θ and mimi-batch optimization of γ are iteratively defined. We then alternatively update θ and γ to seek a global minimizer of (15).

The computational cost of finding optimal γ increases as N increases, however, the network itself is independent of γ. After training, our network acts as a sampler for some target distribution ν without the assumption that ν should have a closed-form distribution function. At this stage, the input data is no longer limited by training data, and an arbitrarily large amount of samples approximately obeying ν can be generated through μ (uniform distribution).

3.4. Full training algorithm

The full training process is outlined in Alg. 1, and carried out on a quad-core CPU desktop with an RTX2070 8 GB GPU at UC Irvine. The training data is collected from the first-order explicit IPM that solves the regularized SDE system (8) by Euler's method in time. The IPM is also the reference solver for evaluating DeepParticle generations.

Algorithm 1: DeepParticle Learning

Randomly initialize weight parameters θ in network $f_\theta:\mathbb{R}^d\to\mathbb{R}^d$;
repeat
  for *physical parameter set* $r\leftarrow 0$ **to** n_η **do**
    randomly select $\{x_{i,r}\}$, $\{y_{j,r}\}$, $i,j=1:N$ from i.i.d. samples of input and target distribution (generated by the IPM) with respect to physical parameter η_r;
    $\gamma_{ij,r}=\delta_{i,j}$, i.e., initialize it as a permutation matrix;
  end
  if *not the first training mini-batch* **then**
    for *physical parameter set* $r\leftarrow 0$ **to** n_η **do**
      $\delta P_r=+\infty$
      while $|\delta P_r|<tol$ **do**
        $P_r=\sum_{i,j=1}^{N}|f_\theta(x_{i,r},\eta_r)-y_{j,r}|^2\gamma_{ij,r}$;
        randomly choose $\{i_{k,r}\}_{k=1}^M$ from $\{1,2,\cdots,N\}$ without replacement;
        find $\{j_{k,r}\}_{k=1}^M$ such that $\gamma_{i_{k,r}j_{k,r},r}=1$;
        solve the linear programming sub-problem (16)–(17) and get γ^*_r that is another permutation matrix;
        update $\{\gamma_{i_{k,r}j_{l,r}}\}_{k,l=1}^M$ with $\{\gamma^*_{i_{k,r}j_{l,r}}\}_{k,l=1}^M$;
        $\delta P_r=\sum_{i,j=1}^{N}|f_\theta(x_{i,r},\eta_r)-y_{j,r}|^2\gamma^*_{ij,r}-P_r$.
      end
    end
  end
  repeat
    $P=\sum_{r=1}^{N_r}\sum_{i,j=1}^{N}|f_\theta(x_{i,r},\eta_r)-y_{j,r}|^2\gamma_{ij,r}$;
    $\theta\leftarrow\theta-\delta_1\nabla_\theta P$, δ_1 is the learning rate;
    repeat
      for *physical parameter set* $r\leftarrow 0$ **to** n_η **do**
        $P_r=\sum_{i,j=1}^{N}|f_\theta(x_{i,r},\eta_r)-y_{j,r}|^2\gamma_{ij,r}$;
        randomly choose $\{i_{k,r}\}_{k=1}^M$, from $\{1,2,\cdots,N\}$ without replacement;
        find $\{j_{k,r}\}_{k=1}^M$, such that $\gamma_{i_{k,r}j_{k,r},r}=1$;
        solve the linear programming sub-problem (16)–(17) and get γ^*_r which is another permutation matrix;
        update $\{\gamma_{i_{k,r}j_{l,r}}\}_{k,l=1}^M$ with $\{\gamma^*_{i_{k,r}j_{l,r}}\}_{k,l=1}^M$.
      end
    until *given linear programming steps,* N_{LP};
  until *given steps for each training mini-batch*;
until *given number of training mini-batches,* N_{dict};
Return

4. Numerical examples

4.1. 2D KS simulation and generation in the absence of advection

First, we consider the KS model without advection, namely the Eq. (2). In addition, we choose $\mu = \chi = 1$. A straightforward derivation shows that if the initial mass $M > 8\pi$ and has a finite second moment, the system will blow up in finite time.

As the first numerical example, we consider learning the change of distribution depending on evolution time T starting from the initial distribution. The initial distribution is assumed to be a uniform distribution on a ball with a radius 1 centered at the origin. Assuming the total mass is 16π, we have then the initial second moment as 8π, which is the same as in [7]. By Eq. (6), the system will blow up when $t > 0.125$. Before applying the deep learning algorithm, we first investigate the error of regularized method. In Fig. 1(a), we show the second moment of IPM simulation with various regularization parameters. We see that except for the value $\delta^2 = 10^{-2}$, a smaller regularization parameter δ does not affect the accuracy of the IPM simulation when $t \in [0, 0.1]$.

Then, we turn to investigate the performance of the proposed DeepParticle algorithms. To get the training data, we apply the IPM with regularization parameter $\delta^2 = 10^{-3}$ for $T = 0.1$ with $J = 10000$ particles. We keep the snapshots of the empirical distribution every 0.05 time interval. During the training process, we consider a mini-batch of size 8×2000, which means that we take $N_{dict} = 8$ sets of training data at various times t. Namely, we replace the general notation for physical parameter η in the network expression $f_\theta(x, \eta)$. Notice that the physical parameter η is the evolution time t in this example. In each data set (mini-batch), we have $N = 2000$ subsamples from $J = 10000$ samples from the IPM. We apply the Adam stochastic gradient descent method to learn the parameters (weights and bias) in the network. We renew the mini-batch every 1000 steps and renew γ every 200 steps.

