RESEARCH ARTICLE | MAY 11 2022

Physics-informed neural networks for phase-field method in two-phase flow

Rundi Qiu (丘润荻); Renfang Huang (黄仁芳); Yao Xiao (肖姚); ... et. al

Check for updates

Physics of Fluids 34, 052109 (2022)
https://doi.org/10.1063/5.0091063

Physics-informed neural networks for phase-field method in two-phase flow

Cite as: Phys. Fluids **34**, 052109 (2022); doi: 10.1063/5.0091063
Submitted: 11 March 2022 · Accepted: 22 April 2022 ·
Published Online: 11 May 2022

Rundi Qiu (丘润荻),[1,2] Renfang Huang (黄仁芳),[1] Yao Xiao (肖姚),[3] Jingzhu Wang (王静竹),[1] Zhen Zhang (张珍),[4] Jieshun Yue (岳杰顺),[1] Zhong Zeng (曾忠),[3] and Yiwei Wang (王一伟)[1,2,5,a)]

AFFILIATIONS

[1]Key Laboratory for Mechanics in Fluid Solid Coupling Systems, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China
[2]School of Future Technology, University of Chinese Academy of Sciences, Beijing 100049, China
[3]Department of Engineering Mechanics, College of Aerospace Engineering, Chongqing University, Chongqing 400044, China
[4]School of Civil Engineering, Shijiazhuang Tiedao University, Shijiazhuang 050043, China
[5]School of Engineering Science, University of Chinese Academy of Sciences, Beijing 100049, China

Note: This paper is part of the special topic, Artificial Intelligence in Fluid Mechanics.
[a)]**Author to whom correspondence should be addressed:** wangyw@imech.ac.cn

ABSTRACT

The complex flow modeling based on machine learning is becoming a promising way to describe multiphase fluid systems. This work demonstrates how a physics-informed neural network promotes the combination of traditional governing equations and advanced interface evolution equations without intricate algorithms. We develop physics-informed neural networks for the phase-field method (PF-PINNs) in two-dimensional immiscible incompressible two-phase flow. The Cahn–Hillard equation and Navier–Stokes equations are encoded directly into the residuals of a fully connected neural network. Compared with the traditional interface-capturing method, the phase-field model has a firm physical basis because it is based on the Ginzburg–Landau theory and conserves mass and energy. It also performs well in two-phase flow at the large density ratio. However, the high-order differential nonlinear term of the Cahn–Hilliard equation poses a great challenge for obtaining numerical solutions. Thus, in this work, we adopt neural networks to tackle the challenge by solving high-order derivate terms and capture the interface adaptively. To enhance the accuracy and efficiency of PF-PINNs, we use the time-marching strategy and the forced constraint of the density and viscosity. The PF-PINNs are tested by two cases for presenting the interface-capturing ability of PINNs and evaluating the accuracy of PF-PINNs at the large density ratio (up to 1000). The shape of the interface in both cases coincides well with the reference results, and the dynamic behavior of the second case is precisely captured. We also quantify the variations in the center of mass and increasing velocity over time for validation purposes. The results show that PF-PINNs exploit the automatic differentiation without sacrificing the high accuracy of the phase-field method.

I. INTRODUCTION

In recent years, complex flow modeling with machine learning has achieved remarkable progress to efficiently reconstruct the high-fidelity dynamic model, including dimensionality reduction of fluid dynamic models,[1–6] clustering and classification problems in fluid mechanics,[7–10] closure models based on machine learning,[11–16] and neural network modeling of ordinary and partial differential equations.[17–22] A crucial consensus has been reached in fluids research: combining data-driven methods with known physical information can improve the interpretability, generalizability, and explainability of the training results.[23] Physics-informed neural networks (PINNs) proposed by Raissi *et al.*[18] are a typical framework to hybridize data with laws obtained from supervisors.[24] This framework is achieved by combining the residual form of the governing equations with a data-driven part, such as initial conditions and boundary conditions. Thus, the proposed neural network can satisfy the physics-constrained part and the data-driven part simultaneously. The framework connects network structures with determinant physical laws, so that fewer data are needed in the training process, which greatly enhances the modeling ability of neural networks.

The variation of physics-informed neural networks has been applied in various fields. Jin *et al.*[25] put forward NSFnets based on three-dimensional Navier–Stokes equations by considering the velocity–pressure formulation and the vorticity–velocity formulation. They comprehensively tested the NSFnets via several cases, and they considered the situation that is hard for traditional computational fluid dynamics (CFD) solvers, including inferring the unknown Reynolds numbers and predicting flow fields without the complete boundary conditions. Geneva and Zabaras[26] designed a physics-constrained deep auto-regressive network (AR-DenseED) to construct the temporal differential operator in a dynamic system, and several non-linear transient partial differential equations were discussed. Their results show that AR-DenseED can extrapolate outside the training range, and the framework can build surrogate models faster than the Fenics numerical solvers. Laubscher[27] proposed an improved multi-level PINNs for heat conduction in a two-dimensional rectangular domain filled with dry air. A single PINNs and a numerical result obtained from OpenFOAM were used for comparison. The author proved that an improved multi-level network works better than a single network. Xu *et al.*[28] constructed a deep neural network for identifying unknown artificial viscosity terms according to the dataset. This method can explore the missing flow dynamics with suitable information, even if the flow field reaches turbulence. Cai *et al.*[29] trained a fully connected neural network to infer the velocity and pressure distribution around an espresso cup, which indicated that PINNs have great potential for data assimilation. Mao *et al.*[30] used PINNs to solve the Euler equation in hypersonic flow. The oblique shock wave problem and Sod or Lax problem were presented to explore the characteristic of PINNs. Their result indicates that the position of collocation points is essential in discontinuous problems. Appropriate collocation points could essentially increase the convergence rate of PINNs. They also pointed out that special formulation may achieve better performance compared with the original formulation. Wang and Perdikaris[31] used PINNs with multiple inputs and outputs to solve an important type of free boundary problem called the Stephan problem. They explained how their work restores the solution near the dynamic interface and systematically moves the boundary. They also pointed out that PINNs have advantages for calculating the case with discontinuous parameters, due to the arbitrariness of choosing the sampling points. The accuracy of discontinuous parameters can be improved by adding sampling points near the discontinuous region, where there is too hard to generate high-quality meshes.

The aforementioned studies indicate that PINNs are applicable to two-phase flow with a distinct interface. Recently, Buhendwa *et al.*[32] used PINNs to investigate their application in detailed, incompressible two-phase flow. Forward and inverse problems were proposed to test the stability of PINNs. The framework of PINNs has sufficient accuracy if the dataset is sufficiently large, even if some noise exists in the dataset. They also adopted a refinement method near the interface of the bubble, which facilitates the capture of the interface. The existing results perform well with a density ratio of up to 10; however, the proposed hybrid algorithm may suffer from stability problem when the density ratio is large. To date, few reports are available on using PINNs to simulate two-phase flow at large density ratios.

An accurate interface-capturing method plays an important role in tracking the interface of two-phase flow. The interface-capturing method used by Buhendwa *et al.*[32] is the volume-of-fluid (VoF) method.[33] Other traditional interface-capturing methods include the level-set[34,35] and front-tracking methods.[36] Each method has its pros and cons. VoF conserves mass but the interface dissipates significantly, whereas level-set captures the interface accurately but does not conserve mass. The front-tracking method needs to track the position of the interface at all time, which takes significant time to update the information of the interface. In addition, the boundary conditions at the interfaces are generally imposed by applying a continuous surface force[37] to the conventional method. The divergence of the velocity field remains zero when both fluids are incompressible. However, this condition is not satisfied near the interface if the density ratio of two fluids is much greater than unity. Therefore, a special strategy should be followed to correctly reconstruct the interface in the aforementioned methods, and this can be very complicated, especially given a sufficiently large density ratio between two fluids.

