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]*Department of Statistics and CCAM, The University of Chicago, Chicago, IL 60637, USA.*
[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 SAR, China.*

Abstract

We study a regularized interacting particle method for computing aggregation patterns and near singular solutions of a Keller-Segal (KS) chemotaxis system in two and three space dimensions, then further develop DeepParticle (DP) method to learn and generate solutions under variations of physical parameters. The KS solutions are approximated as empirical measures of particles which 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 finite time T prior to blowup without assuming 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 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 DP 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 small diffusivity of chemo-attractant or the reciprocal of the flow amplitude in the advection-dominated regime.

AMS subject classification: 35K57, 37M25, 49Q22, 65C35, 68T07.

Keywords: Keller-Segel system, stochastic interacting particle approximation, optimal transportation, deep neural networks.

1. Introduction

Chemotaxis partial differential equations (PDEs) were introduced by Keller and Segel (KS [17]) 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 [28], see [27] for an analysis of basic taxis behaviors (aggregation, blowup and collapse) based on reinforced random walks. Recall a common form of KS model [10]:

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

*Corresponding author
Email addresses: zhongjian@statistics.uchicago.edu (Zhongjian Wang), jxin@math.uci.edu (Jack Xin), zhangzw@hku.hk (Zhiwen Zhang)

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, [19, 20, 18, 15, 12] studied equation (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 [19, 20, 15, 12] and references therein, and related convection induced smoothing in fluids [14, 13]. Eq. (3) is the macroscopic limit (McKean-Vlasov equation) of the interacting particle system below as $J \uparrow \infty$:

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

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

In this work, $\mathbf{v}$ is a prescribed divergence free vector field. We shall approximate ρ of equation (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 [37]) 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 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 similar spirit to the vortex blob method for fluids ([22] and references therein) and [9].

Deep learning tools have been applied broadly for scientific computing in recent years, such as solving PDEs and their inverse problems, see [2, 37] and references therein. DeepParticle [37] 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 training. 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 [37]. 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 details of 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. Concluding remarks are made in section 5.

2. Regularized interacting particle method of KS equation

2.1. Blow up behavior in KS model

We start from the simplest KS model without advection, namely (2) with $\mu = \chi = 1$. This system has been extensively studied by many authors, see survey article [29]. 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 fixed time derivative, i.e.,

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

where 8π is called critical mass of the 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 blow up as $t \to \infty$; 3) if $M < 8\pi$, the system has a global smooth solution.

In [18], an extra advection term is introduced to the KS equation, which reflects the organism movement in some prescribed fluid flows. Then the second moment identity in Eq.(6) and the subsequent blowup vs. global evolution results are no long valid. By comparison principle, [18] shows that if the total mass is smaller than critical mass, the system has 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, [12] 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 $\mathbf{v}(\mathbf{x}) = (v(\mathbf{x}_-), \mathbf{0}_-)$, with $\mathbf{x}_- = (x_2, \ldots, x_d)$ 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 [8] 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 [21] 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 [3] . Nonetheless, as the regularization parameter approaches zero, the regularized vortex sheet solution converges to a weak solution of the Euler equation [23]. 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 represent the density function (solution of Eq.(3)) with empirical distributions of particle positions. In SDE (4) on particle positions, the chemo-attractant term $\frac{\chi M}{J}\nabla_{X^j}\sum_{i=1:J,\,i\neq j}\mathcal{K}(|X^j-X^i|)\,dt$ causes numerical instabilities when particles tend to concentrate. To overcome this difficulty, we replace the singular kernel in (4) by a smoothed approximation K_δ, 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 kernel K_δ, we obtain a system of regularized SDEs for the particles as follows:

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

where J is the number of particles, $X^j\in R^d$ is the position of the j-th particle, where dW^j 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 [24]. For a stochastic particle method using heuristic collision and splitting rules to bypass the singular behavior of Green's function, see [11].

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 e.g. finite element [5] and spectral [31] 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 diffusivites in chaotic or random flows [34, 25, 35, 36] and KPP front speeds in chaotic flows [26]. There are also deterministic particle methods ([4, 9] and references therein) for a class of KS and degenerate diffusion equations that fall in our DeepParticle framework (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 (evolution time T and amplitude of advection A). Also, the numbers of particles J cannot be too large as the chemo-attractant term has $O(J^2)$ complexity in each step. 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 machine learning algorithm to learn the feature of continuous dependency on parameters and generate approximate 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:

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

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 $dist$ denotes the metric (distance) on Y. A straightforward derivation yields:

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

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 $dist$ denotes the metric (distance) on Y. We approximate μ and ν by empirical distribution functions: $\mu = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_i}$, $\nu = \frac{1}{N} \sum_{j=1}^{N} \delta_{y_j}$, where N is the number of samples of distribution. Under the setting of learning distribution from interacting particle methods, we take $N < J$ sub-samples from

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 as *mini-batch* in the machine learning literature.