In Fig. 1(b), we show the training loss of $\hat{W}^2(f_\theta)$ computed by Eq. (14). Since we do not renew γ at every step of the gradient descent of parameters, the training loss is not uniformly decreasing. However, the loss is on a decreasing trend overall, which shows that the network successively learns the feature of the distribution as training progresses. In addition, to validate the generalization capabilities of the model, we generate a set of validation distributions at time $t = [0.005, 0, 025, 0, 045, 0.065, 0.095]$ and compute the Wasserstein distance between output distribution and validation distribution every 10 steps. The trend is similar to the training loss.

After the training process, we denote the trained parameters (weights and bias) in the network as θ_1 and evaluate the network $f_{\theta_1}(x, t)$ at various time t with a larger batch $\{x_i\}_{i=1}^{J'}$ of size $J' = 1M$. Though $J' \gg J = 10K$, the computational cost of the network evaluation, $O(J')$, is obviously smaller than that of the IPM, $O(J^2 \frac{t}{\Delta t})$ as discussed in Section 2.2. In this example, the DeepParticle method takes 0.015 s to generate $J' = 1M$ samples while the reference IPM needs 120 seconds to generate $10K$ samples.

We emphasize that the generated output of the network is the image of a continuous map f_{θ_1} acting on finer samples of the input distribution, which provides an efficient method to generate samples for the output distribution. In fact, it is not a direct replica of training data when the physical parameter (e.g. the time t) overlaps with a value in the training set. The generated output of the network will not congregate on points in the training empirical distribution; see Fig. 8. Leveraging this feature, we can further use these generated samples as a warm-up step to accelerate the IPM simulation. A similar idea appears in our recent paper [12] when computing invariant measures.

In Fig. 1(c), we plot the second moment of the particles obtained by the IPM solver (denoted as reference solution) and by our DeepParticle (denoted as network) generation. We see that the slope of the second moment by reference IPM solver deviates from the network output after $t = 0.075$. This is due to the fact that the regularization parameter $\delta > 0$ in the IPM. For training data, we rely on the IPM solver most of the time when it has good accuracy prior to the time when the δ effect kicks in.

In Fig. 2, we plot the histogram of the network output (using 10^6 particles) and reference distribution (using 10^4 particles) at different times t. We see that the network learns the feature of particle concentration and even has a more concentrated output than the reference solver near the blowup time.

4.2. KS simulation and generation in the presence of 2D Laminar flows

Next, we consider the KS model with advection, i.e., $\mathbf{v} \neq 0$ in the Eq. (3). In this case, the movement of the organism is not only driven by the chemotaxis gradient but also driven by some given environmental fluid velocity field. In [7], the blow-up behavior of the KS model given various strengths of environment velocity is investigated. Now we let

$$\mathbf{v}(x, y) = A \begin{pmatrix} \exp(-y^2) \\ 0 \end{pmatrix}, \tag{19}$$

which represents a laminar flow of amplitude A traveling along x direction with y-dependent speed. There are two physical parameters to learn in the model: the amplitude A of the advection field and the evolution time t.

Learning the dependence on A. We first study the dependence of the aggregation patterns on the amplitude of the advection field while fixing the evolution time. In this example, we use the IPM to generate $J = 10000$ samples of solution after $T = 0.02$ with $A = 10^{0.2i}$, $i = 0, \ldots, 10$. The initial distribution of the IPM for each A is a uniform distribution on the unit ball. In Fig. 3, we can see that the distribution of the particles turns to a V shape as A increases. This is due to the stretching effect by the laminar flow (19). Numerical results show that our network can learn this important feature. In addition, our network can also predict distribution when A is slightly outside the range of the training set; see the case when $A = 150$ in Fig. 3(d). More precisely, Fig. 3(c) shows that most of the outputs in the training set ($1 \leq A \leq 100$) satisfy $x < 3$ while at $A = 150$, a reasonable proportion of particles is on the right side of $x = 3$ and our network indeed predicts it.

Learning the dependence on evolution time. Next, we study the dependence of the aggregation patterns on the evolution time at fixed $A = 100$. To generate training data, we run the IPM with $t \in [0, 0.1]$ and $J = 10000$ particles, and take snapshots of the empirical distribution at $t = 0, 0.01, 0.02, \ldots, 0.1$. In Fig. 4, we compare the output distribution generated by our network with the reference distribution generated by the IPM at various evolution times. We see that both IPM and DeepParticle methods reproduce the near blow-up behaviors, which are consistent with the results obtained in [7].

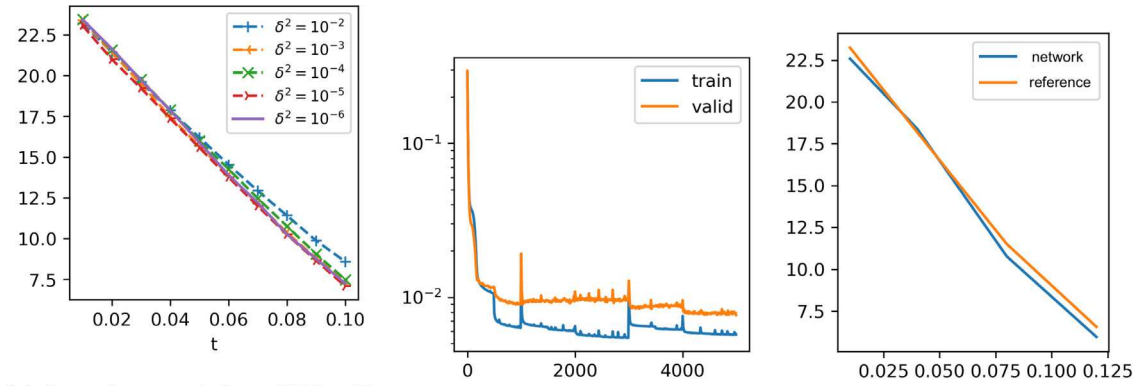
4.3. KS simulation and generation in the presence of 3D flows