The Cahn–Hillard model of the phase-field (CH-PF) is a promising approach to overcome the difficulties in two-phase flow at the large density ratio. In the CH-PF model, the thickness of the interface is assumed to be a small but finite value. The influence of interface thickness in the governing equations is mainly reflected in the diffusion term of the phase-field variable equation. A free-energy-density-based diffusion term is introduced to accurately describe the diffusion term.

The CH-PF model has significant advantages. It globally conserves mass, and it requires no complicated interface reconstructions. In the past two decades, the CH-PF model in two-phase flow has been systematically investigated. Ding *et al.*[38] used a modified CH-PF model to discuss its capacity to deal with large density ratios. Their results show that the CH-PF model accurately captures the evolution of the interface. Aland and Voigt[39] analyzed three various diffuse-interface approximations based on the CH-PF model by using the bubble-rising case with different density ratios. The parameters including interface thickness, mobility, and discretized time were discussed. The circularity, center of mass, and rising velocity of the bubble for each case were compared. Huang *et al.*[40] proposed three consistency conditions for the CH-PF model and implemented a balanced-force algorithm for the surface force. The proposed scheme guaranteed second-order accuracy in time and space. Zhang *et al.*[41] added an interface compression term to the Cahn–Hillard equations, to suppress diffusion at the interface. Xiao *et al.*[42] proposed a spectral element-based phase-field method for the Navier–Stokes/Cahn–Hillard equations for incompressible two-phase flows. Their method conserves mass and performs well at large density ratios and high Reynolds number. Despite many successful applications, improvements are still possible for dealing with two-phase flow using the CH-PF model. The governing equations of the CH-PF model generally contain high-order derivates, which cannot be discretized by a common method but require a complicated process.

Based on this literature, the traditional interface-capturing method cannot accurately deal with two-phase flow at the large density ratio. The CH-PF model is a potential method, but a complicated process restricts its industrial application. PINNs is a promising approach to build a flow modeling with high-order derivates efficiently and flexibly. With the help of automatic differentiation, high-order derivates are calculated accurately. The sampling position in PINNs is flexible, so we can adjust the position of sampling points arbitrary. A special sampling method can be used to increase the efficiency of PINNs. In addition, the application of the forward problem is beneficial to the implementation of the inverse problem for two-phase flow at the large density ratio, which provides a reliable method for data assimilation.

The motivation of this paper is to construct a robust and exact method for two-phase flow at the large density ratio. Toward this end, we incorporate the phase-field method with PINNs and make a preliminary investigation of the application of the proposed method. The performance of the method is validated by applying it to two cases called "reversed single vortex" and "bubble rising at large density ratio." The first case is treated to demonstrate the effectiveness of the CH-PF model as an interface-capturing method. To prevent the divergence of PF-PINNs, an adaptive time marching strategy is adopted, and the result is compared with those of the original PINNs to prove the accessibility of this strategy in the second case. We also quantitatively evaluate the difference between PF-PINNs and the reference results. The relative errors in horizontal velocity u exceed those of the vertical velocity v. In addition, these errors are sufficiently small, which shows the quantitative accuracy of PF-PINNs. PF-PINNs fit well with the quantitative parameters of the reference results, including the center of mass and the increasing velocity. PF-PINNs can, thus, exactly describe the characteristic of rising bubbles at the large density ratio.

The rest of this paper is organized as follows: we introduce the framework of physics-informed neural networks for the phase-field method (PF-PINNs) in two-phase flow at the large density ratio and the adaptive time-marching strategy in Sec. II. The training results for two cases are presented in Sec. III, where we confirm the capturing ability of PF-PINNs from the first case, and we quantify the dynamics of the rising bubble in the second case. The comparison between reference results and PF-PINNs in both cases is also conducted for validation purpose. In Sec. IV, the conclusions to this paper are summarized, and future works are proposed.

II. METHODOLOGY

A. Phase-field model for interface capturing

In the phase-field model, the Cahn–Hillard equation is usually used to capture the interface between two incompressible and immiscible fluids.[43–47] The subscripts "L" and "G" indicate the liquid phase and gas phase, respectively. The order parameter in incompressible two-phase flow is defined as the phase-field variable C. When C reaches unity, the physical properties of the mixture take the subscript L. On the contrary, the subscript changes to G if $C = -1$. The mixture is divided into two parts by the interface where $C = 0$. Based on the definition of the order parameter, C is restricted between -1 and 1, corresponding to two mixtures of fluids. The phase-field variable equation of C is[48]

$$\frac{\partial C}{\partial t} + (\mathbf{u}\cdot\nabla)C = \nabla\cdot(M_0\nabla\phi), \tag{1}$$

where M_0 is the mobility and ϕ is the chemical potential, which is derived from the free energy of mixture. According to Van der Waals,[49] the free-energy density of an immiscible isothermal two-phase fluid is

$$f_{\text{mix}} = \frac{1}{2}\varepsilon\sigma_0\alpha|\nabla C|^2 + \frac{\sigma_0\alpha}{\varepsilon}\psi(C), \tag{2}$$

$$\psi(C) = \frac{1}{4}C^2(1-C)^2, \tag{3}$$

where f_{mix} is the free energy per unit volume, σ_0 is the coefficient of surface tension, ε is the interface thickness, $\psi(C)$ is the bulk energy density, and α is a constant. The term $\frac{1}{2}\varepsilon\sigma_0\alpha|\nabla C|^2$ accounts for the excess free energy in the interfacial region.

In the phase-field method, the interface between two fluids can be interpreted as a transition region that has a thickness ε, so the surface tension is continuously distributed throughout the thickness of the interface. The surface tension is defined as the excess free energy per unit surface area.[50] For a one-dimensional stable interface at equilibrium, the following relation can be derived:[51]

$$\varepsilon\alpha\int_{-\infty}^{\infty}\left(\frac{\partial C}{\partial z}\right)^2 dz = 1. \tag{4}$$

The one-dimensional distribution of C near an interface at equilibrium is

$$C(z) = \tanh\left(\frac{z}{\sqrt{2}\varepsilon}\right), \tag{5}$$

where z is the signed normal distance to the interface. Applying Eq. (5) to Eq. (4), we can derive that α is $\frac{3\sqrt{2}}{4}$. The introduction of the one-dimensional interface is only for convenience of derivation. Universal physical parameters do not vary with the dimensionality of the case, so α remains $\frac{3\sqrt{2}}{4}$ in the following cases. The free energy F_{mix} in the spacial domain S is defined as the integral of the free energy density, leading to $F_{\text{mix}} = \int_S f_{\text{mix}} dS$. Thus, the chemical potential in Eq. (1) can be defined as the derivative of free energy with respect to parameter C,

$$\phi = \frac{\delta F_{\text{mix}}}{\delta C} = C(C^2-1) - \varepsilon^2\nabla^2 C. \tag{6}$$