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

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

Then (10) becomes

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

$\hat{W}(f)$ 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 y' to y by $dist(f(x),y)^2$ 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 network configuration

Note that given any fixed set of $\{x_i\}_{i=1}^N\subset R^d$ and $\{y_j\}_{j=1}^N\subset R^d$ (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 scenario, more than one set of data ($\{x_i\}$ and $\{y_j\}$ consists 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 as $\{x_{i,r}\}$ and $\{y_{j,r}\}$ for $r=1\cdots N_{dict}$. Associated with the r-th data set ($\{x_{i,r}\}$ and $\{y_{j,r}\}$), we assume that there is a physical parameter $\eta_r\in R^p$. To represent this in the network, we encode η_r to each data in the set, i.e., the input of the network is $\{(x_{i,r},\eta_r)\}_i\subset R^{d+p}$. This procedure is also called *padding* in the literature. The output of network is then denoted as $f_\theta(x;\eta)$ where θ's are all trainable parameters of the network.

Between the input (l_0) and output (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.,

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

where W_i is (weight) matrix width of l_{i+1} and l_i; b_i is the (bias) vector with same dimension as l_i; tanh is the activation function. For the output layer, the formula is similar except we do not apply the activation function. In case of $d=2$ and $p=1$ (e.g. the first numerical example in computing blowup behaviour of KS without advection, see Sec.4.1), there are in total 4966 parameters (weight and bias) to update by gradient descent. In 3D cases, we find that the network performs well even without altering the width of latent layers.

Remark 3.0.1. *Comparing with par-net approach of [37], we found that padding achieves similar performance when learning distribution in particle simulation of KS model.*

In our computation, by equipping $\mathbb{R}^d$ with Euclidean metric, the training loss function is

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

The goal of deep particle algorithm is then to find $(\theta, \{\gamma_r\})$ to minimize

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

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 linear programming with 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 k_r dependence in f_θ.

The problem (14) is a linear program 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 [30] 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 $\{1, ..., N\}$, 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 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 $\{1, 2, \cdots, N\}$ without replacement. Then j_k is selected such that $\gamma_{i_k, j_k} = 1$. The cost function of the sub-problem is

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

subject to

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

Then γ^* is again a permutation matrix. The linear programming sub-problem of much smaller size is solved by the interior point method [38]. In addition, as the goal is to find a permutation matrix, we set the tolerance of interior point to be relatively large

and project the resulting approximation to a permutation matrix. In our approach, we terminate the iteration of 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 minimum 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.

Comparing with Wasserstein GAN proposed in [1], (15) is a Min-Min optimization. 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 minimum of (15).

The 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 from some target distribution ν without assumption of closed-form distribution of ν. At this stage, the input data is no longer limited by training data, 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 8GB GPU at UC Irvine. The training data is collected from the first order explicit IPM which solves the regularized SDE system (8) by Euler's method in time. IPM is also the reference solver for evaluating DeepParticle output.

4. Numerical Examples

4.1. 2D KS Simulation and Generation in the Absence of Advection

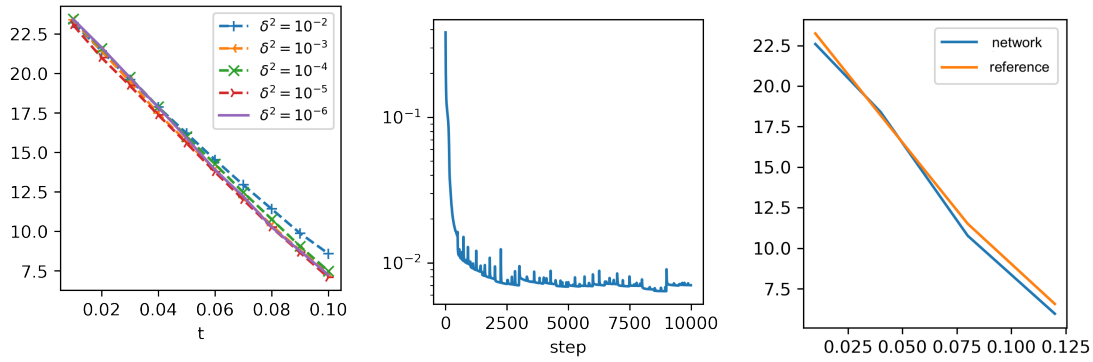
First we consider the KS model without advection, namely (2) with $\mu = \chi = 1$. A straightforward derivation shows if the initial mass $M > 8\pi$ and has finite second moment, the system will blow up in finite time.

