Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Learning High-Order Geometric Flow Based on The Level Set Method

Chun Li (lichun2020@hit.edu.cn)
Harbin Institute of Technology Shenzhen https://orcid.org/0000-0002-2021-751X
Yunyun Yang
Harbin Institute of Technology Shenzhen
Hui Liang
Harbin Institute of Technology Shenzhen
Boying Wu
Harbin Institute of Technology

Research Article

Keywords: High-Order Geometric Flow, Physics constrained learning, deep learning, level set method

Posted Date: July 8th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-594711/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Version of Record: A version of this preprint was published at Nonlinear Dynamics on January 23rd, 2022. See the published version at https://doi.org/10.1007/s11071-021-07043-5.

Nonlinear Dynamics manuscript No.
(will be inserted by the editor)

Learning High-Order Geometric Flow Based on The Level Set Method

Chun Li [1] · **Yunyun Yang**[1,*] · **Hui Liang**[1] · **Boying Wu**[2]

Received: date / Accepted: date

Abstract Recently, the development of deep learning (DL), which has accomplished unbelievable success in many fields, especially in scientific computational fields. And almost all computational problems and physical phenomena can be described by partial differential equations (PDEs). In this work, we proposed two potential high-order geometric flows. Motivation by the physical-information neural networks (PINNs) and the traditional level set method (LSM), we have integrated deep neural networks (DNNs) and LSM to make the proposed method more robust and efficient. Also, to test the sensitivity of the system to different input data, we set up three sets of initial conditions to test the model. Furthermore, numerical experiments on different input data are implemented to demonstrate the effectiveness and superiority of the proposed models compared to the state-of-the-art approach.

Keywords High-Order Geometric Flow · Physics constrained learning · deep learning · level set method

1 Introduction

Recently, the development of artificial intelligence (AI) has received comprehensive attention and assistance through the breakthrough of deep learning (DL) technology. Which has reached a consensus on it is becoming a development hotspot over the world. DL has accomplished unbelievable success in many fields, especially in scientific computational fields, including computer vision, medical imaging, and control problem [1–6]. Fortunately, almost all computational problems and physical phenomena can be described by differential equations, especially geometric partial differential equations (GPDEs). Such multi-phase flows play an increasing role in several scientific and engineering applications [7–9]. The PDEs can be solved analytically and numerically. However, so far, the analytical solutions of many PDEs have not been solved. Therefore, in terms of PDEs that have not been analytically solved, we can only understand the physical phenomenon described by them from the perspective of the numerical solution. Usually, the numerical approaches are used to discretized the solution domain and construct algebraic equations, after that, they are solved analytically or iterative. The traditional numerical methods include the finite volume method (FVM), finite difference method (FDM), finite element method (FEM), and so on. The computational cost of the equations becomes extremely expensive when the number of equations increases. Moreover, the GPDEs are generally defined on the surface (manifold), which can fall into the curse of dimensionality. Furthermore, GPDEs solutions can be distinctively different, and there is no general approach that applies to all kinds of GPDEs.

Happily, since the universal approximation theorem (UAT), which is the fundamental theoretical basis of DL. And it opens another door for the numerical solution of GPDEs. According to the UAT, the complex or even dynamical GPDEs can be approximated via a DNN. To find the solutions of them, the DNN is trained on the solution domain of the GPDEs. Moreover, a surface (manifold) can be implicitly represented by a level set method (LSM), which was first introduced by Osher and Sethian [10], and has a significant im-

Chun Li, Yunyun Yang (*Corresponding author), Hui Liang
E-mail: lichun2020, yangyunyun, lianghui@hit.edu.cn
· Boying Wu
E-mail: mathwby@hit.edu.cn)
[1] School of Science, Harbin Institute of Technology, Shenzhen, 518055, China
[2] School of Mathematics, Harbin Institute of Technology, Harbin, 150001, China

pact on the computational field. Currently, DNN has applied successfully for solving PDEs, such as hidden physics models [11] and physics-informed neural networks (PINNs) [12].

In this work, motivated by hidden physics models, PINNs, and LSM, a robust deep LSM learning of the data-driven high-order GPDEs method is proposed. Our contributions are summarized as follows:

- **We focus on the data-driven and learning methods to solve two GPDEs: (1) Quasi Xuguo flow [13], (2). High-order surface diffusion flow of Cahn–Hilliard model.**
- **Theoretically, our framework is flexible to adapt to different high-order GPDEs, which enables effective integration of traditional LSM and DNN to improve computational efficiency. Our experiments confirm this property.**

The rest of the paper is organized as follows, in Section 2, we first present some related works about learning high-order geometric flow. Furthermore, we introduce the proposed algorithm about learning high-order geometric flow based on the level set method in Section 3. After that, in Section 4, we describe the experimental settings and experimental results. Finally, we summarize this paper and present the limitations of the current study, and present several future research directions in Section 5.

2 Related Work

In this section, we briefly review the most relevant studies from the following three aspects: 1) The manifold learning and some definitions, 2) The level set method, and 3) The deep learning-based approach for solving PDEs.

2.1 Manifold learning and some definitions

Manifold learning is a method for nonlinear dimension reduction. Algorithms for this task think that the dimension of several data sets is only artificially high. Set $S = \left\{u\left(x,y\right) \in R^3 : (x,y) \in D \in R^2\right\}$ be a sufficiently smooth, regular and the parametric surface. And let $g = \langle u_x, u_y \rangle$ be the coefficients of the first and second fundamental forms of surfaces with $u_x = \frac{\partial u}{\partial x}, u_y = \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial x \partial y} = u_{xy}$, where $\langle \cdot, \cdot \rangle$ refers to the usual inner product in Euclidean space R^3.

Definition 1 (**Tangential gradient operator**). Suppose $|\nabla \phi| \neq 0$ on some open neighborhood Ω of the level set $\Gamma = \{(x,y) : \phi\left(x,y,t\right) = 0\}$, f is a differentiable function on Ω, then the tangential gradient operator ∇_S acting on f is given by $\nabla_S f = P \nabla f$, where $P = \mathbf{I} - \mathbf{n} \otimes \mathbf{n} = \mathbf{I} - \mathbf{n}\mathbf{n}^T$ is the projection operator to the tangential plane of the surface Γ, $\mathbf{n} = \frac{\nabla \phi}{|\nabla \phi|}$, and $\mathbf{I}$ refers to the identify mapping.

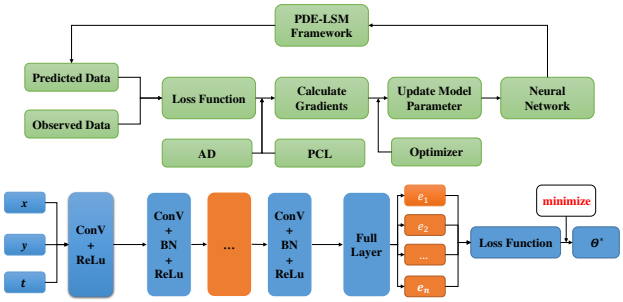


Fig. 1: Schematic illustration of our LSM-Physics Constrained Learning (PCL) framework for learning GPDE.