B. Governing equations

The governing equations for incompressible, isothermal, and immiscible two-phase flow in two dimension include a continuity equation, momentum equations, and interface evolution equations,

$$\nabla\cdot\mathbf{u} = 0, \tag{7}$$

$$\rho_{\text{M}}\left(\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla p + \nabla\cdot\left[\mu_{\text{M}}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right] + \mathbf{f}_\sigma + \rho_{\text{M}}\mathbf{g}, \tag{8}$$

$$\begin{cases} \dfrac{\partial C}{\partial t} + (\mathbf{u}\cdot\nabla)C = \nabla\cdot(M_0\nabla\phi), \\ \phi = C(C^2-1) - \varepsilon^2\nabla^2 C, \end{cases} \tag{9}$$

$$\mathbf{f}_\sigma = \frac{3\sqrt{2}}{4}\frac{\sigma_0\phi}{\varepsilon}\nabla C, \tag{10}$$

where $\mathbf{u} = (u, v)$ is the velocity vector for two-dimensional flow, p is the pressure, $\mathbf{g} = (g_x, g_y)$ is the gravitation acceleration in the horizontal and vertical directions, respectively, ρ_{M} and μ_{M} are the mixture density and mixture dynamic viscosity, respectively, and $\mathbf{f}_\sigma = (f_{\sigma x}, f_{\sigma y})$ is the surface tension along the interface, which can be projected onto the horizontal and vertical directions. The physical properties of the mixture are related to C,

$$\rho_{\text{M}} = \frac{1+C}{2}\rho_{\text{L}} + \frac{1-C}{2}\rho_{\text{G}}, \tag{11}$$

$$\mu_{\text{M}} = \frac{1+C}{2}\mu_{\text{L}} + \frac{1-C}{2}\mu_{\text{G}}. \tag{12}$$

C. Physics-informed neural networks

After the governing equations are obtained, the physics-informed neural networks can be constructed by adding governing equations, initial conditions, and boundary conditions to the loss terms, so that optimization can be applied to solve the equations. The illustration of a physics-informed neural network for the phase-field method is shown in Fig. 1. A fully connected neural network is constructed to express the relationship between coordinates and physical quantities, which can be expressed as follows:

$$(u, v, p, C) = F_{NN}(x, y, t; \Theta), \tag{13}$$

where $X = (x, y, t)$, including time and two-dimensional space, is considered as the input value of the neural network. Velocities, pressure, and phase-field variable [i.e. (u, v, p, C)] are predicted by the neural network F_{NN}. Θ represents the trainable parameters, such as weight W and bias B. The relationship between the n-th hidden layer and the (n − 1)-th hidden layer is written as

$$X^n = \sigma(W^{n-1}X^{n-1} + B^{n-1}). \tag{14}$$

Here, the nonlinear activation function σ in Eq. (14) and Fig. 1 is the swish function $\sigma(\mathrm{Y}) = \mathrm{Y} * \mathrm{sigmoid}(Y)$. The physics-informed part of neural networks is derived from the residuals of the Cahn–Hillard equations and the two-dimensional Navier–Stokes equations corresponding to Eqs. (7)–(12),

$$L_{\mathrm{M}} = \partial_x u + \partial_y v, \tag{15}$$

$$L_C = \partial_t C + u\partial_x C + v\partial_y C - M_0(\partial_{xx}\phi + \partial_{yy}\phi), \tag{16}$$

$$\begin{aligned} L_U = (&\rho_{\mathrm{M}}(\partial_t u + u\partial_x u + v\partial_y u) + \partial_x p \\ &- 0.5(\mu_{\mathrm{L}} - \mu_{\mathrm{G}})\partial_y C(\partial_y u + \partial_x v) \\ &- (\mu_{\mathrm{L}} - \mu_{\mathrm{G}})\partial_x C\partial_x u - \mu_{\mathrm{M}}(\partial_{xx}u + \partial_{yy}u) \\ &- f_{\sigma x} - \rho_{\mathrm{M}}g_x)/\rho_{\mathrm{L}}, \end{aligned} \tag{17}$$

$$\begin{aligned} L_V = (&\rho_{\mathrm{M}}(\partial_t v + u\partial_x v + v\partial_y v) + \partial_y p \\ &- 0.5(\mu_{\mathrm{L}} - \mu_{\mathrm{G}})\partial_x C(\partial_x v + \partial_y u) \\ &- (\mu_{\mathrm{L}} - \mu_{\mathrm{G}})\partial_y C\partial_y v - \mu_{\mathrm{M}}(\partial_{xx}v + \partial_{yy}v) \\ &- f_{\sigma y} - \rho_{\mathrm{M}}g_y)/\rho_{\mathrm{L}}, \end{aligned} \tag{18}$$

where ∂_a represents $\frac{\partial}{\partial a}$, ∂_{ab} denotes $\frac{\partial^2}{\partial a \partial b}$, (L_M, L_C, L_U, L_V) correspond to the residual terms of the continuity equation, phase-field variable equation, and momentum equations in the x and y directions, respectively. In physics-informed part of neural networks, the derivate of physical quantities (i.e., $\partial_x u$, $\partial_y v$, and so on) in each term (L_M, L_C, L_U, L_C) is derived from automatic differentiation,[52] and the residuals of the governing equations are imposed. The total loss is composed of all residual terms of the governing equations and the ground truth of the initial and boundary conditions. The neural network can be trained by minimizing the mean squared total loss L_{total} in discretized sampling points,

$$L_{total} = L_{Eqn} + L_{ICs} + L_{BCs}, \tag{19}$$

$$\begin{aligned} L_{Eqn} = &\frac{1}{N_{Eqn}} \sum_{j=1}^{N_{Eqn}} (|L_C(x^j_{Eqn}, y^j_{Eqn}, t^j_{Eqn})|^2 + |L_U(x^j_{Eqn}, y^j_{Eqn}, t^j_{Eqn})|^2 \\ &+ |L_V(x^j_{Eqn}, y^j_{Eqn}, t^j_{Eqn})|^2 + |L_M(x^j_{Eqn}, y^j_{Eqn}, t^j_{Eqn})|^2), \end{aligned} \tag{20}$$

$$\begin{aligned} L_{ICs} = &\frac{1}{N_{ICs}} \sum_{j=1}^{N_{ICs}} (|C^j_{ICs} - C(x^j_{ICs}, y^j_{ICs}, t^j_{ICs})|^2 \\ &+ |u^j_{ICs} - u(x^j_{ICs}, y^j_{ICs}, t^j_{ICs})|^2 + |v^j_{ICs} - v(x^j_{ICs}, y^j_{ICs}, t^j_{ICs})|^2), \end{aligned} \tag{21}$$

$$\begin{aligned} L_{BCs} = &\frac{1}{N_{BCs}} \sum_{j=1}^{N_{BCs}} (|C^j_{BCs} - C(x^j_{BCs}, y^j_{BCs}, t^j_{BCs})|^2 \\ &+ |u^j_{BCs} - u(x^j_{BCs}, y^j_{BCs}, t^j_{BCs})|^2 + |v^j_{BCs} - v(x^j_{BCs}, y^j_{BCs}, t^j_{BCs})|^2), \end{aligned} \tag{22}$$

where $(L_{Eqn}, L_{ICs}, L_{BCs})$ are the discretized forms of physics-informed part of neural networks, and $(x^j_{Eqn}, y^j_{Eqn}, t^j_{Eqn})$, $(x^j_{ICs}, y^j_{ICs}, t^j_{ICs})$, and