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 radius 1 centered at the origin. Assuming the total mass is 16π, we have then the initial second moment as 8π, same as in [18]. By Eq.(6), the system will blow-up when $t > 0.125$. For training data, we applied IPM with regularization $\delta^2 = 1e-3$ for $T = 0.1$ with $J = 10000$ particles. We keep the snapshots of the empirical distribution every 0.05 interval. During training, we consider a mini-batch of size 8×2000, which means, we take $N_{dict} = 8$ sets of training data at various time t and in each set (mini-batch) we have $N = 2000$ subsamples from $J = 10000$ samples from IPM. Adams gradient descent is applied to learn the parameters (weight and bias) in the network. We renew the mini-batch every 1000 steps and renew γ every 200 steps.

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 IPM) with respect to physical parameter η_r;
    $\gamma_{ij,r} = \delta_{i,j}$, i.e., initialize as 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 $|\delta P_r|_{fro} < tol$ **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) to get γ_r^* 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 step size;
    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) to get γ_r^* 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

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 factors. We see that except for the value $\delta^2 = 10^{-2}$, the smaller regularization factor δ does not affect the accuracy of the IPM simulation when $t \in [0, 0.1]$. In Fig.1(b), we show the training loss computed by Eq.(14). Since we do not renew γ at every steps of 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 Fig.1(c), we plot the second moment of the particles by IPM solver (denoted as reference) and by our DP (denoted as network) prediction. We see that the slope of second moment by reference IPM solver deviates from the network generated line after $t = 0.075$. This is due to regularization $\delta > 0$ in IPM. For training data, we rely on IPM solver most of the time when it has good accuracy prior to the time when δ effect kicks in. In Fig.2, we plot



(a) Second moment from IPM with various regularization factors

(b) Training loss

(c) Second moment comparison

Figure 1: Performance of IPM (reference) and DeepParticle (network) algorithms.

the histogram of the output (1M particles) and reference distribution (10K particles) at different time t. We see that the network learns the feature of particle concentration and even has more concentrated output than the reference solver near the blowup time.

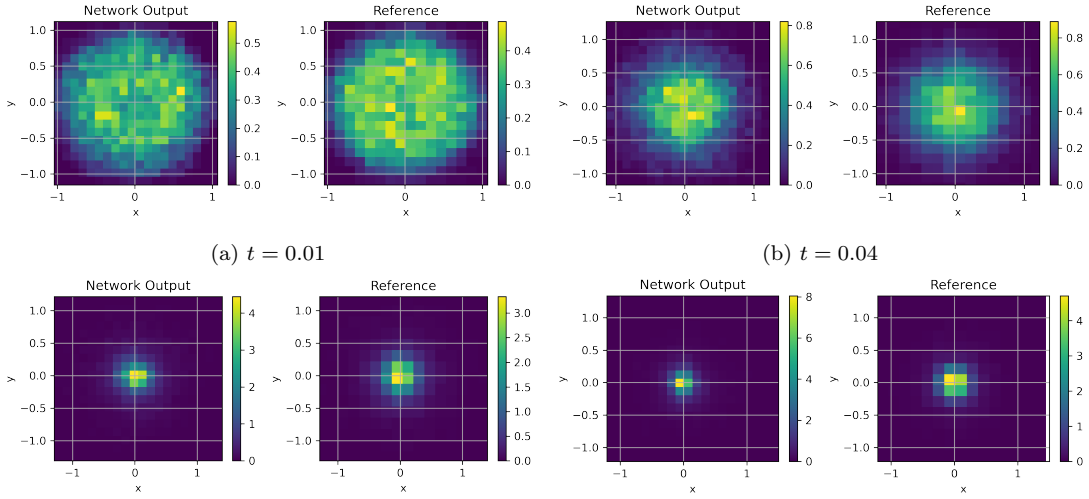
4.2. KS Simulation and Generation in the Presence of 2D Laminar Flows

Next we consider $\mathbf{v} \neq 0$, so in addition to chemotaxis gradient, the movement of organism is also driven by some given environmental fluid velocity field. In [18], blow-up behavior of KS model given various strength 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 parameters to learn in the model: the advection amplitude A and the evolution time T.