Definition 2 (**Tangential divergence operator**). Suppose $|\nabla \phi| \neq 0$ on some open neighborhood Ω of the level set $\Gamma = \{(x,y) : \phi\left(x,y,t\right) = 0\}$, $\mathbf{v}$ is a smooth vector field defined on Ω, the divergence operator div_S acting on $\mathbf{v}$ is given by $\mathrm{div}_S\left(\mathbf{v}\right) = \mathrm{div}\left(\mathbf{v}\right) - \mathbf{n}^T\left(\nabla \mathbf{v}\right)\mathbf{n}$, where div is the usual divergence operator.

Definition 3 (**Laplace-Beltrami operator (LBO)**). Suppose $|\nabla \phi| \neq 0$ on some open neighborhood Ω of the level set $\Gamma = \{(x,y) : \phi\left(x,y,t\right) = 0\}$, f is twice differentiable function on Ω, then the Laplace-Beltrami operator Δ_S acting on f is given by $\Delta_S f = \mathrm{div}_S\left(\nabla_S f\right)$.

Recently, manifold learning has made great progress. For example, Bachmann et al. [1] developed constant curvature graph convolutional networks (GNN), bridging the gap that popular GNNs consider the data only via Euclidean geometry and associated vector space operations. Also, Sanchez-Gonzalez et al. [3] used the GNN to simulate complex physical phenomenon. Moreover, Chen et al. [14] utilized the convolutional kernel networks for learning the graph-structured data. Other related works about manifold learning, include and not only include the following references [15–19].

2.2 Level set method

The LSM was first developed by Osher and Sethian [10], which made a huge impact on computational methods for interface motion. Since the surface manifold can be

implicitly represented by the LSM, therefore, which is widely used in various fields including computational geometry, fluid mechanics, computer vision, and materials science [20]. Moreover, Fedkiw et al. [21] used the LSM for representing the dynamic implicit surfaces. More recently, Lin et al. [22] developed the LSM for solving constrained convex optimization. More about the LSM please refer to [23–27].

2.3 Deep learning-based approach for solving PDEs

Recently, with the rapid development of DL, the numerical methods of PDEs have made significant progress. Advantageously, as mesh-free approximators, compared with the traditional mesh-based numerical schemes include FVM, FDM, and FEM, the DNNs are inherently mesh-free function-approximators. Such that they not only can avoid the curse of dimensionality but also approximate the solutions of PDEs on complex geometries effectively. One remarkable application of DNNs is the physics-informed neural networks (PINNs) [12], which can solve both forward and inverse problems with the desired accuracy. Also, Sirignano et al. [28] developed a method called DGM for solving PDEs. Moreover, Long et al. [2] proposed PDE-Net for learning PDEs from data. For more examples on solving differential equations with DL, please refer to [11, 15, 29–36].

3 The proposed method

3.1 High-Order Quasi Xuguo flow

Algorithm 1: The framework of learning high-order geometric flow based on the LSM.

initialization:
(1) Initialize the parameters Θ, and the learning rate α_n;
(2) Initialize u^0;
for $n = 1, 2, 3, ..., N$ **do**
(1) Read current $L(\Theta)$ via Eq. (5);
(2) Update the parameters Θ via gradient descent

$$\Theta_{n+1} = \Theta_n - \alpha_n \nabla_\Theta L_r(\Theta_n) - \alpha_n \sum_{i=1}^{N} \lambda_i \nabla_\Theta L_i(\Theta_n);$$

end
output:
(1) Output the learned solution;
(2) Output the parameters Θ_{n+1};

In this section, we describe more details about the algorithm of learning high-order geometric flow based on the LSM. In [13], Xu and Zhang constructed the Quasi Xuguo flow from the perspective of computational geometry. However, the higher-order geometric flow is not solved analytically and numerically in that paper, according to [13] and LSM, the high-order GPDE is we obtained as follows,

$$\begin{aligned}\phi_t &= \Delta_s^3\left(\operatorname{div}\left(\frac{\nabla\phi}{\|\nabla\phi\|_\varepsilon}\right)\right)\|\nabla\phi\|_\varepsilon \\ &= \Delta_s\left(\Delta_s\left(\Delta_s \operatorname{div}\left(\frac{\nabla\phi}{\|\nabla\phi\|_\varepsilon}\right)\right)\right)\|\nabla\phi\|_\varepsilon,\end{aligned} \tag{1}$$

where Δ_s refers to LBO, $\|\nabla\phi\|_\varepsilon = \sqrt{\phi_x^2 + \phi_y^2 + \varepsilon}$, $\kappa = \operatorname{div}\left(\frac{\nabla\phi}{\|\nabla\phi\|_\varepsilon}\right)$ refers to the mean curvature (MC), and $\varepsilon > 0$. According to [37], take κ as an example, the LBO can be explicitly formulated as following,

$$\Delta_s\kappa = \frac{\kappa_{xx}\left(1+\kappa_y^2\right) + \kappa_{yy}\left(1+\kappa_x^2\right) - 2\kappa_x\kappa_y\kappa_{xy}}{\left(1+\kappa_x^2+\kappa_x^2\right)^2}. \tag{2}$$

Based on Eq. (2), such that Eq. ((1) can be rewritten as follows,

$$\begin{cases} \kappa = \frac{\phi_{xx}\left(1+\phi_y^2\right)+\phi_{yy}\left(1+\phi_x^2\right)-2\phi_x\phi_y\phi_{xy}}{\left(1+\phi_x^2+\phi_x^2\right)^{\frac{3}{2}}}, \\ \phi_t = \Delta_s^3(\kappa)\|\nabla\phi\|_\varepsilon, \\ \Delta_s\kappa = \frac{\kappa_{xx}\left(1+\kappa_y^2\right)+\kappa_{yy}\left(1+\kappa_x^2\right)-2\kappa_x\kappa_y\kappa_{xy}}{\left(1+\kappa_x^2+\kappa_x^2\right)^2}. \end{cases} \tag{3}$$

To obtain the order reduction of GPDEs, the new variables $q = \Delta_s\kappa$, $w = \Delta_s q$ are introduced. Hence, $\Delta_s q$ and $\Delta_s w$ can be explicitly formulated via the same approach as $\Delta_s\kappa$. Therefore, Eq. (1) can be reformulated as follows,