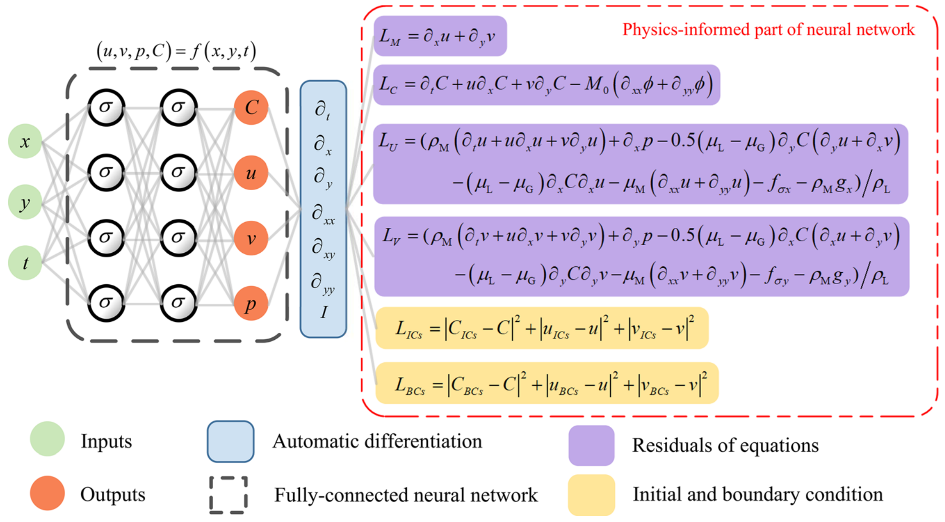
$(u,v,p,C)=f(x,y,t)$
x
y
t
σ
C
u
v
p
∂_t ∂_x ∂_y ∂_{xx} ∂_{xy} ∂_{yy} I
Physics-informed part of neural network
$L_M=\partial_x u+\partial_y v$
$L_C=\partial_t C+u\partial_x C+v\partial_y C-M_0(\partial_{xx}\phi+\partial_{yy}\phi)$
$L_U=(\rho_M(\partial_t u+u\partial_x u+v\partial_y u)+\partial_x p-0.5(\mu_L-\mu_G)\partial_y C(\partial_y u+\partial_x v)-(\mu_L-\mu_G)\partial_x C\partial_x u-\mu_M(\partial_{xx}u+\partial_{yy}u)-f_{\sigma x}-\rho_M g_x)/\rho_L$
$L_V=(\rho_M(\partial_t v+u\partial_x v+v\partial_y v)+\partial_y p-0.5(\mu_L-\mu_G)\partial_x C(\partial_x u+\partial_y v)-(\mu_L-\mu_G)\partial_y C\partial_y v-\mu_M(\partial_{xx}v+\partial_{yy}v)-f_{\sigma y}-\rho_M g_y)/\rho_L$
$L_{ICs}=|C_{ICs}-C|^2+|u_{ICs}-u|^2+|v_{ICs}-v|^2$
$L_{BCs}=|C_{BCs}-C|^2+|u_{BCs}-u|^2+|v_{BCs}-v|^2$
Inputs
Outputs
Automatic differentiation
Fully-connected neural network
Residuals of equations
Initial and boundary condition

FIG. 1. Illustration of physics-informed neural networks for the phase-field method (PF-PINNs). A fully connected neural network is constructed. The inputs of the network are variable (*x*, *y*, *t*), and (u, v, p, C) are regarded as the output variables of the neural network. After the derivate of each term is obtained by automatic differentiation (AD), the physics-informed part of neural network is used to express the incompressible Navier–Stokes equations and the Cahn–Hillard equation.

$(x^j_{BCs}, y^j_{BCs}, t^j_{BCs})$ denote the sampling point j generated from the internal field, initial field, and boundary field, respectively. The number of sampling points for various types of residual are N_{Eqn}, N_{ICs}, and N_{BCs}, respectively. The subscript "Eqn" indicates that these sampling points are chosen at random from the computational domain $\Omega \in (x, y, t)$. The subscripts "ICs" and "BCs" indicate the given data at the corresponding point $(x^j_{ICs}, y^j_{ICs}, t^j_{ICs})$ or $(x^j_{BCs}, y^j_{BCs}, t^j_{BCs})$. In PF-PINNs, the initial and boundary conditions are considered as supervised learning, while the residual terms of equations are considered as unsupervised learning. The proposed network is dimensional, which means that the inputs and outputs of the network have dimensions. It is appropriate for the proposed loss function to obtain reasonable results. In this work, all of the physical parameters are of the same order of magnitude, so the mixture total loss still converges. If this algorithm needs to be expanded, proper nondimensionalization is needed according to the characteristic of the physical background.

The optimizer updates the weights and biases in PF-PINNs by adjusting them to make sure the total loss of network L_{total} is decreasing. A method named *Adam*[53] for stochastic optimization is applied to optimize the aforementioned total loss L_{total}. To ensure the robustness of the training result, the Xavier initialization method[54] is used to randomly set the weight and bias. Once the total loss decreases to a minimum or the maximum epoch is achieved, the training progress of the neural network ends, and output values in the given domain are calculated. The PF-PINNs model is implemented in PyTorch,[55] and the training is conducted on two Tesla V100 Graphics Processing Unit (GPU) cards.

D. Adaptive time marching strategy

When the optimizer trains the neural network, it forces the total loss to decrease by tuning up the weights and biases. Thus, the size of the temporal domain for each network affects the convergence of the training process significantly. In general, training a network in the whole temporal domain is hard to converge. Dividing the sampling domain into several parts can improve the accuracy of the proposed network and accelerate the training process. Although there exist multiple networks, the size of each network is sufficiently small to ensure the efficiency of the entire network without losing accuracy. To obtain accurate training results, an adaptive time marching strategy[56] is introduced. The computational domain $\Omega \in (x, y, t)$ is split up into N smaller sub domains $(\Omega^1, \Omega^2, \ldots, \Omega^N)$ over time, and then we use N sub networks to solve the problem separately. The schematic of original single network and improved multiple networks is illustrated in Figs. 2(a) and 2(b), respectively.

In this strategy, the initial condition for each subdomain over time is obtained from the converged value of the former network. The initial condition for the first subdomain can be given directly, and then the first network is well trained. The converged value of the first network is then used as the initial condition for the second subdomain. The procedure continues until we obtain the converged value of the final network. The strategy trains the networks one by one. This strategy cannot save training time directly but promotes the convergence of each network. A detailed description of PF-PINNs with this strategy is shown in Algorithm 1.

III. NUMERICAL RESULTS AND DISCUSSION

In this section, PF-PINNs introduced in Sec. II are applied to solve two cases: a reversed single vortex and a bubble-rising problem. The first case is used to validate the ability of the phase-field method to capture the interface, and the second case is a classical case to test the numerical method with a large density ratio.