Learning dependence on A. We start from learning amplitude dependence while fixing the evolution time. In this example, we used IPM to generate $J = 10000$ samples of solution after $T = 0.02$ with $A = 10^{[0:0.2:2]}$. The initial distribution of IPM for each A is

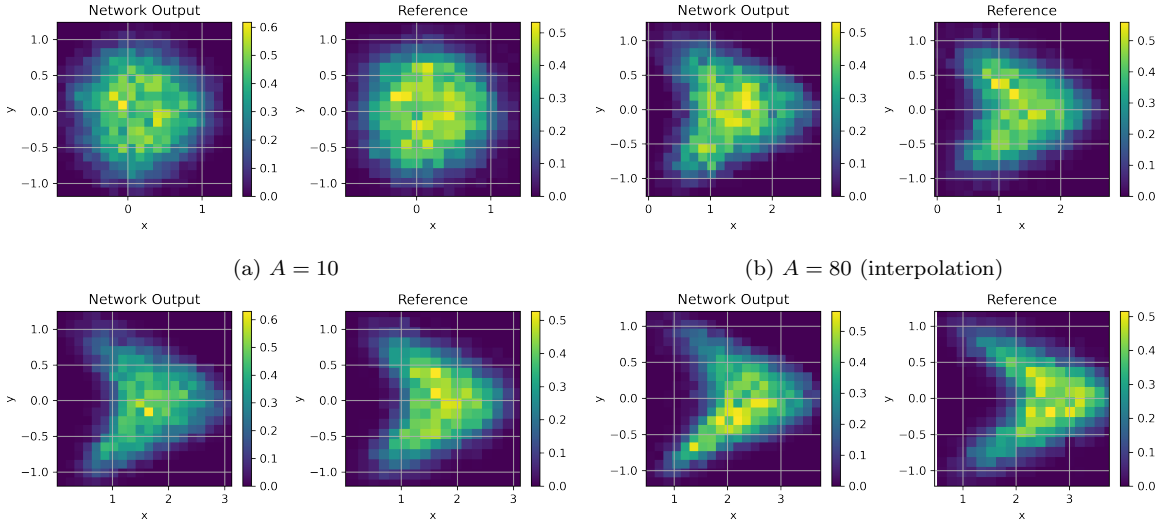


(a) $t = 0.01$ (b) $t = 0.04$

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

Figure 2: Comparison of particle distributions from DeepParticle (prediction) and IPM (reference) solver at different times and $A = 0$.

a uniform distribution on the unit ball. In Fig.3 we see that the distribution turns to a V shape as A increases. This can be attributed to the stretching effect by the laminar flow (19). Our network is shown to learn this feature. In addition, our network can also predict distribution when A is slightly outside the range of training set, see the case when $A = 150$ in Fig.3(d). More precisely, Fig.3(c) shows that most of 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.



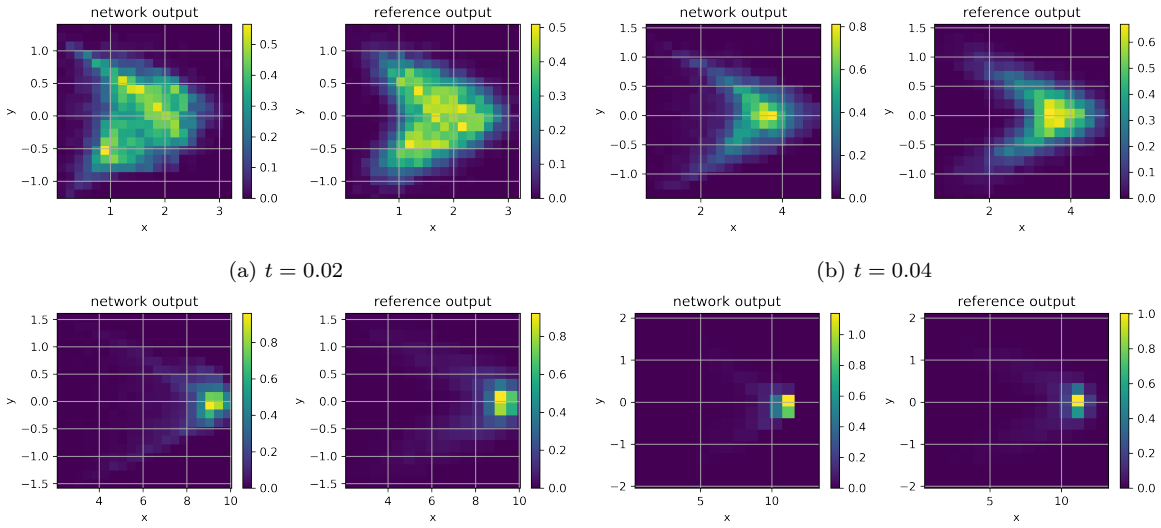
(a) $A = 10$ (b) $A = 80$ (interpolation)

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

Figure 3: Learning particle aggregation at different A values with $t = 0.02$ fixed.