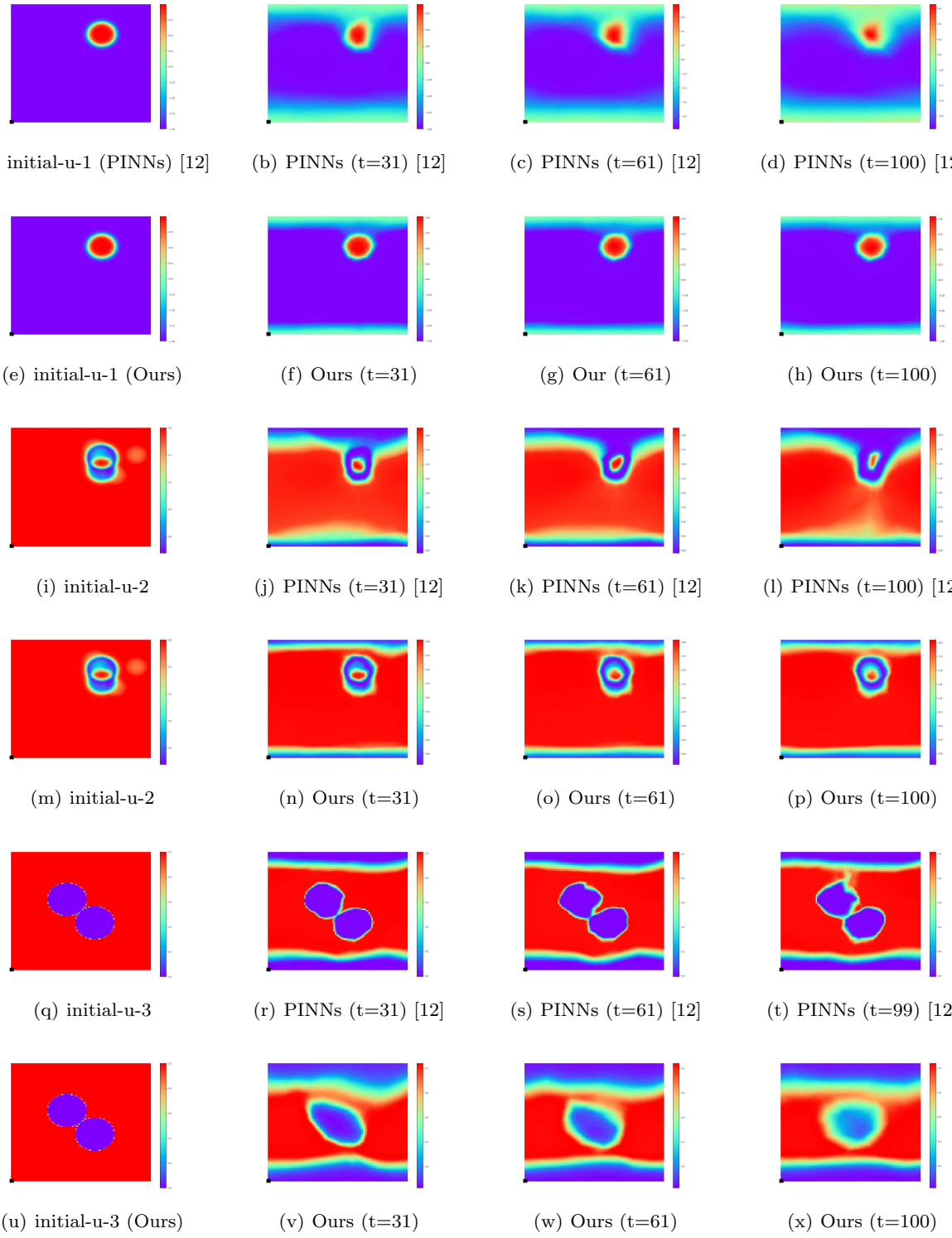
$$\begin{cases} q = \Delta_s\kappa, \\ w = \Delta_s q, \\ \phi_t = \Delta_s w\|\nabla\phi\|_\varepsilon. \end{cases} \tag{4}$$

Motivated by PINNs, the GPDEs are encoded into the loss function, and the partial derivatives can be computational via automatic differentiation (AD). For convenience, the following symbols are introduced, $e_1 = \phi_t - \Delta_s w\|\nabla\phi\|_\varepsilon$, $e_2 = \kappa - \frac{\phi_{xx}\left(1+\phi_y^2\right)+\phi_{yy}\left(1+\phi_x^2\right)-2\phi_x\phi_y\phi_{xy}}{\left(1+\phi_x^2+\phi_x^2\right)^{\frac{3}{2}}}$,

$$e_3 = \Delta_s\kappa - \frac{\kappa_{xx}\left(1+\kappa_y^2\right) + \kappa_{yy}\left(1+\kappa_x^2\right) - 2\kappa_x\kappa_y\kappa_{xy}}{\left(1+\kappa_x^2+\kappa_x^2\right)^2},$$

$$e_4 = \Delta_s q - \frac{q_{xx}\left(1+q_y^2\right) + q_{yy}\left(1+q_x^2\right) - 2q_xq_yq_{xy}}{\left(1+q_x^2+q_x^2\right)^2},$$

$$e_5 = \Delta_s w - \frac{w_{xx}\left(1+w_y^2\right) + w_{yy}\left(1+w_x^2\right) - 2w_xw_yw_{xy}}{\left(1+w_x^2+w_x^2\right)^2},$$



(a) initial-u-1 (PINNs) [12] (b) PINNs (t=31) [12] (c) PINNs (t=61) [12] (d) PINNs (t=100) [12]

(e) initial-u-1 (Ours) (f) Ours (t=31) (g) Our (t=61) (h) Ours (t=100)

(i) initial-u-2 (j) PINNs (t=31) [12] (k) PINNs (t=61) [12] (l) PINNs (t=100) [12]

(m) initial-u-2 (n) Ours (t=31) (o) Ours (t=61) (p) Ours (t=100)

(q) initial-u-3 (r) PINNs (t=31) [12] (s) PINNs (t=61) [12] (t) PINNs (t=99) [12]

(u) initial-u-3 (Ours) (v) Ours (t=31) (w) Ours (t=61) (x) Ours (t=100)

Fig. 2: Examples of the visualization process of evolutionary for high-order Quasi Xuguo flow with initial conditions 1-3 for PINNs and Ours.

Table 1: The comparison of the training loss values and times for high-order Quasi Xuguo flow with different initial conditions.