A. Reversed single vortex

Here, a reversed single vortex is used as a test case to demonstrate the performance of PF-PINNs in capturing the interface. In this case, the momentum equations and continuity equation are not considered, so only the interface evolution equation Eq. (9) is solved. The pressure field, density, and viscosity are no longer needed, whereas the velocity fields are close to Eq. (9). The analytical solutions of the velocity field in the x and y directions are

$$u = -U_0 \sin^2\left(\frac{\pi x}{L_0}\right) \sin\left(\frac{\pi y}{L_0}\right) \cos\left(\frac{\pi y}{L_0}\right) \cos\left(\frac{\pi t}{T_0}\right), \quad (23)$$

$$v = U_0 \sin\left(\frac{\pi x}{L_0}\right) \cos\left(\frac{\pi x}{L_0}\right) \sin^2\left(\frac{\pi y}{L_0}\right) \cos\left(\frac{\pi t}{T_0}\right), \quad (24)$$

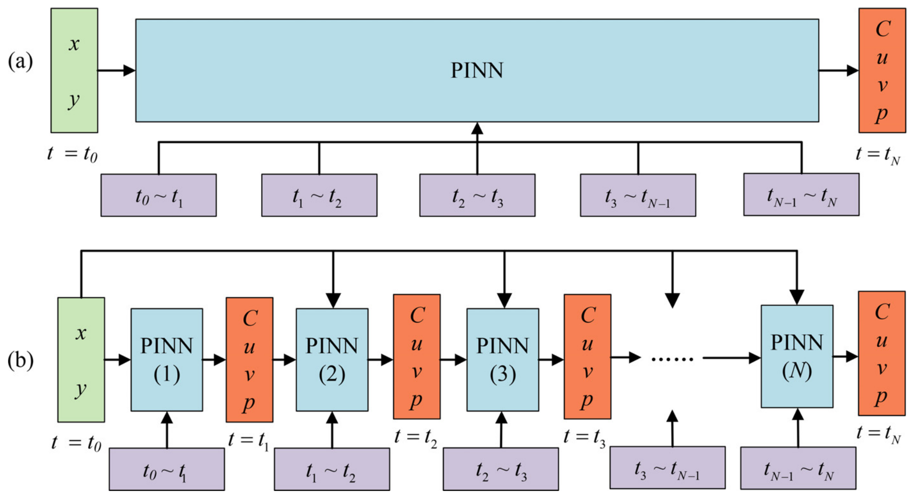

FIG. 2. Different time marching strategies of physics-informed neural networks for the phase-field method (PF-PINNs). (a) Single network training in the whole time domain. The output of the single network is the variables (u, v, p, C) in $\Omega \in (x, y, t_0 \sim t_N)$; (b) multiple-networks training in various time sequences. The first network is trained to obtain the variables (u, v, p, C) in $\Omega^1 \in (x, y, t_0 \sim t_1)$. The output variables (u, v, p, C) at $t = t_1$ and $\Omega^2 \in (x, y, t_1 \sim t_2)$ are considered as the input for the next network. The procedure continues until the final network sampled in $\Omega^N \in (x, y, t_{N-1} \sim t_N)$ is trained.

ALGORITHM 1: Physics-informed neural networks for the phase-field method (PF-PINNs).

```
Input: Computational domain $\Omega \in (x, y, t)$, number of time interval $N$, time sequence $(t_0, t_1, t_2, \ldots, t_N)$, initial variables $(u(t=t_0), v(t=t_0), p(t=t_0), C(t=t_0))$, boundary variables $(u_{BCs}, v_{BCs}, p_{BCs}, C_{BCs})$, numbers of different sampling points $N_{Eqn}, N_{ICs}, N_{BCs}$, learning rate $l_r$, training epoch $K$.
Output: Variables $(u, v, p, C)$ in $t \in [t_0, t_N]$, network parameter $\theta_i$.
for $i = 1$ to $N$ do
    Extract the sub domain $\Omega^i \in (x, y, t_{i-1} \sim t_i)$ from computational domain $\Omega$;
    Set $u(t=t_{i-1}), v(t=t_{i-1}), p(t=t_{i-1}), C(t=t_{i-1})$ obtained from previous iteration or from input as initial condition;
    Initialize $\theta_i$, $l_r$;
    for $epoch = 1$ to $K$ do
        Generate sampling points $(x^j_{Eqn}, y^j_{Eqn}, t^j_{Eqn})$, $(x^j_{ICs}, y^j_{ICs}, t^j_{ICs})$, and $(x^j_{BCs}, y^j_{BCs}, t^j_{BCs})$ in sub domain $\Omega^i$ according to the number of different sampling points $N_{Eqn}, N_{ICs}, N_{BCs}$;
        Calculate variables $(u, v, p, C)$ in $\Omega^i \in (x, y, t_{i-1} \sim t_i)$ by neural network using the relationship $(u, v, p, C) = f(x, y, t)$;
        Use automatic differentiation to infer the derivate of variables $(u, v, p, C)$;
        Construct the residuals of governing equations $L_{Eqn}$, initial conditions $L_{ICs}$, and boundary conditions $L_{BCs}$ in terms of Eqs. (13)–(20);
        Train the network through improved stochastic gradient descent algorithm Adam;
        Update network parameter $\theta_i$.
    end
    Output network parameter $\theta_i$ and variables $u(t=t_i), v(t=t_i), p(t=t_i), C(t=t_i)$ as the initial condition of the next network.
end
```

where the characteristic length L_0 is 1 m, the characteristic velocity U_0 is 2 m/s, and the characteristic time T_0 is 2 s. The initial configuration and boundary conditions are depicted in Fig. 3. The size of spacial computational domain $S \in (x, y)$ is $[0, 1]$ m $\times$ $[0, 1]$ m, and the time interval is $t \rightarrow [0, 2]$ s. The domain is filled with fluid L, where $C = 1$, except for a bubble located at $(x_c, y_c) = (0.5, 0.75\,\text{m})$; the bubble radius is $r = 0.15$ m. The bubble is occupied by fluid G, where $C = -1$. The phase-field variable in the whole spatial domain S at $t = 0$ s is

$$C(x, y, 0) = -\tanh\left(\frac{\sqrt{(x - x_c)^2 + (y - y_c)^2}}{\sqrt{2}\varepsilon}\right), \tag{25}$$

where $\varepsilon = 0.01$ m constitutes the interface thickness. Initially, the bubble shape is circular. The bubble is stretched by the given flow field, and the approximate trajectory of the bubble is shown in Fig. 3. The bubble moves toward the yellow dashed line clockwise from 0 to 1 s, and the maximum deformation is achieved when $t = 1$ s. Afterward, the bubble returns to its initial shape counterclockwise along the original trajectory from 1 to 2 s. In this case, the network regards the

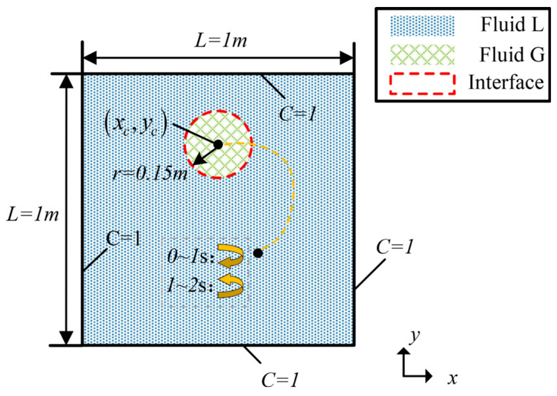

FIG. 3. Initial configuration and boundary conditions of reversed single vortex. Blue shadow region: fluid L. Green cross line region: fluid G. Red dashed line: interface at $t = 0$ s. The size of the spatial computational domain is $S \in (x, y) \rightarrow [0, 1]$ m $\times [0, 1]$ m. The bubble center is at $(x_c, y_c) = (0.5, 0.75\,\text{m})$, and the bubble radius is $r = 0.15$ m. All boundaries require phase-field variable $C = 1$. The bubble moves toward the yellow dashed line clockwise from 0 to 1 s and then returns to the initial shape counterclockwise along the original trajectory from 1 to 2 s.

sampling space $\Omega \in (x, y, t)$ as input, and the only output is the phase-field variable C. We consider a neural network consisting of 10 hidden layers with 100 neuron per layer, and the *Adam* optimizer is used to train and 7×10^4 epochs with a learning rate ranging from 10^{-3} to 10^{-6} are used to train the network. The learning rate decreases as the epoch increases, as shown in Table I. All sampling points are generated in advance on a uniformly spaced structured grid. The cell size in the x, y, and t directions is $\Delta x = 0.005$, $\Delta y = 0.005$ m, and $\Delta t = 0.01$ s, respectively. We randomly select 1000 points at $t = 0$ s and 200 points from every fixed boundary condition (i.e., $N_{ICs} = 1000$ and $N_{BCs} = 200 \times 4$). Fifty thousand points are also randomly extracted inside the computational domain $\Omega \in (x, y, t) \rightarrow [0, 1]$ m $\times [0, 1]$ m $\times$ $[0, 2]$ s.