Learning dependence on T. We turn to study the dependence on the evolution time at fixed $A = 100$. To generate training data, we run IPM with $t \in [0, 0.1]$ and $J = 10000$ particles, and take snapshots of the empirical distribution at $t = 0, 0.01, 0.02, \cdots, 0.1$. In Fig.4, we compare our network prediction with IPM generated reference solution at

various evolution times. We see that both IPM and DP methods reproduce the near blow-up behaviors consistent with [18].



(a) $t = 0.02$ (b) $t = 0.04$

(c) $t = 0.1$ (largest t in training set) (d) $t = 0.14$ (prediction)

Figure 4: Learning particle aggregation at different times with fixed flow amplitude $A = 100$.

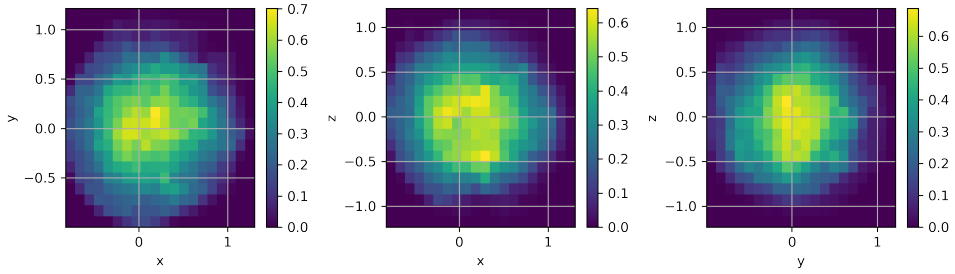
4.3. KS Simulation and Generation in the Presence of 3D flows

In this subsection, we turns to KS model in three dimension. There are two kinds of flows under consideration, the first is 3D laminar flow which is a natural generalization of 2D laminar flow. The second is Kolmogorov flow which is a well-known example of chaotic flow [7, 16].

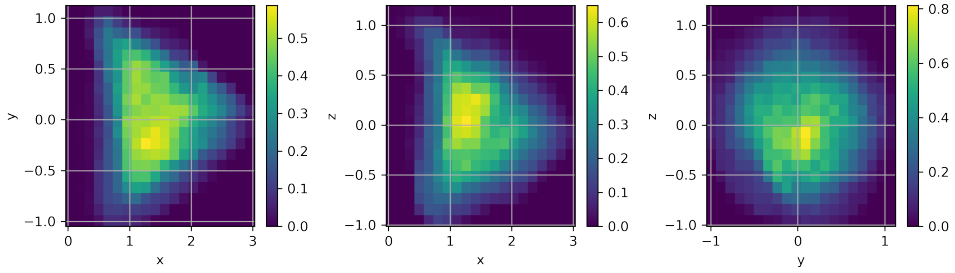
3D Laminar Flow. As the first 3D example, we consider advection field in the form:

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

It describes the organism travelling 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 prediction 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 is now 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 laminar setting of the flow. In yz projection, the distribution remains as radial. In Fig.6, we show the xy projection of the distribution with various A. It confirms that in addition to interpretation, our network is able to generalize the influence of amplitude on the distribution when A is not far from the range in the training set.