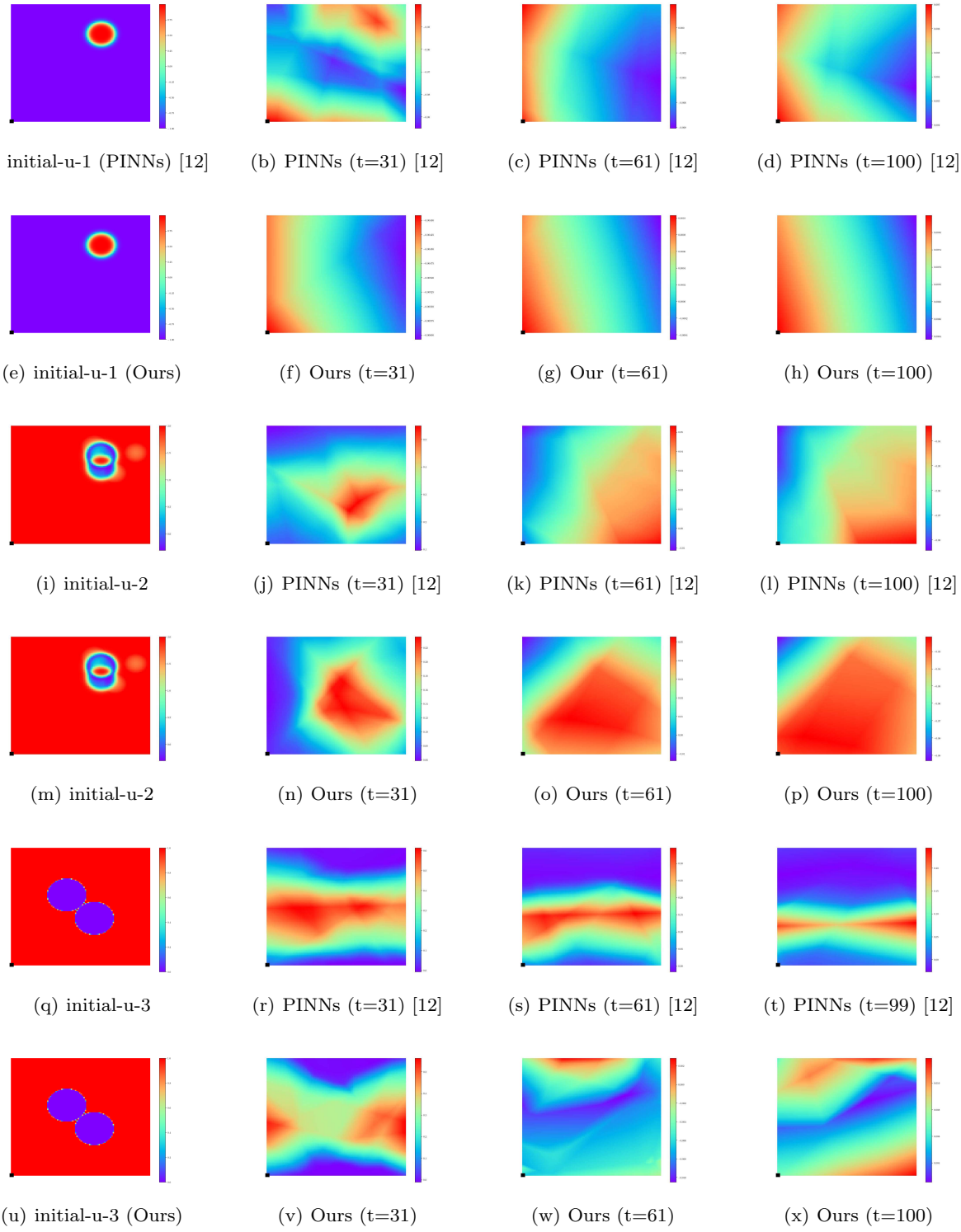
Initial condition	**Training loss values**	**Training times (s)**
initial condition-1		
PINNs [12]	0.1574	267.8658
Ours	**0.1306**	309.3411
initial condition-2		
PINNs [12]	0.7473	269.5971
Ours	0.7473	313.4896
initial condition-3		
PINNs [12]	0.1970	270.0388
Ours	1.6325	311.6468

Table 2: Summary of some initial conditions of u (observed data).

Synthetic observed data	$u\left(x,y,0\right)$
Initial Condition-1	$u_0=\frac{\tanh\left(0.2-\sqrt{(x-0.3)^2+(y-0.5)^2}\right)}{0.03\sqrt{2}}$
Initial Condition-2	$u_0=\left(\begin{array}{c}10\max\left(0.04-(x-0.2)^2-(y-0.65)^2,0\right)\\+12\max\left(0.03-(x-0.5)^2-(y-0.2)^2,0\right)\\+12\max\left(0.03-(x-0.8)^2-(y-0.55)^2,0\right)\\+\frac{\tanh\left(0.2-\sqrt{(x-0.3)^2+(y-0.3)^2}\right)}{0.03\sqrt{2}}\\+\frac{\tanh\left(0.2-\sqrt{(x-0.3)^2+(y-0.5)^2}\right)}{0.03\sqrt{2}}\end{array}\right)\\\times\frac{\tanh\left(0.2-\sqrt{(x-0.3)^2+(y-0.5)^2}\right)}{0.03\sqrt{2}}$
Initial Condition-3	$\begin{cases}u_0=0.5\tanh\frac{-r+\sqrt{(x-a_x)^2+\left(y-a_y\right)^2}}{2\sqrt{2}\varepsilon}\\+0.5\tanh\frac{-r+\sqrt{(x-b_x)^2+\left(y-b_y\right)^2}}{2\sqrt{2}\varepsilon},\varepsilon=0.01\\a_x=-\frac{r}{\sqrt{2}};a_y=\frac{r}{\sqrt{2}}b_x=\frac{r}{\sqrt{2}};b_y=-\frac{r}{\sqrt{2}}r=0.2\sqrt{2}\end{cases}$

Table 3: The comparison of the training loss values and times for high-order Quasi Xuguo flow with different initial conditions for DnCNN and FNN.

Initial condition	**Training loss values**	**Training times (s)**
initial condition-1		
DnCNN [38]	19.9318	354.0328
FNN	**0.1306**	**309.3411**
initial condition-2		
DnCNN [38]	57.6121	354.2204
FNN	**0.7473**	**313.4896**
initial condition-3		
DnCNN [38]	19.6188	353.8251
FNN	**1.6325**	**311.6468**



(a) initial-u-1 (PINNs) [12] (b) PINNs (t=31) [12] (c) PINNs (t=61) [12] (d) PINNs (t=100) [12]

(e) initial-u-1 (Ours) (f) Ours (t=31) (g) Our (t=61) (h) Ours (t=100)