The predicted results for the phase-field variable at $t = 0$, $t = 0.5$, $t = 1.0$, $t = 1.5$, and $t = 2$ s are shown in Fig. 4, and the loss history is shown in Fig. 5. The contour indicates that the phase-field variable may be slightly out of range within $[-1, 1]$. Nevertheless, the evolution of the interface over time is still well captured by the neural network. The total bubble volume remains unchanged at various times. Figure 6 compares the results of PF-PINNs with those by Huang *et al.*[40] when $t = 1$ and $t = 2$ s. Our result proves that a physics-informed neural network suffices for tracking the shape of the interface regardless of the momentum equations.

B. Bubble rising

The classical numerical benchmark proposed by Hysing *et al.*[57] is introduced to evaluate the performance of PINNs in complex flow. The benchmark considers an isothermal, incompressible flow of

TABLE I. Learning rate settings in reversed single vortex.

Epoch	25 000	25 000	10 000	10 000
Learning rate	10^{-3}	10^{-4}	10^{-5}	10^{-6}

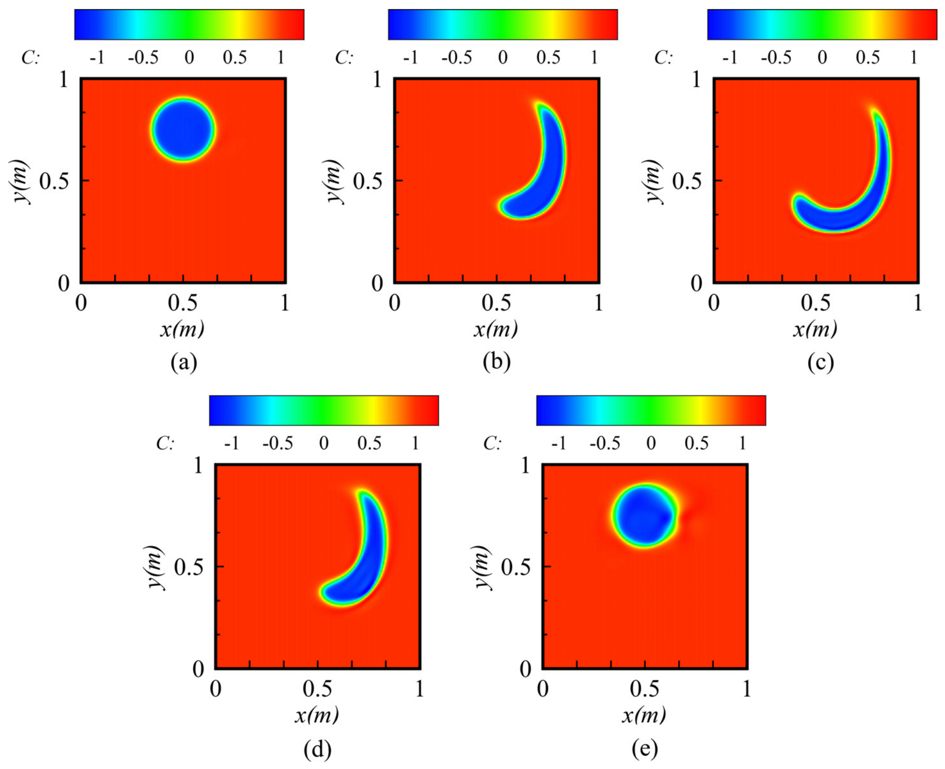

FIG. 4. Predicted results of phase-field variable C at (a) $t=0$, (b) $t=0.5$, (c) $t=1.0$, (d) $t=1.5$, and (e) $t=2.0$ s. The circle bubble is stretched into a crescent shape from 0 to 1 s, and then the crescent bubble returns along the original path and recovers the initial shape from 1 to 2 s. A slightly out-of-bound phase-field variable is observed in each time, but the bubble volume in the whole process remains unchanged.

immiscible fluids, and all of the equations and parameters are involved during the training. The initial configuration and boundary conditions are depicted in Fig. 7. To demonstrate the reliability of the proposed method in interface capturing, the validation of mass conservation is applied in Appendix A. The spatial computational domain is $S \in (x, y) \rightarrow [-0.5, 0.5]\,\text{m} \times [-1, 1]\,\text{m}$, and the time interval is $t \rightarrow [0, 3]$ s. The center position of the bubble in initial time is at $(x_c, y_c) = (0\,\text{m}, -0.5\,\text{m})$, and the bubble radius is $r = 0.25$ m. Similar to Sec. III A, $C=1$ represents the fluid L and $C = -1$ represents the fluid G. Curve $C=0$ defines the position of the interface between the two fluids. The upper and lower boundaries are considered as wall boundaries, and the left and right boundaries are considered as slip boundaries. The initial value of the phase-field variable at $t = 0$ s is imposed by Eq. (25), and the initial velocity field is zero. In the traditional phase-field method, the interface thickness and mobility are two representative parameters. To evaluate the proper value of these

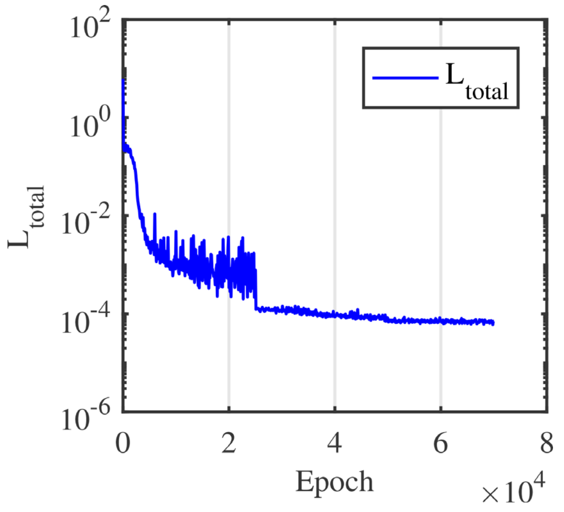

FIG. 5. Total training loss of the reversed single vortex case. L_{total} is given by Eq. (19). The total loss suddenly becomes smooth due to a decrease in the learning rate from 10^{-3} to 10^{-4} when the epoch reaches 25 000.