In this subsection, we study the KS model with advection in three-dimensional space. There are two kinds of flows under consideration. The first flow is the 3D laminar flow which is a natural generalization of 2D laminar flow. The second one is the Kolmogorov flow which is a well-known example of chaotic flow [27,37,38].

A 3D Laminar flow. In the first 3D example, we consider an advection field of the following form:

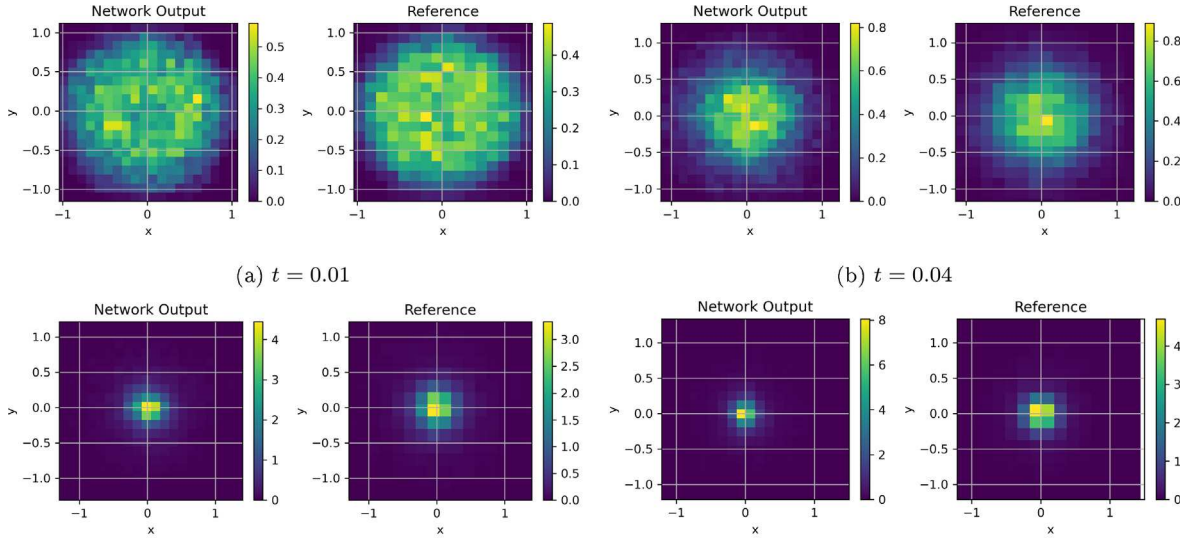
$$\mathbf{v}(x, y, z) = A \begin{pmatrix} \exp(-y^2 - z^2) \\ 0 \\ 0 \end{pmatrix}. \tag{20}$$

It describes the organism traveling along the x-direction while its speed depends on the radial position of y and z variable. In Fig. 5 we show the histogram of the generated distribution of our deep learning algorithms with $A = 10$ and $A = 100$, which reproduces the distribution of corresponding IPM simulation. The configuration of learning A dependence is the same as one in 2D cases, except the input and output are now



(a) Second moment from IPM with various regularization parameters (b) Training and validation loss of $\hat{W}^2$ (c) Comparison of the second moment

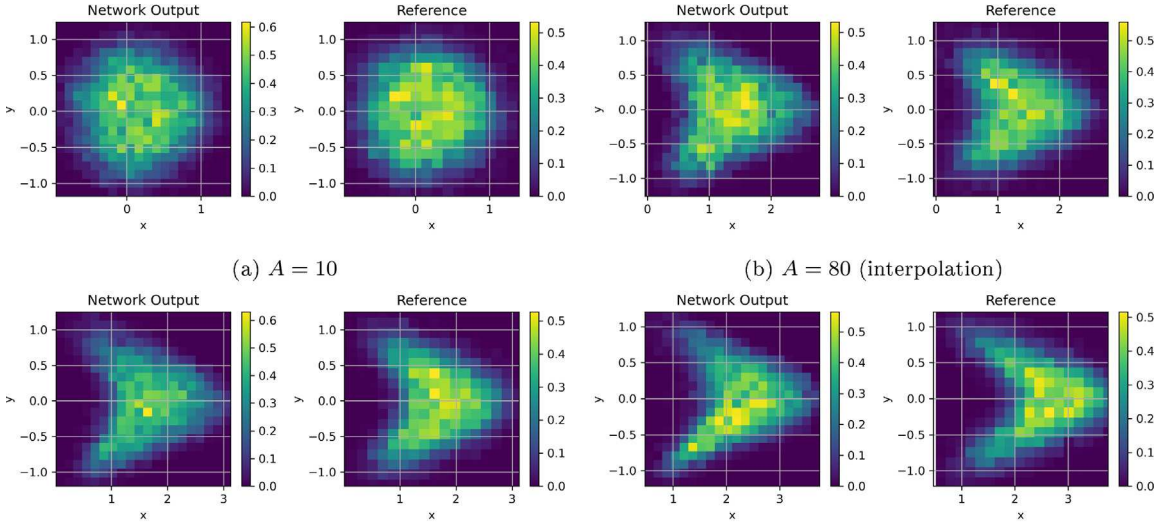
Fig. 1. Performance of the IPM (reference) and DeepParticle (network) algorithms.