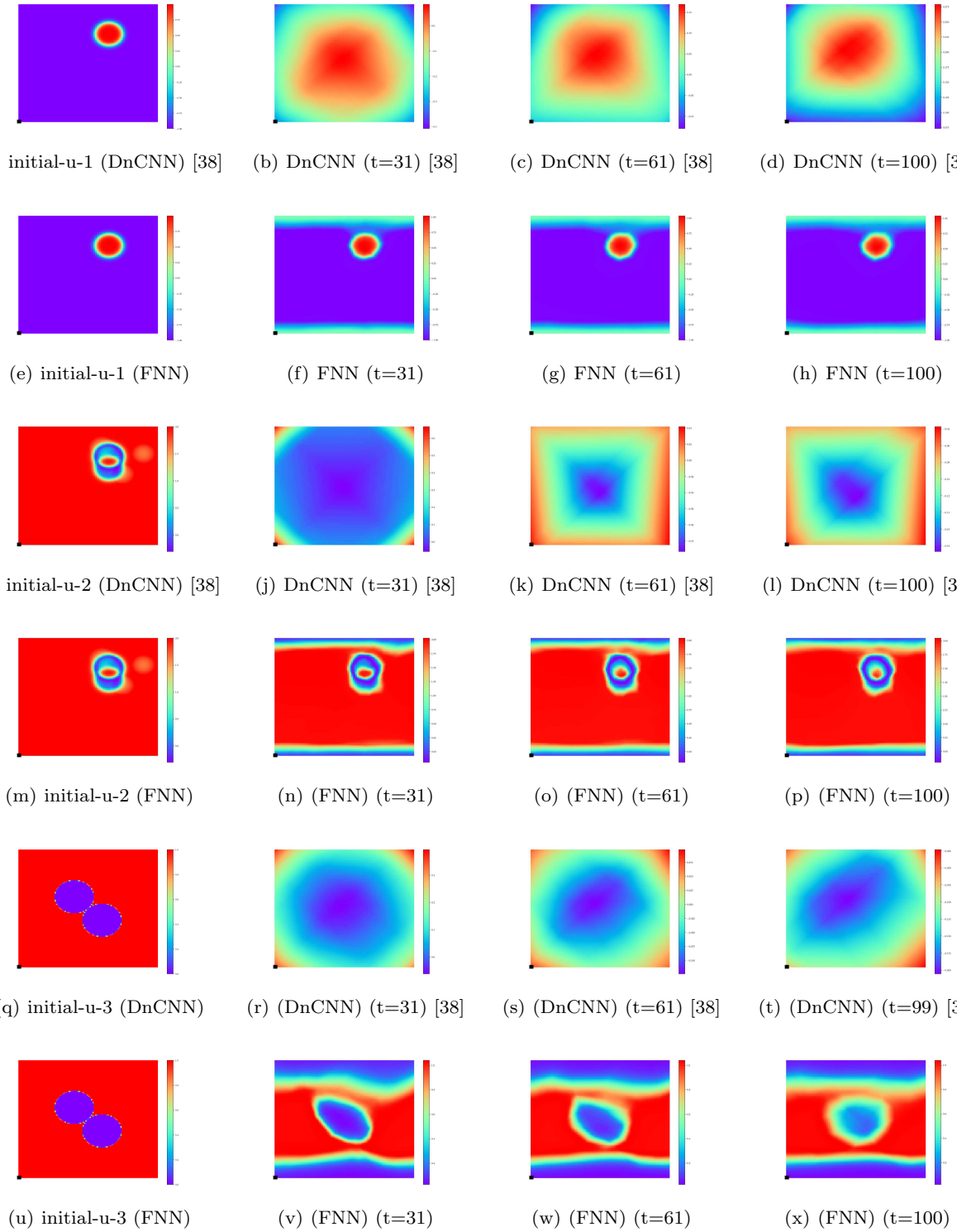
(i) initial-u-2 (j) PINNs (t=31) [12] (k) PINNs (t=61) [12] (l) PINNs (t=100) [12]

(m) initial-u-2 (n) Ours (t=31) (o) Ours (t=61) (p) Ours (t=100)

(q) initial-u-3 (r) PINNs (t=31) [12] (s) PINNs (t=61) [12] (t) PINNs (t=99) [12]

(u) initial-u-3 (Ours) (v) Ours (t=31) (w) Ours (t=61) (x) Ours (t=100)

Fig. 3: Examples of the visualization process of evolutionary for the new high-order surface diffusion Cahn–Hilliard flow with different initial conditions for PINNs and Ours.



(a) initial-u-1 (DnCNN) [38] (b) DnCNN (t=31) [38] (c) DnCNN (t=61) [38] (d) DnCNN (t=100) [38]

(e) initial-u-1 (FNN) (f) FNN (t=31) (g) FNN (t=61) (h) FNN (t=100)

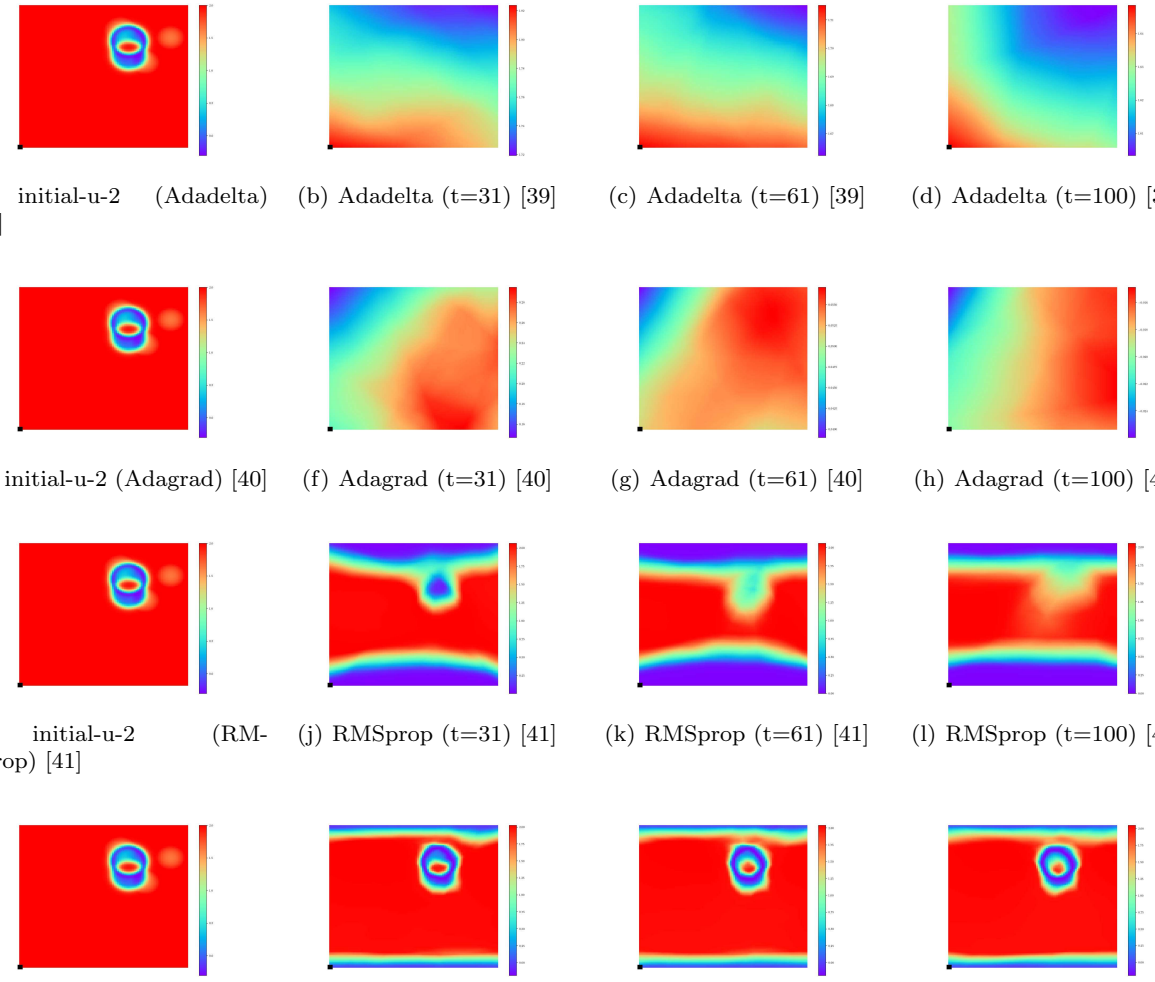
(i) initial-u-2 (DnCNN) [38] (j) DnCNN (t=31) [38] (k) DnCNN (t=61) [38] (l) DnCNN (t=100) [38]