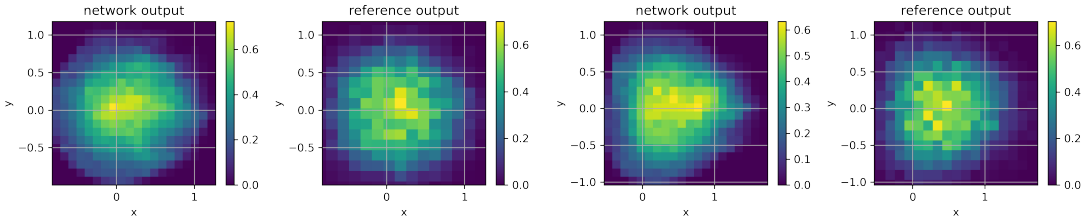


(a) $A = 10$

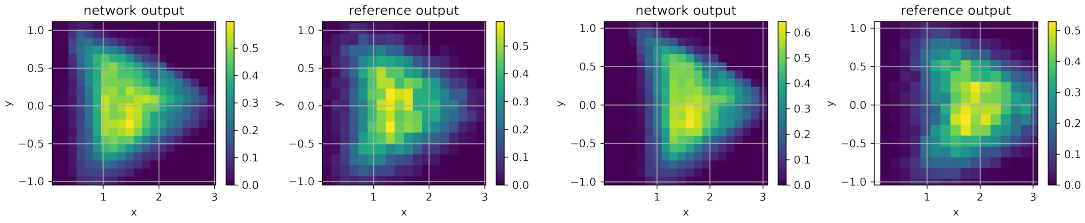


(b) $A = 100$

Figure 5: Generated outputs in three cross sections at different A values in a 3D laminar flow (20).



(a) $A = 10$ (b) $A = 30$



(c) $A = 100$ (d) $A = 130$

Figure 6: Generated vs. reference distributions projected to the xy plane in a 3D laminar flow (20).

3D Kolmogorov Flow. In the last example, we investigate the case when the organism travels and aggregates in chaotic streamlines given by Kolmogorov flow [7, 16]:

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

In Fig.7, we compare the network generated and reference distributions when $A =$

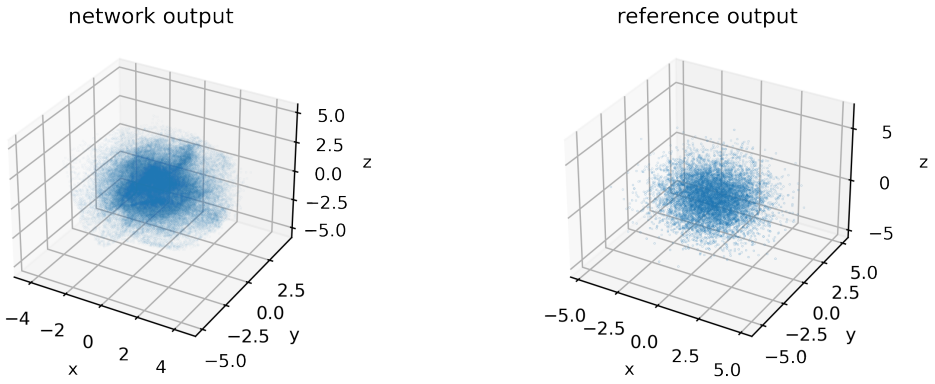
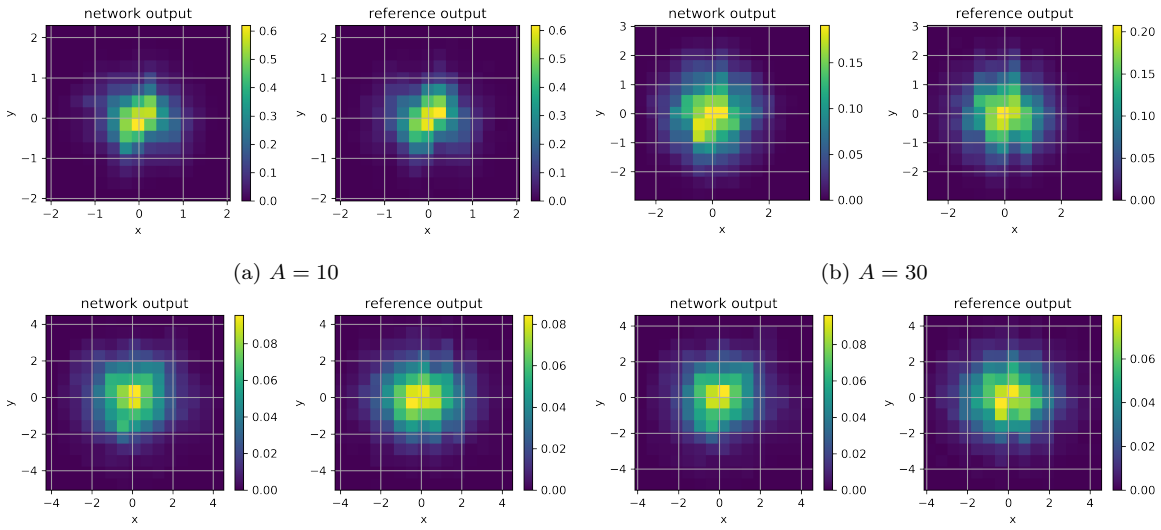


Figure 7: Network output ($N = 1M$) vs. training data ($N = 10K$) in flow (21) at $A = 100$, $t = 0.1$.