(a) $t = 0.01$ (b) $t = 0.04$

(c) $t = 0.1$ (the largest t in the training data) (d) $t = 0.12$ (near blowup time)

Fig. 2. Comparison of particle distributions obtained by the DeepParticle method (network output) and IPM (reference) solver at different times and $A = 0$.



(a) $A = 10$ (b) $A = 80$ (interpolation)

(c) $A = 100$ (the largest A in the training set) (d) $A = 150$ (extrapolation)

Fig. 3. Learning particle aggregation at different A values with $t = 0.02$ fixed. Both the interpolation and extrapolation performances of the network are tested.

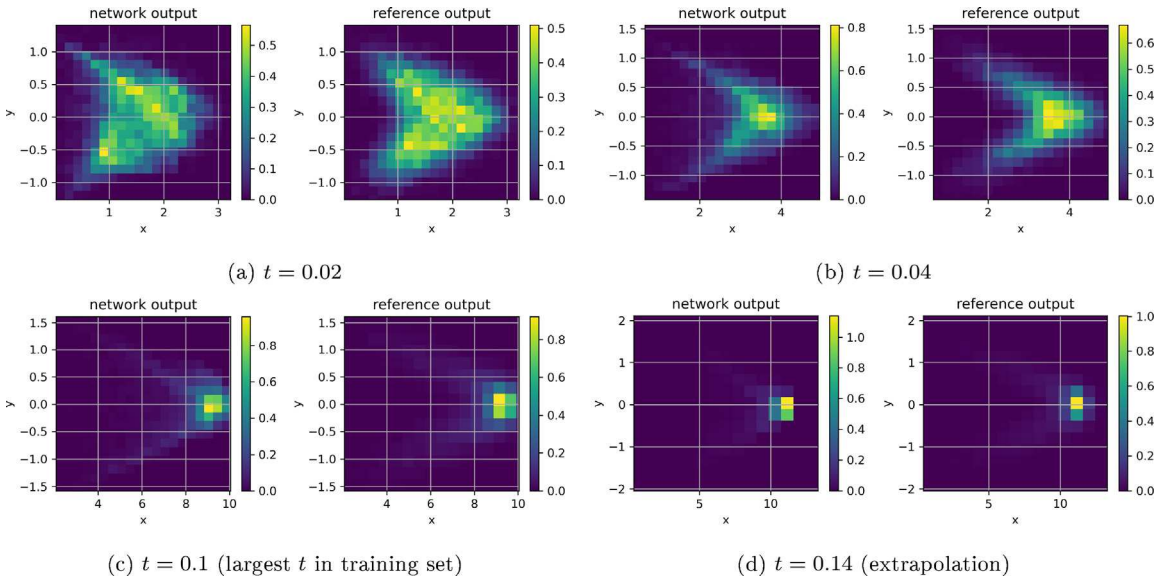


Fig. 4. Learning particle aggregation at different times with fixed flow amplitude $A = 100$. The extrapolation performance of the network is also tested.

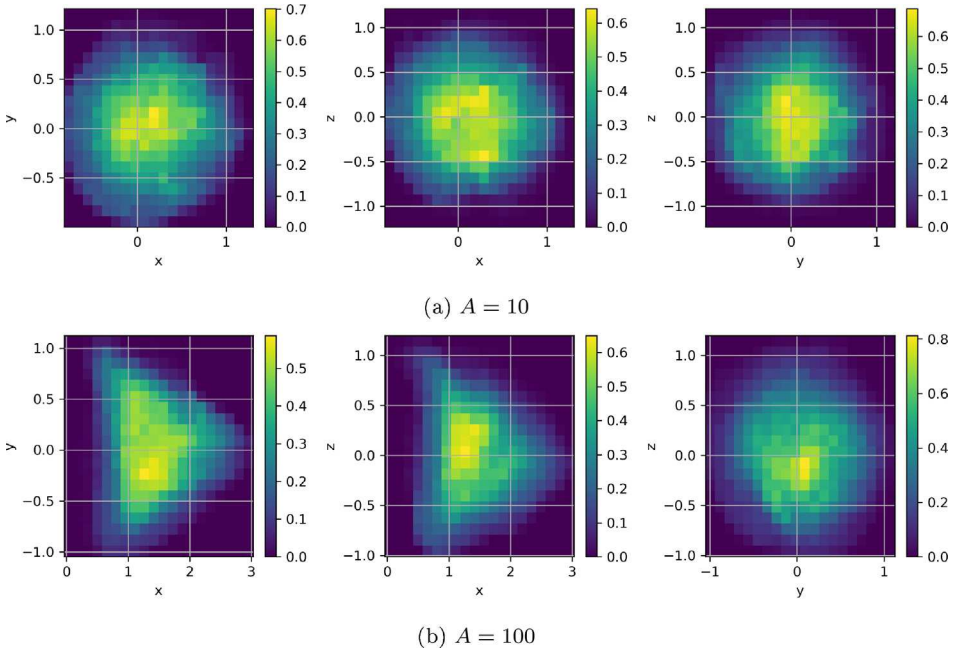


Fig. 5. Three cross sections of generated distributions at different A values in a 3D laminar flow (20).

in 3-dimension. From the numerical experiments (not shown), there is no need to increase the width of our network. By comparing (a) and (b) in Fig. 5, we see that the distribution becomes V shape in xy projection and xz projection as the amplitude A increases. This is due to the setting of the laminar flow. In the yz projection, the distribution remains radial.