(m) initial-u-2 (FNN) (n) (FNN) (t=31) (o) (FNN) (t=61) (p) (FNN) (t=100)

(q) initial-u-3 (DnCNN) (r) (DnCNN) (t=31) [38] (s) (DnCNN) (t=61) [38] (t) (DnCNN) (t=99) [38]

(u) initial-u-3 (FNN) (v) (FNN) (t=31) (w) (FNN) (t=61) (x) (FNN) (t=100)

Fig. 4: Examples of the visualization process of evolutionary for high-order Quasi Xuguo flow with initial conditions 1-3 for DnCNN and FNN.



(a) initial-u-2 (Adadelta) [39] (b) Adadelta (t=31) [39] (c) Adadelta (t=61) [39] (d) Adadelta (t=100) [39]

(e) initial-u-2 (Adagrad) [40] (f) Adagrad (t=31) [40] (g) Adagrad (t=61) [40] (h) Adagrad (t=100) [40]

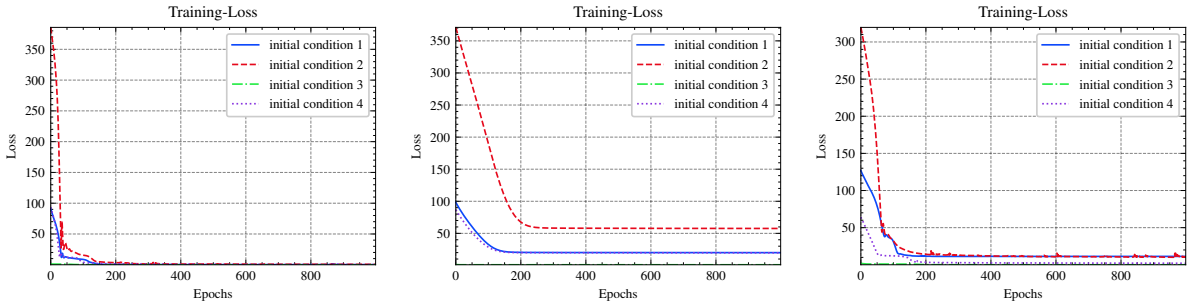
(i) initial-u-2 (RMSprop) [41] (j) RMSprop (t=31) [41] (k) RMSprop (t=61) [41] (l) RMSprop (t=100) [41]

(m) initial-u-2 (Adam) [42] (n) (Adam) (t=31) [42] (o) (Adam) (t=61) [42] (p) (Adam) (t=100) [42]

Fig. 5: Examples of the visualization process of evolutionary for high-order Quasi Xuguo flow with initial condition 2 and different optimizers.

Table 4: The comparison of the training loss values and times for high-order Quasi Xuguo flow with initial condition 2 for different optimizers.

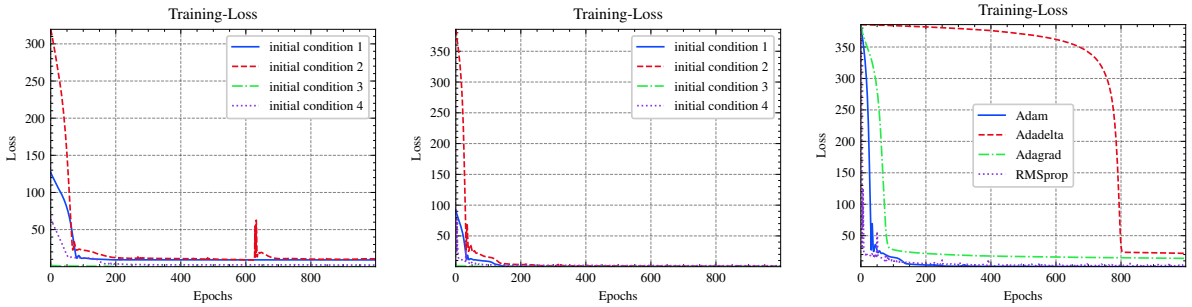
Initial condition initial condition-2	**Training loss values**	**Training times (s)**
Adadelta [39]	21.8484	**310.3310**
Adagrad [40]	13.9149	315.1199
RMSprop [41]	2.4678	316.8016
Adam [42]	**0.7473**	313.4896



(a) loss for high-order Quasi Xuguo flow with different initial conditions (PINNs)

(b) loss for high-order Quasi Xuguo flow with different initial conditions (CNN)

(c) loss for CH flow with different initial conditions (Ours)



(d) loss for CH flow with different initial conditions (PINNs)

(e) loss for CH flow with different initial conditions (Ours)

(f) loss for different optimizers

Fig. 6: The training loss for different situations.

$e_6 = q - \Delta_s \kappa$, and $e_7 = w - \Delta_s q$. Based on the analysis above, then the total loss function is obtained as follows,

$$Loss = Loss_{IC} + Loss_{BC} + \sum_{i=1}^{7} \|e_i\|_2^2, \tag{5}$$

where IC and BC refer to the initial and bound conditions, respectively. In this work, the initial conditions are discussed in more detail later, and the boundary condition is the periodic boundary one.