(a) $A = 10$

(b) $A = 30$

(c) $A = 100$

(d) $A = 110$

Figure 8: Generated vs. reference distributions projected to the xy plane in a 3D chaotic flow (21).

100 and $t = 0.1$ and Fig.8 compares the distributions with various amplitude A 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.9). 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 chemotaxis aggregation.

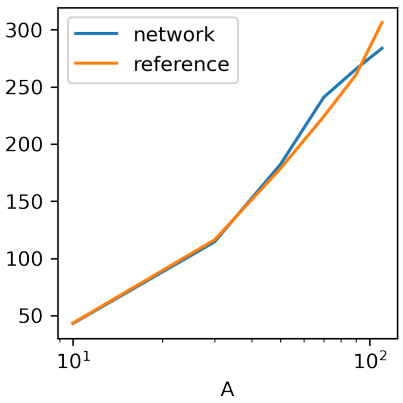


Figure 9: The second moment of generated/reference distribution vs. A of 3D chaotic flow (21).

5. Conclusions and Future Works

We proposed a regularized interacting particle method (IPM) to compute aggregation patterns and near singular solutions in multi-dimensional Keller Segel (KS) systems. We then studied a DeepParticle (DP) method to learn and generate solutions for KS systems with dependence on physical parameters (e.g. the small diffusivity of chemo-attractant and the reciprocal of the flow amplitude in the advection-dominated regime) 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 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 KS aggregation learning and generation without and with laminar and chaotic advections. In future work, we plan to study DP 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 [6].

CRediT authorship contribution statement

Zhongjian Wang: Conceptualization, Programming, Methodology, Writing-Original draft preparation and Editing. **Jack Xin**: Conceptualization, Methodology, Writing-Reviewing and Editing. **Zhiwen Zhang**: Conceptualization, Writing-Reviewing and 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.

Acknowledgements

The research of JX is partially supported by NSF grants DMS-1924548 and DMS-1952644. The research of ZZ is supported by Hong Kong RGC grant (Projects 17300318 and 17307921), National Natural Science Foundation of China (Project 12171406), Seed Funding Programme for Basic Research (HKU) 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 cell growth and cancer modeling.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[2] M. Burger, L. Ruthotto, and S. Osher. Connections between deep learning and partial differential equations. *European Journal of Applied Mathematics*, 32(3):395–396, 2021.

[3] R. Caflisch and O. Orellana. Singular solutions and ill-posedness for the evolution of vortex sheets. *SIAM Journal on Mathematical Analysis*, 20(2):293–307, 1989.

[4] J. Carrillo, K. Craig, and F. Patacchini. A blob method for diffusion. *Calculus of Variations*, 58(53), 2019.

[5] J. Carrillo, N. Kolbe, and M. Lukáčová. A hybrid mass transport finite element method for Keller–Segel type systems. *Journal of Scientific Computing*, 80(3):1777–1804, 2019.

[6] M. Chaplain and G. Lolas. Mathematical modelling of cancer invasion of tissue: Dynamic heterogeneity. *Networks and Heterogeneous Media*, 1(3):399–439, 2006.

[7] S. Childress and A. Gilbert. *Stretch, Twist, Fold: The Fast Dynamo*. Lecture Notes in Physics Monographs, No. 37, Springer, 1995.

[8] A. Chorin and P. Bernard. Discretization of a vortex sheet, with an example of roll-up. *Journal of Computational Physics*, 13(3):423–429, 1973.

[9] K. Craig and A. Bertozzi. A blob method for the aggregation equation. *Math. Comp.*, 85(300):1681–1717, 2016.

[10] I. Fatkullin. A study of blow-ups in the Keller–Segel model of chemotaxis. *Nonlinearity*, 26(1):81, 2012.

[11] J. Haškovec and C. Schmeiser. Stochastic particle approximation for measure valued solutions of the 2D Keller-Segel system. *Journal of Statistical Physics*, 135(1):133–151, 2009.

[12] S. He, E. Tadmor, and A. Zlatoš. On the fast spreading scenario. *Comm. Amer. Math. Soc.*, 2:149–171, 2022.