In Fig. 6, we show the xy projection of the distribution with various A's. It confirms that in addition to interpretation, our network is able to extrapolate (predict) the aggregation pattern associated with the amplitude A that is beyond and not far from the range in the training set.

To further investigate the generalization capabilities of our learning algorithms, we generate $J_1 = 10^6$ realizations using the network, denoted as ρ_F, and $J_2 = 10^4$ realizations using the IPM, denoted as ρ_Y. As discussed in Section 3.3, direct computation of the Wasserstein distance between point cloud data ρ_F and ρ_Y involves linear programming with $J_1 \times J_2 = 10^{10}$ degrees of freedom. Therefore, we consider two types of rough comparisons.

First, we observe through the training data that the distribution of the first component increases as A increases. In Fig. 7(a), we compare the means of the first component of the generation at various A's. Second, we compute an approximation of the W^2 distance between ρ_F and ρ_Y. More precisely, we compute a 3D histogram of ρ_F (ρ_Y correspondingly) with B^3 uniform cells ($B = 20$). Then, we approximate the distribution of ρ_F by moving all points within a single cell of the histogram to the center of the cell. Finally, we compute the Wasserstein distance between the approximated distributions using the iterative method in Section 3.3. See Fig. 7(b) for the comparison of the W^2 distance (solid line) as well as the absolute distance of the mean (dashed line). These results show that our network can provide a reliable approximation to the reference output associated with different

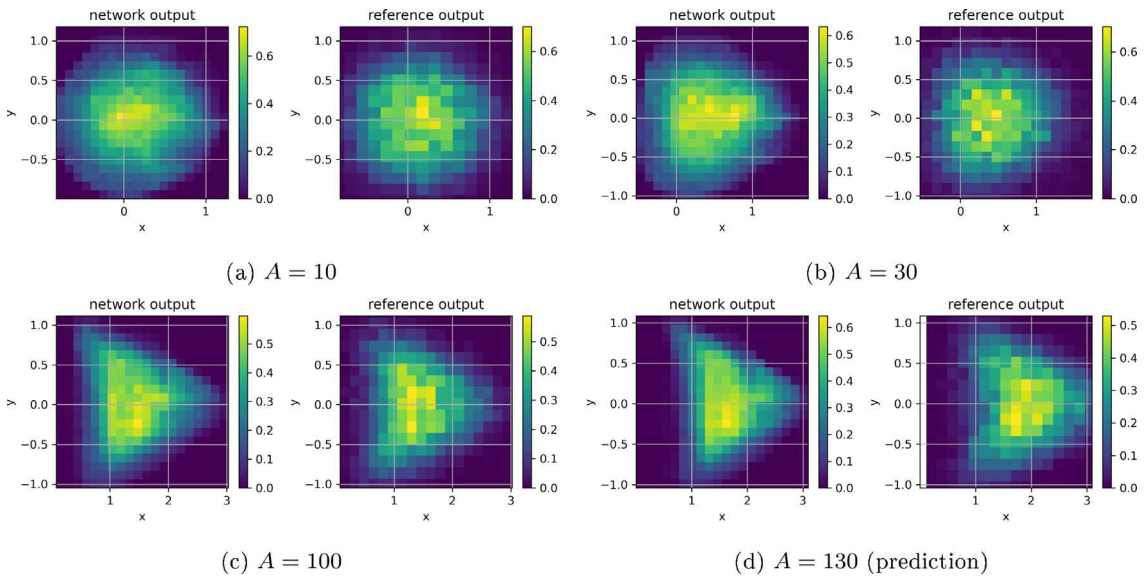
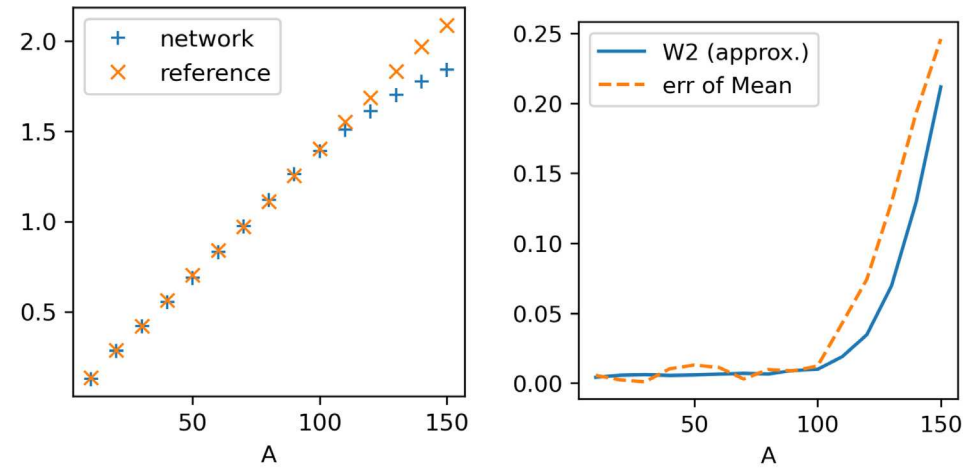


Fig. 6. Network generated vs. reference distributions projected to xy plane in a 3D laminar flow (20).