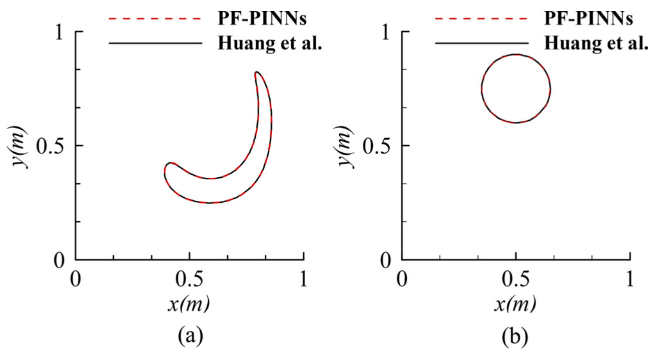

FIG. 6. Interfaces $C=0$ between fluid L and fluid G at (a) $t = 1$ s and (b) $t = 2$ s. Red dashed lines correspond to the present results, and black solid lines correspond to the result reported by Huang *et al.*[40]

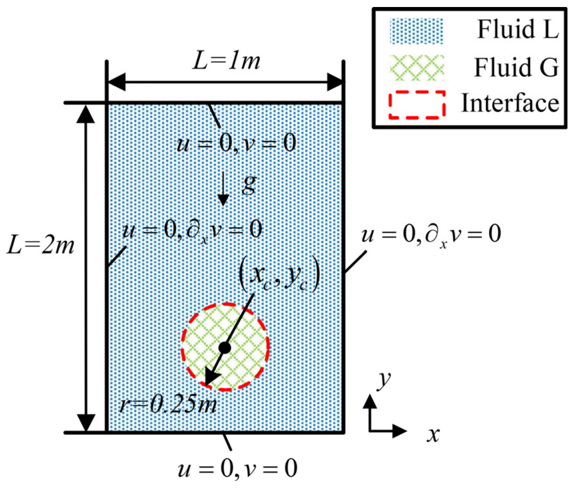

FIG. 7. Initial conditions and boundary conditions for rising-bubble case. The spatial computational domain is $S \in (x, y) \rightarrow [-0.5, 0.5]$ m $\times [-1, 1]$ m, and the time interval is $t \rightarrow [0, 3]$ s. The center position of the bubble at the initial time is $(x_c, y_c) = (0\,\text{m}, -0.5\,\text{m})$, and the bubble radius is $r = 0.25$ m. The initial value of the phase-field variable at $t = 0$ s is obtained by Eq. (25), and the initial velocity in both directions is zero. The upper and lower boundaries are considered as wall boundaries, and the left and right boundaries are considered as slip boundaries.

parameters, we systematically study the influence of thickness and mobility in Appendix B. The interface thickness and mobility are $\varepsilon = 0.01$ m and $M_0 = 10^{-4}\,\text{m}/(\text{N}\cdot\text{s})$, respectively. In this case, the momentum equations are no longer omitted, and the transport properties of fluids L and G should be considered. To illustrate the performance of PF-PINNs in two-phase flow, a large density ratio is applied. We assume $\rho_L = 1000$, $\rho_G = 1\,\text{kg/m}^3$, $\mu_L = 10$, and $\mu_G = 0.1\,\text{N s/m}^2$. The gravitational acceleration $\mathbf{g} = (g_x, g_y)$ is $(0, 0.98)\,\text{m/s}^2$ in the x and y directions. The coefficient σ_0 of surface tension is 1.96 N/m. The Reynolds number, which is the ratio of viscous effects to inertia, and the Eotvos number, which is the ratio of gravitation to surface-tension effects, are defined as

$$Re = \frac{\rho_L U_g L}{\mu_L}, \tag{26}$$

$$Eo = \frac{\rho_L U_g^2 L}{\sigma_0}, \tag{27}$$

where $L = 2r_0$ is the characteristic length, $U_g = \sqrt{2gr_0}$ is the characteristic velocity, and r_0 is the initial bubble radius, according to Hysing *et al.*[57] The physical parameters and dimensionless numbers defined in this case are given in Table II. The circular bubble inside an initially motionless rectangular domain is filled with fluid G, and the remaining region contains fluid L. In this case, fluid L is denser than fluid G. The bubble first stretches horizontally, then deforms at the bottom of the bubble, and finally preserves the stable shape and rising velocity.

TABLE III. Learning rate settings in bubble rising for a sub network.

Epoch	15 000	35 000	15 000	10 000	5000
Learning rate	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}

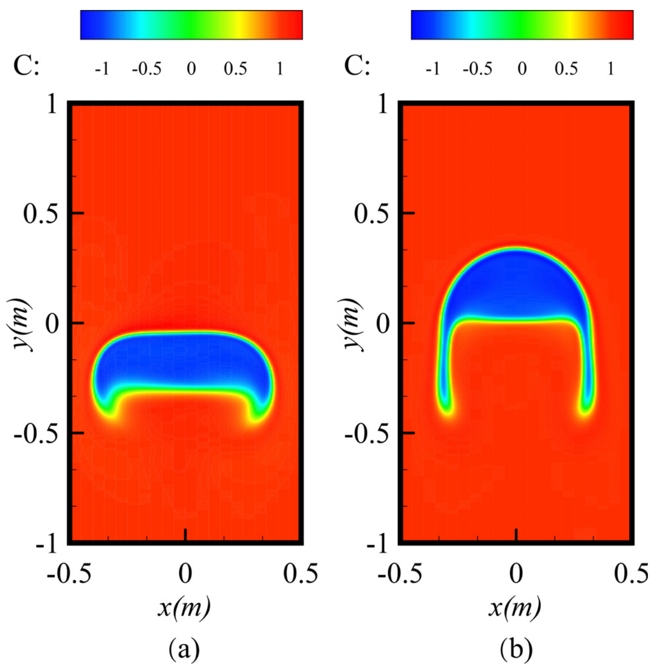

FIG. 8. Training results for $t = 3$ s using various strategies: (a) single network without improvement and (b) multiple networks with time marching strategy. The result generated from the normal training strategy does not converge, whereas the result generated by the multiple networks with the time-marching strategy provides adequate accuracy.

We consider a neural network consisting of 10 layers and 100 neurons per layer, and an identical optimizer is applied to train the networks. This network constructs a sampling space (x, y, and t) as input, and the velocity field, pressure, and phase-field variable ($u, v, p,$ and C) are considered as output. The initial learning rate for *Adam* decays from 10^{-3} to 10^{-6}, and the total number of epochs reaches 8×10^4. Different epochs are used to make sure the network is converged. The hyperparameters are depicted in Table III. All sampling points are generated in advance on a structured uniform spacing grid. The cell size in the x, y, and t directions is $\Delta x = 0.005$, $\Delta y = 0.005$ m, and $\Delta t = 0.005$ s, respectively. For the training data, we place 2000 points in initial conditions and 300 points in each boundary condition (i.e., $N_i = 2000$ and $N_b = 300 \times 4$). Twenty thousand points are extracted from the sampling space $\Omega \in (x, y, t) \rightarrow [0, 1]\,\text{m} \times [0, 2]\,\text{m} \times [0, 3]$ s.

TABLE II. Physical parameters and dimensionless numbers.