Usually, the systems of PDEs are viewed as the PDE-constrained optimization problem,

$$\min_{\Theta} L(\Theta) \; s.t. F(\Theta, u) = 0, \tag{6}$$

where Θ is the set of model parameters, and L is the loss function. When we replace Θ with NN_Θ, the following physics-constraint optimization learning one is obtained,

$$\min_{\Theta} L(\Theta) \; s.t. F(NN_\Theta, u) = 0. \tag{7}$$

The gradient descent approach can be used for solving it, and the parameter update equation is,

$$\Theta_{n+1} = \Theta_n - \alpha_n \nabla_\Theta L(\Theta_n) \tag{8}$$

where L refers to Eq. (5). The algorithm can be summarized as in **Algritghm 1**, and the schematic illustration of our LSM-physics constrained learning (PCL) framework for learning GPDE is shown in **Fig.1**.

3.2 High-order surface diffusion flow of Cahn–Hilliard model

Such as [7], we want to approximate the surface diffusion flow using LSM and high-order LBO of κ, to smoothing the system of PDEs, the Heaviside function is introduced as,

$$\begin{cases} H_\varepsilon(u) = 0.5 + \frac{1}{\pi}\arctan\left(\frac{u}{\varepsilon}\right), \\ \delta_\varepsilon(u) = H'_\varepsilon(u) = \frac{\varepsilon}{\pi(\varepsilon^2+u^2)}. \end{cases} \tag{9}$$

Then, the proposed new Cahn-Hilliard model reads as,

$$\begin{cases} \phi_t = N(\phi)\,\mathrm{div}_s\left(M(\phi)\nabla_s\left(N(\phi)\mu\right)\right)\|\nabla\phi\|_\varepsilon\delta_\varepsilon(\phi), \\ \mu = \frac{1}{\varepsilon}W'(\phi) - \Delta_s\kappa\|\nabla\phi\|_\varepsilon\delta_\varepsilon(\phi), \\ M(\phi) = W(\phi) + \gamma\varepsilon^2, \gamma > 0, \\ N(\phi) = \sqrt{M(\phi)}; W(\phi) = 0.5\phi^2(1-\phi)^2. \end{cases} \tag{10}$$

Since the **definition 1-2**, we can get a smooth vector field $\mathbf{v}$ defined on Ω, and the divergence operator div_S acting on $\mathbf{v}$, which has the following explicit

representation,

$$\begin{aligned}\operatorname{div}_S(\mathbf{v}) &= \operatorname{div}(\mathbf{v}) - \mathbf{n}^T(\nabla\mathbf{v})\,\mathbf{n} \\ &= \mathbf{v}_x + \mathbf{v}_y - \tfrac{\phi_x^2\mathbf{v}_x+\phi_x\phi_y(\mathbf{v}_x+\mathbf{v}_y)+\phi_y^2\mathbf{v}_y}{\phi_x^2+\phi_y^2+1}\,.\end{aligned} \tag{11}$$

Since Eq. (11) and **Definition 1**, Eq. (10) is explicitly represented, thereafter, which is solved by **Algorithm 1** with order reduction.

4 Experiments

In this section, we describe more details about the experimental setup and results while test the performance of our method. We train the deep neural network models on our equipment with a GeForce RTX 1080 super GPU. The software is developed based on the PyTorch framework (`www.pytorch.org/`). We present several numerical examples in two dimensions, including various phenomena to test the convergence of the proposed framework on the synthetic data. Our computational domain is the square $\Omega = [0,1] \times [0,1]$, and $t \in [0,5]$. We use periodic boundary conditions in all directions. And the computational parameters with a uniform space-grid $N_x = N_y = 100$ and a uniform time-grid $N_t = 100$ in all examples. Also, we test two NNs, namely, the feed-forward NN with 14 hidden layers (each layer contains 50 neurons), and the DnCNN [38], to test the robustness of our framework for various networks. We choose 50 boundary sampling points, 50000 inner sampling points, and 5000 initial sampling points for training. Examples of the visualization process of evolutionary for high-order flows with initial conditions for PINNs and Ours are shown in **Fig. 2-3**. And the training loss of PINNs and Ours for high-order flows with different initial conditions is shown in **Fig. 6**. Also, the comparison of the training loss values and times for high-order Quasi Xuguo flow with different initial conditions is shown in **Table 1**.

Examples of the visualization process of evolutionary for high-order Quasi Xuguo flow with initial conditions 1-3 for DnCNN and FNN is shown in **Fig. 4**. Also, some initial conditions of u is given in **Table 2**. And the comparison of the training loss values and times for high-order Quasi Xuguo flow with different initial conditions for DnCNN and FNN is shown in **Table 3**. Moreover, the examples of the visualization process of evolutionary for high-order Quasi Xuguo flow with initial condition 2 and different optimizers is found in **Fig. 5**. Furthermore, the comparison of the training loss values and times for high-order Quasi Xuguo flow with initial condition 2 for different optimizers can be shown in **Table 4**.

Conclusion

In this work, we explore the problem of high-order Quasi Xuguo flow and high-order surface diffusion Cahn–Hilliard flow with different initial conditions based on deep LSM. Also, we use different initial conditions that aim to study the sensitivity of the algorithm on different input data. Moreover, to verify the robustness of the algorithm, we used two networks for testing our algorithm. Theoretically, almost all networks fit our framework. Besides, through the combination of the traditional LSM and DNN, such that our framework becomes a powerful tool for solving high-order GPDEs. Furthermore, we study the influence of different optimizers on the learning results and convergence of the algorithm. And these optimization methods mainly include Adadelta, Adagrad, Adam, and RMSprop. The test results show that Adam is more suitable for our framework. In future work, we will use real-world data to test our framework and solve practical problems in computer vision such as denoising, inpainting, reconstruction, and segmentation problem.

Acknowledgements Thank you for your valuable comments from any anonymous reviewers.

Funding No funding provided.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

Open Access This article is permitted use, sharing, adaptation, distribution, and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source. The images or other third-party material in this article are included in the papers. If material is not included in the article and your intended use or exceeds the permitted use, you will need to obtain permission directly from my GitHub, please visit (`www.github.com/lichun0503/learning-Geom`

References

1. G. Bachmann, G. Bécigneul, and O.-E. Ganea, "Constant Curvature Graph Convolutional Networks," in *International Conference on Machine Learning*, 2020.
2. Z. Long, Y. Lu, X. Ma, and B. Dong, "PDE-Net: Learning PDEs from Data," in *International Conference on Machine Learning*, pp. 3208–3216, PMLR, 2018.
3. A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to Simulate Complex Physics with Graph Networks," in *International Conference on Machine Learning*, pp. 8459–8468, PMLR, 2020.