(a) Mean of the first component of the generation at various A's.

(b) Comparison between the approximated W^2 distance and the sample mean.

Fig. 7. Comparison of the generation at various A's in a 3D laminar flow (20).

A's within a certain range of the largest A in the training set. Then, the network gradually generates larger errors if we further increase A.

The 3D Kolmogorov flow. In the second example, we investigate the case when the organism travels and aggregates in chaotic streamlines given by the Kolmogorov flow [27,37,38]:

$$\mathbf{v}(x, y, z) = A \begin{pmatrix} \sin(2\pi z) \\ \sin(2\pi x) \\ \sin(2\pi y) \end{pmatrix}. \tag{21}$$

In Fig. 8, we compare the distributions generated by the network method and the reference solver (i.e., the IPM) when $A = 100$ and $t = 0.1$. This demonstrates that our network method, after learning from discrete empirical data, is capable of generating continuous distributions. And in Fig. 9, we compare the distributions associated with various amplitude A's when projected to xy plane. The distributions are in general a radial distribution and the radius of the distribution increases when A increases (see also the second moment plot in Fig. 10). The phenomenon differs from the one in laminar flow. This may result from the mixing mechanism of the chaotic flow that spreads and acts against the chemotaxis aggregation.

Finally, we summarize the performance of our algorithms measured by the Wasserstein distance in the validation and prediction data set in Table 2 as follows. These results show that the DeepParticle method is an efficient method for learning and generating aggregation patterns in 2D and 3D KS chemotaxis systems.

5. Conclusions and future works

We proposed a regularized interacting particle method (IPM) to compute aggregation patterns and near singular solutions in multi-dimensional KS systems. We then studied a DeepParticle method to learn and generate solutions for KS systems with dependence on physical parameters (e.g. the flow amplitude in the advection-dominated regime and the evolution time) by minimizing the 2-Wasserstein distance between the source and target distributions. During the training stage, we seek a mapping in the form of a deep neural network from source to target distributions and update network parameters based

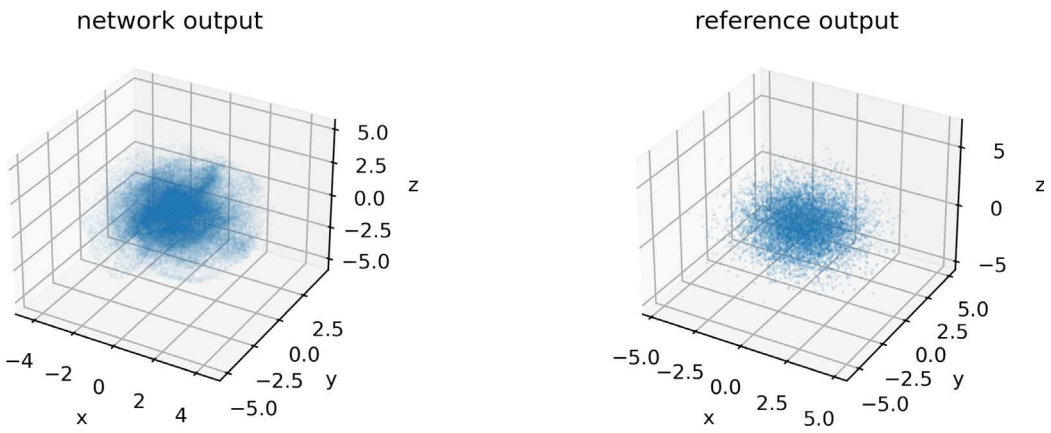


Fig. 8. Network output ($N = 1M$) vs. training data ($N = 10K$) for the Kolmogorov flow (21) with $A = 100$, where the evolution time $t = 0.1$.