[13] T. Hou and Z. Lei. On the stabilizing effect of convection in 3D incompressible flow. *Commun. Pure Appl. Math.*, 62(4):501–564, 2009.

[14] T. Hou and R. Li. Dynamic depletion of vortex stretching and non-blowup of the 3D incompressible Euler equations. *Journal of Nonlinear Science*, 16(6):639–664, 2006.

[15] G. Iyer, X. Xu, and A. Zlatoš. Convection-induced singularity suppression in the Keller-Segel and other non-linear PDEs. *Trans. Amer. Math. Soc.*, 374:6039–6058, 2021.

[16] C. Kao, Y-Y Liu, and J. Xin. A Semi-Lagrangian Computation of Front Speeds of G-equation in ABC and Kolmogorov Flows with Estimation via Ballistic Orbits. *Multiscale Modeling and Simulation*, 20(1):107–117, 2022.

[17] E. Keller and L. Segel. Initiation of slime mold aggregation viewed as an instability. *Journal of theoretical biology*, 26(3):399–415, 1970.

[18] S. Khan, J. Johnson, E. Cartee, and Y. Yao. Global regularity of chemotaxis equations with advection. *Involve*, 9(1):119–131, 2016.

[19] A. Kiselev and L. Ryzhik. Biomixing by chemotaxis and enhancement of biological reactions. *Communications in PDE*, 37:298–318, 2012.

[20] A. Kiselev and X. Xu. Suppression of chemotactic explosion by mixing. *Arch. Ration. Mech. Anal.*, 222(2):1077–1112, 2016.

[21] R. Krasny. Desingularization of periodic vortex sheet roll-up. *Journal of Computational Physics*, 65(2):292–313, 1986.

[22] R. Krasny. Vortex sheet computations: roll-up, wakes, separation. *Lectures in Applied Mathematics*, 28(1):385–401, 1991.

[23] J. Liu and Z. Xin. Convergence of vortex methods for weak solutions to the 2D Euler equations with vortex sheet data. *Communications on Pure and Applied Math*, 48(6):611–628, 1995.

[24] J. Liu and R. Yang. A random particle blob method for the Keller-Segel equation and convergence analysis. *Mathematics of Computation*, 86(304):725–745, 2017.

[25] J. Lyu, Z. Wang, J. Xin, and Z. Zhang. Convergence analysis of stochastic structure-preserving schemes for computing effective diffusivity in random flows. *SIAM J. Numer. Anal.*, 58(5):3040–3067, 2020.

[26] J. Lyu, Z. Wang, J. Xin, and Z. Zhang. A convergent interacting particle method and computation of KPP front speeds in chaotic flows. *SIAM J. Numer. Anal.*, 60(3):1136–1167, 2022.

[27] H. Othmer and A. Stevens. Aggregation, blowup, and collapse: the ABC's of taxis in reinforced random walks. *SIAM J. Appl. Math*, 57:1044–1081, 1997.

[28] C. Patlak. Random walk with persistence and external bias. *Bull. Math. Biol.*, 15:311–338, 1953.

[29] B. Perthame. PDE models for chemotactic movements: parabolic, hyperbolic and kinetic. *Applications of Mathematics*, 49(6):539–564, 2004.

[30] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[31] J. Shen and J. Xu. Unconditionally bound preserving and energy dissipative schemes for a class of Keller–Segel equations. *SIAM Journal on Numerical Analysis*, 58(3):1674–1695, 2020.

[32] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

[33] C. Villani. *Topics in optimal transportation*, volume 58. American Math. Soc., 2021.

[34] Z. Wang, J. Xin, and Z. Zhang. Computing effective diffusivity of chaotic and stochastic flows using structure-preserving schemes. *SIAM J. Numer. Anal*, 56(4):2322–2344, 2018.

[35] Z. Wang, J. Xin, and 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 Modeling and Simulation*, 93(3):1167–1189, 2021.

[36] Z. Wang, J. Xin, and Z. Zhang. Computing effective diffusivities of 3D time-dependent chaotic flows with a convergent Lagrangian numerical method. *ESAIM: Mathematical Modeling and Numerical Analysis*, 56:1521–1544, 2022.

[37] Z. Wang, J. Xin, and Z. Zhang. DeepParticle: learning invariant measure by a deep neural network minimizing Wasserstein distance on data generated by an interacting particle method. *J. Computational Physics*, 464:111309, 2022.

[38] S. Wright. *Primal-dual interior-point methods*. SIAM Publications, Philadelphia, 1997.