4. T. Xue, A. Beatson, S. Adriaenssens, and R. Adams, "Amortized Finite Element Analysis for Fast PDE-Constrained Optimization," in *International Conference on Machine Learning*, pp. 10638–10647, PMLR, 2020.
5. C. Li, Y. Yang, H. Liang, and B. Wu, "Robust PCL Discovery of Data-Driven Mean-Field Game Systems and Control Problems," *IEEE Transactions on Circuits and Systems I-regular Papers*, pp. 1–14, 2021.
6. C. Li, Y. Yang, H. Liang, and B. Wu, "Transfer Learning for Establishment of Recognition of COVID-19 on CT Imaging Using Small-Sized Training Datasets," *Knowledge-Based Systems*, vol. 218, pp. 106849 – 106849, 2021.
7. E. Bretin, S. Masnou, A. Sengers, and G. Terii, "Approximation of Surface Diffusion Flow: A Second Order Variational Cahn–Hilliard Model with Degenerate Mobilities," *arXiv preprint arXiv:2007.03793*, 2020.
8. W. Dang, Z. ke Gao, L. Hou, D. Lv, S. Qiu, and G. Chen, "A Novel Deep Learning Framework for Industrial Multiphase Flow Characterization," *IEEE Transactions on Industrial Informatics*, vol. 15, pp. 5954–5962, 2019.
9. Z. Gao, W. Dang, C. Mu, Y. Yang, S. Li, and C. Grebogi, "A Novel Multiplex Network-Based Sensor Information Fusion Model and Its Application to Industrial Multiphase Flow System," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 3982–3988, 2018.
10. S. Osher and J. A. Sethian, "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations," *Journal of Computational Physics*, vol. 79, no. 1, pp. 12–49, 1988.
11. M. Raissi and G. E. Karniadakis, "Hidden Physics Models: Machine Learning of Nonlinear Partial Differential Equations," *Journal of Computational Physics*, vol. 357, pp. 125–141, 2018.
12. M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.
13. X. Guoliang and Z. Qin, "Construction of Geometric Partial Differential Equations in Computational Geometry," *mathematica Numerica Sinica Chinese Edition*, vol. 28, no. 4, p. 337, 2006.
14. D. Chen, L. Jacob, and J. Mairal, "Convolutional Kernel Networks for Graph-Structured Data," in *International Conference on Machine Learning*, pp. 1576–1586, PMLR, 2020.
15. A. R. Chowdhury, T. Rekatsinas, and S. Jha, "Data-Dependent Differentially Private Parameter Learning for Directed Graphical Models," in *International Conference on Machine Learning*, pp. 1939–1951, PMLR, 2020.
16. G. Sundaramoorthi and A. Yezzi, "Variational PDEs for Acceleration on Manifolds and Application to Diffeomorphisms," in *The Conference on Neural Information Processing Systems*, 2018.
17. W. Zhu, Q. Qiu, J. Huang, R. Calderbank, G. Sapiro, and I. Daubechies, "LDMNet: Low Dimensional Manifold Regularized Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2743–2751, 2018.
18. L. Bai, Y. Shao, W. Chen, Z. Wang, and N. Deng, "Multiple Flat Projections for Cross-manifold Clustering," *IEEE transactions on cybernetics*, vol. PP, 2021.
19. "Multilabel Distribution Learning Based on Multioutput Regression and Manifold Learning., author=C. Tan and Sheng Chen and Genlin Ji and Xin Geng, journal=IEEE transactions on cybernetics," vol. PP, 2020.
20. J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, vol. 3. Cambridge University Press, 1999.
21. S. O. R. Fedkiw and S. Osher, "Level Set Methods and Dynamic Implicit Surfaces," *Surfaces*, vol. 44, p. 77, 2002.
22. Q. Lin, R. Ma, and T. Yang, "Level-Set methods for Finite-Sum Constrained Convex Optimization," in *International Conference on Machine Learning*, pp. 3112–3121, PMLR, 2018.
23. C. Li, R. Huang, Z. Ding, C. Gatenby, D. N. Metaxas, and J. Gore, "A Level Set Method for Image Segmentation in the Presence of Intensity Inhomogeneities With Application to MRI," *IEEE Transactions on Image Processing*, vol. 20, pp. 2007–2016, 2011.
24. T. Brox and J. Weickert, "Level Set Segmentation With Multiple Regions," *IEEE Transactions on Image Processing*, vol. 15, pp. 3213–3218, 2006.
25. C. Li, C. Xu, C. Gui, and M. Fox, "Distance Regularized Level Set Evolution and Its Application to Image Segmentation," *IEEE Transactions on Image Processing*, vol. 19, pp. 3243–3254, 2010.
26. K. Zhang, L. Zhang, K. Lam, and D. Zhang, "A Level Set Approach to Image Segmentation With Intensity Inhomogeneity," *IEEE Transactions on Cybernetics*, vol. 46, pp. 546–557, 2016.
27. S. Yan, X. Tai, J. Liu, and H. Huang, "Convexity Shape Prior for Level Set-Based Image Segmentation Method," *IEEE Transactions on Image Processing*, vol. 29, pp. 7141–7152, 2020.
28. J. A. Sirignano and K. Spiliopoulos, "DGM: A Deep Learning Algorithm for Solving Partial Differential Equations," *Journal of Computational Physics*, vol. 375, pp. 1339–1364, 2018.
29. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," in *The Conference on Neural Information Processing Systems*, 2018.
30. J. Han, A. Jentzen, and E. W., "Solving High-Dimensional Partial Differential Equations Using Deep Learning," *Proceedings of the National Academy of Sciences*, vol. 115, pp. 8505 – 8510, 2018.
31. L. Liang, L. Jin, and Y. Xu, "PDE Learning of Filtering and Propagation for Task-Aware Facial Intrinsic Image Analysis.," *IEEE transactions on cybernetics*, vol. PP, 2020.
32. L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Review*, vol. 63, no. 1, pp. 208–228, 2021.
33. S. Cai, Z. Wang, L. Lu, T. A. Zaki, and G. E. Karniadakis, "DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks," *Journal of Computational Physics*, p. 110296, 2021.
34. L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning Nonlinear Operators via DeepONet Based on the Universal Approximation Theorem of Operators," *Nature Machine Intelligence*, vol. 3, no. 3, pp. 218–229, 2021.
35. L. Zhao, Z. Li, Z. Wang, B. Caswell, J. Ouyang, and G. E. Karniadakis, "Active-And Transfer-Learning Applied to Microscale-Macroscale Coupling to Simulate Viscoelastic Flows," *Journal of Computational Physics*, vol. 427, p. 110069, 2021.
36. L. Yang, X. Meng, and G. E. Karniadakis, "B-PINNs: Bayesian physics-Informed Neural Networks for Forward

and Inverse PDE Problems with Noisy Data," *Journal of Computational Physics*, vol. 425, p. 109913, 2021.
37. G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*, vol. 147. Springer Science & Business Media, 2006.
38. K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.
39. M. D. Zeiler, "Adadelta: An Adaptive Learning Rate Method," *arXiv preprint arXiv:1212.5701*, 2012.
40. J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, no. 7, 2011.
41. A. Graves, "Generating Sequences with Recurrent Neural Networks," *arXiv preprint arXiv:1308.0850*, 2013.
42. D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.