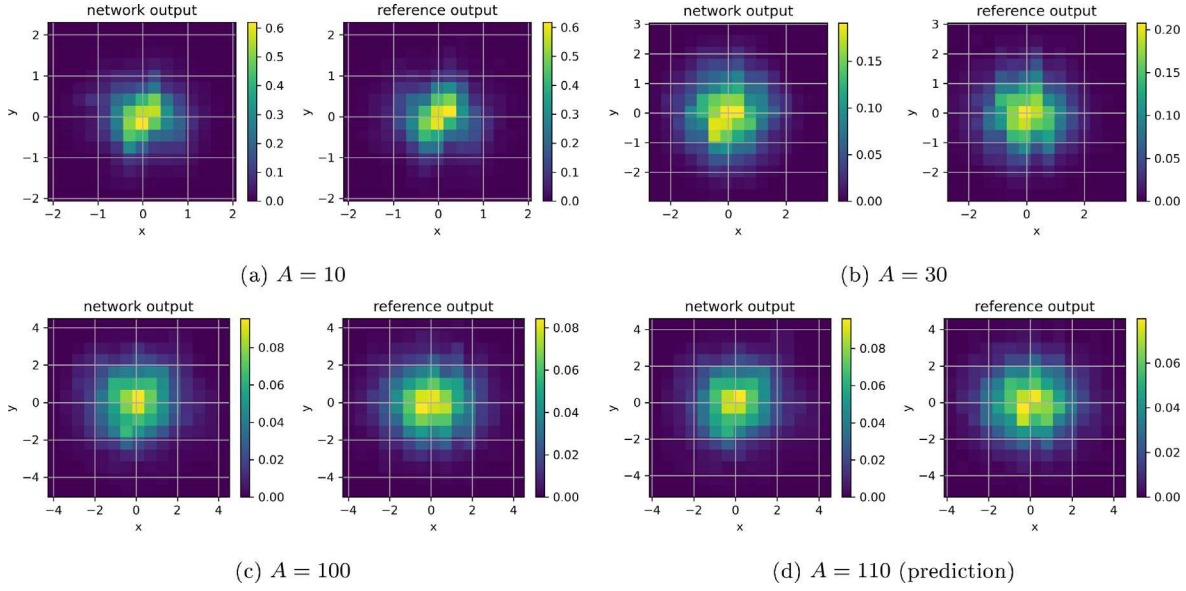


Fig. 9. Network generated vs. reference distributions projected to xy plane in the 3D Kolmogorov flow (21).

Table 2
Wasserstein distance between the network output and reference output in various cases.

Model	Validation config.	W^2	Prediction config.	W^2
2D Laminar $A = 0$	$t = 0.05$	0.0086	$t = 0.12$	0.0120
2D Laminar $A = 100$	$t = 0.05$	0.0116	$t = 0.12$	0.0190
2D Laminar $t = 0.02$	$A = 50$	0.0041	$A = 130$	0.0311
3D Laminar $t = 0.02$	$A = 50$	0.0217	$A = 130$	0.0946
3D K flow $t = 0.02$	$A = 50$	0.0410	$A = 110$	0.0619

on a discretized 2-Wasserstein distance defined on finite distribution samples. Our method is general in the sense that we do not require target distributions to be in closed form and the generation map to be invertible. Our method is fully data-driven and applicable to the fast generation of distributions for more general KS systems with physical parameter dependency. Our iterative divide-and-conquer algorithm reduces considerably the computational cost of finding the optimal transition matrix in the Wasserstein distance. We carried out numerical experiments to demonstrate the performance of our method for learning and generating aggregation patterns in 2D and 3D KS chemotaxis systems without and with laminar and chaotic advection.

In future work, we plan to study the DeepParticle method to learn and generate pattern-forming solutions of parabolic type KS systems ($\epsilon > 0$ in (1)) among other KS like (e.g. chemotaxis-haptotaxis) systems for modeling and predicting cancer cell evolution [39].

CRediT authorship contribution statement

Zhongjian Wang: Conceptualization, Programming, Methodology, Writing – original draft & editing. **Jack Xin:** Conceptualization, Methodology, Writing – review & editing. **Zhiwen Zhang:** Conceptualization, Methodology, Writing – review & editing.

Declaration of competing interest

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

Data availability

Data will be made available on request.

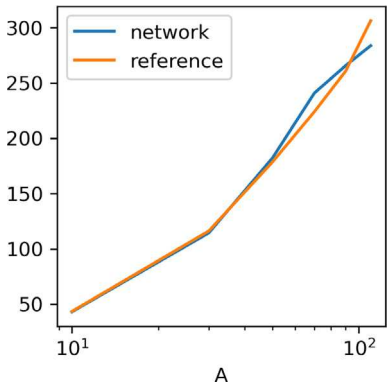


Fig. 10. The second moment of distributions generated by the DeepParticle method and the reference IPM vs. the amplitude A of the 3D Kolmogorov flow (21).

Acknowledgments

The research of ZW is partially supported by NTU, Singapore SUG-023162-00001. The research of JX is partially supported by National Science Foundation (NSF) grants DMS-1952644, and DMS-2309520. The research of ZZ is supported by the National Natural Science Foundation of China (Project 12171406), the Hong Kong RGC grant (Projects 17300318 and 17307921), Seed Funding Programme for Basic Research (HKU), the outstanding young researcher award of HKU (2020–21), and Seed Funding for Strategic Interdisciplinary Research Scheme 2021/22 (HKU). The authors would like to thank Prof. John Lowengrub at UC Irvine and Prof. Philip Maini at Oxford University for helpful discussions of chemotaxis, cell growth, and cancer modeling.

References

[1] E. Keller, L. Segel, Initiation of slime mold aggregation viewed as an instability, J. Theoret. Biol. 26 (3) (1970) 399–415.
[2] C. Patlak, Random walk with persistence and external bias, Bull. Math. Biol. 15 (1953) 311–338.
[3] H. Othmer, A. Stevens, Aggregation, blowup, and collapse: the ABC's of taxis in reinforced random walks, SIAM J. Appl. Math. 57 (1997) 1044–1081.
[4] I. Fatkullin, A study of blow-ups in the Keller–Segel model of chemotaxis, Nonlinearity 26 (1) (2012) 81.
[5] A. Kiselev, L. Ryzhik, Biomixing by chemotaxis and enhancement of biological reactions, Commun. PDE 37 (2012) 298–318.
[6] A. Kiselev, X. Xu, Suppression of chemotactic explosion by mixing, Arch. Ration. Mech. Anal. 222 (2) (2016) 1077–1112.
[7] S. Khan, J. Johnson, E. Cartee, Y. Yao, Global regularity of chemotaxis equations with advection, Involve 9 (1) (2016) 119–131.
[8] G. Iyer, X. Xu, A. Zlatoš, Convection-induced singularity suppression in the Keller-Segel and other non-linear PDEs, Trans. Amer. Math. Soc. 374 (2021) 6039–6058.
[9] S. He, E. Tadmor, A. Zlatoš, On the fast spreading scenario, Commun. Amer. Math. Soc. 2 (2022) 149–171.
[10] T. Hou, R. Li, Dynamic depletion of vortex stretching and non-blowup of the 3D incompressible Euler equations, J. Nonlinear Sci. 16 (6) (2006) 639–664.
[11] T. Hou, Z. Lei, On the stabilizing effect of convection in 3D incompressible flow, Comm. Pure Appl. Math. 62 (4) (2009) 501–564.
[12] Z. Wang, J. Xin, Z. Zhang, DeepParticle: learning invariant measure by a deep neural network minimizing wasserstein distance on data generated by an interacting particle method, J. Comput. Phys. 464 (2022) 111309.
[13] R. Krasny, Vortex sheet computations: roll-up, wakes, separation, Lect. Appl. Math. 28 (1) (1991) 385–401.
[14] K. Craig, A. Bertozzi, A blob method for the aggregation equation, Math. Comp. 85 (300) (2016) 1681–1717.
[15] M. Burger, L. Ruthotto, S. Osher, Connections between deep learning and partial differential equations, European J. Appl. Math. 32 (3) (2021) 395–396.
[16] B. Perthame, PDE models for chemotactic movements: parabolic, hyperbolic and kinetic, Appl. Math. 49 (6) (2004) 539–564.
[17] A. Chorin, P. Bernard, Discretization of a vortex sheet, with an example of roll-up, J. Comput. Phys. 13 (3) (1973) 423–429.
[18] R. Krasny, Desingularization of periodic vortex sheet roll-up, J. Comput. Phys. 65 (2) (1986) 292–313.
[19] R. Caflisch, O. Orellana, Singular solutions and ill-posedness for the evolution of vortex sheets, SIAM J. Math. Anal. 20 (2) (1989) 293–307.
[20] J. Liu, Z. Xin, Convergence of vortex methods for weak solutions to the 2D Euler equations with vortex sheet data, Commun. Pure Appl. Math. 48 (6) (1995) 611–628.
[21] J. Liu, R. Yang, A random particle blob method for the Keller-Segel equation and convergence analysis, Math. Comp. 86 (304) (2017) 725–745.
[22] J. Haškovec, C. Schmeiser, Stochastic particle approximation for measure valued solutions of the 2D Keller-Segel system, J. Stat. Phys. 135 (1) (2009) 133–151.
[23] J. Carrillo, N. Kolbe, M. Lukáčová, A hybrid mass transport finite element method for Keller–Segel type systems, J. Sci. Comput. 80 (3) (2019) 1777–1804.
[24] J. Shen, J. Xu, Unconditionally bound preserving and energy dissipative schemes for a class of Keller–Segel equations, SIAM J. Numer. Anal. 58 (3) (2020) 1674–1695.
[25] Z. Wang, J. Xin, Z. Zhang, Computing effective diffusivity of chaotic and stochastic flows using structure-preserving schemes, SIAM J. Numer. Anal. 56 (4) (2018) 2322–2344.
[26] J. Lyu, Z. Wang, J. Xin, Z. Zhang, Convergence analysis of stochastic structure-preserving schemes for computing effective diffusivity in random flows, SIAM J. Numer. Anal. 58 (5) (2020) 3040–3067.
[27] Z. Wang, J. Xin, Z. Zhang, Sharp uniform in time error estimate on a stochastic structure-preserving Lagrangian method and computation of effective diffusivity in 3D chaotic flows, Multiscale Model. Simul. 93 (3) (2021) 1167–1189.
[28] Z. Wang, J. Xin, Z. Zhang, Computing effective diffusivities of 3D time-dependent chaotic flows with a convergent Lagrangian numerical method, ESAIM: Math. Model. Numer. Anal. 56 (2022) 1521–1544.
[29] J. Lyu, Z. Wang, J. Xin, Z. Zhang, A convergent interacting particle method and computation of KPP front speeds in chaotic flows, SIAM J. Numer. Anal. 60 (3) (2022) 1136–1167.
[30] T. Zhang, Z. Wang, J. Xin, Z. Zhang, A convergent interacting particle method for computing KPP front speeds in random flows, 2023, arXiv preprint arXiv: 2308.14479.
[31] J. Carrillo, K. Craig, F. Patacchini, A blob method for diffusion, Calc. Var. 58 (53) (2019).
[32] R. Sinkhorn, A relationship between arbitrary positive matrices and doubly stochastic matrices, Ann. Math. Stat. 35 (2) (1964) 876–879.
[33] C. Villani, Topics in Optimal Transportation, vol. 58, American Math. Soc., 2021.
[34] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, vol. 24, Springer Science & Business Media, 2003.
[35] S. Wright, Primal-Dual Interior-Point Methods, SIAM Publications, Philadelphia, 1997.
[36] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 214–223.
[37] S. Childress, A. Gilbert, Stretch, Twist, Fold: The Fast Dynamo, in: Lecture Notes in Physics Monographs, (No. 37) Springer, 1995.
[38] C. Kao, Y.-Y. Liu, J. Xin, A Semi-Lagrangian Computation of Front Speeds of G-equation in ABC and Kolmogorov Flows with Estimation via Ballistic Orbits, Multiscale Model. Simul. 20 (1) (2022) 107–117.
[39] M. Chaplain, G. Lolas, Mathematical modelling of cancer invasion of tissue: Dynamic heterogeneity, Netw. Heterog. Media 1 (3) (2006) 399–439.