ρ_L (kg/m^3)	ρ_G (kg/m^3)	μ_L (N s/m^2)	μ_G (N s/m^2)	g (m/s^2)	σ_0 (N/m)	Re	Eo	$\frac{\rho_L}{\rho_G}$	$\frac{\mu_L}{\mu_G}$
1000	1	10	0.1	0.98	1.96	35	125	1000	100

If the density ratio is much greater than unity, the result of the phase-field variable may exceed the range $[-1, 1]$, leading to an unphysical solution. In this case, the density ratio reaches 1000, so some constraints should be applied to ensure convergence. In Sec. III A, the phase-field variable may escape the range from -1 to 1. When the density ratio is not so large, a slightly out-of-bounds phase-field variable is acceptable. However, when the density ratio is sufficiently large, the density and viscosity calculated by Eqs. (11) and (12) become negative at specific points, causing difficulties in convergence. To overcome these difficulties, an auxiliary variable $\bar{C}$ is defined to constrain the density and viscosity,[58]

$$\bar{C} = \begin{cases} C & \text{if } |C| \leqslant 1, \\ \operatorname{sign}(C) & \text{if } |C| > 1. \end{cases} \tag{28}$$

The density and viscosity can be derived by Eq. (28), so that the negative value is prevented. An improved description of density and viscosity is

$$\rho_M = \frac{1+\bar{C}}{2}\rho_L + \frac{1-\bar{C}}{2}\rho_G, \tag{29}$$

$$\mu_M = \frac{1+\bar{C}}{2}\mu_L + \frac{1-\bar{C}}{2}\mu_G. \tag{30}$$

Thereby, the corrected density and viscosity are calculated. Although the original phase-field variable is still out of bounds, the training result is satisfactory.

In Sec. II D, an adaptive time-marching strategy proposed by Wight and Zhao[56] is introduced to improve the convergence of the neural network. The process of a bubble rising in water is influenced by buoyancy and surface tension. Thus, the mean rising velocity of the bubble changes significantly with time, especially when the bubble begins to move upward. The optimizer trains the network at different times simultaneously, but the total loss at various times may not be of the same order of magnitude. The optimizer prefers to delay the loss, so the time with greater loss is optimized in priority. According to Eqs. (15)–(18), the velocity field determines the magnitude of the total loss,

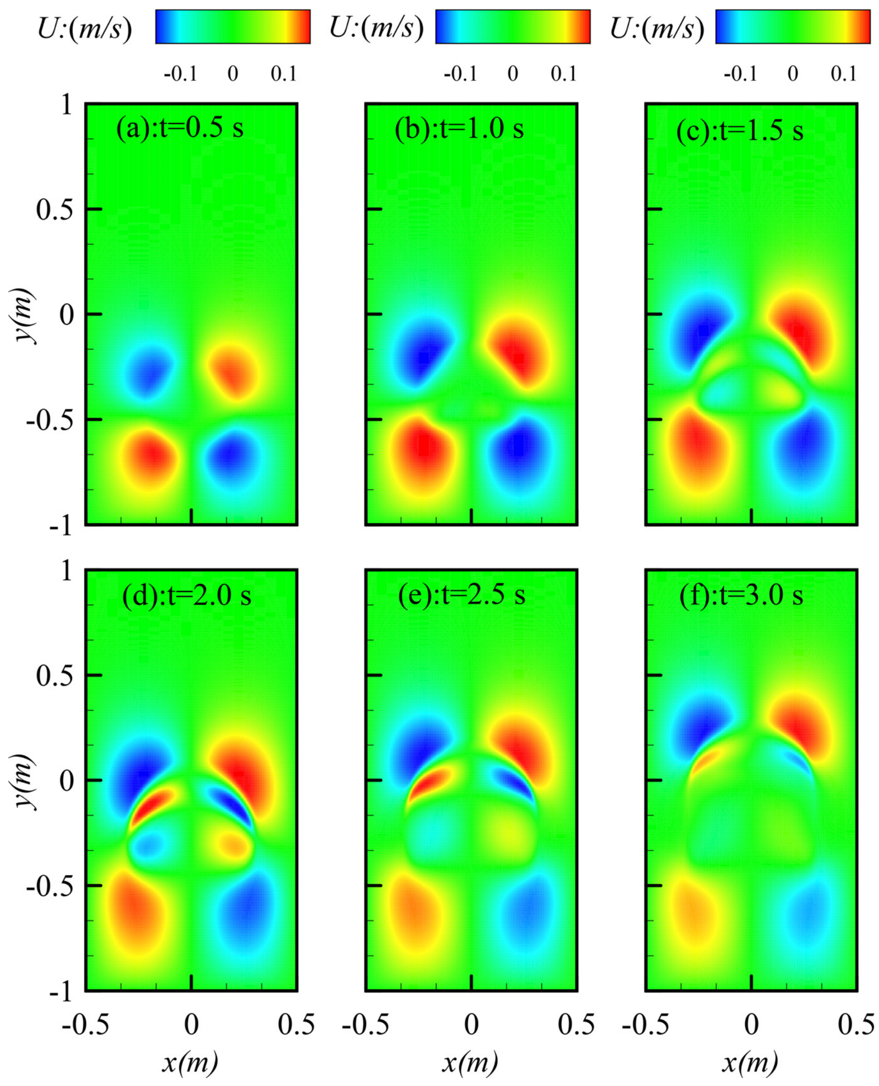

FIG. 9. Contours of horizontal velocity field u from 0.5 to 3 s from the present result: (a) $t = 0.5$, (b) $t = 1.0$, (c) $t = 1.5$, (d) $t = 2.0$, (e) $t = 2.5$, and (f) $t = 3.0$ s.

so the latter time is always optimized first. However, if the loss of the latter time cannot reach the same order of magnitude as the loss of the former time, the result will not converge the network hardly trains the former time.

To explain the difficulties, we compare multiple networks with a single network. In multiple networks, we divide the time interval $t \rightarrow [0, 3]$ s into seven time periods $(t_0, t_1, t_2, t_3, t_4, t_5, t_6) = (0, 0.5, 1, 1.5, 2, 2.5, 3\,\text{s})$, including intervals $[0, 0.5]$, $[0.5, 1]$, $[1, 1.5]$, $[1.5, 2]$, $[2, 2.5]$, and $[2.5\,\text{s}, 3\,\text{s}]$, whereas the entire time interval $[0\,\text{s}, 3\,\text{s}]$ is used in the single network. Each network in the time-marching strategy is identical to the network mentioned before, except that the sampling time is split to the corresponding interval. The training results in $t = 3$ s using various strategies are compared in Fig. 8. The result generated by the single network without improvement does not converge, whereas the result generated by the multiple networks with the time-marching strategy is adequate accurate. It is, thus, necessary to use the time-marching strategy.

In Figs. 9–11, PF-PINNs are used to describe the velocity distribution and interface evolution. The neural network successfully captures the velocity distribution near the interface, even though the density ratio is sufficiently large. To better illustrate the efficiency of the proposed method, a relative L_2 error ε in time t between the reference parameter $\hat{U}_{ref}$ and the predicted parameter $\hat{U}_{NN}$ is written as

$$\varepsilon(t) = \frac{\sqrt{\sum_{S\in(x,y)} |\hat{U}_{ref}(x,y,t) - \hat{U}_{NN}(x,y,t)|^2}}{\sqrt{\sum_{S\in(x,y)} |\hat{U}_{ref}(x,y,t)|^2}}, \tag{31}$$

where the subscript "*ref*" indicates the parameters obtained from Xiao *et al.*[42] and $\hat{U}$ has the velocity components (u, v). The relative L_2 errors at different time are given in Table IV. We find that the relative L_2 errors in u and v fluctuate from 0.0555 to 0.1512 and from 0.0356 to

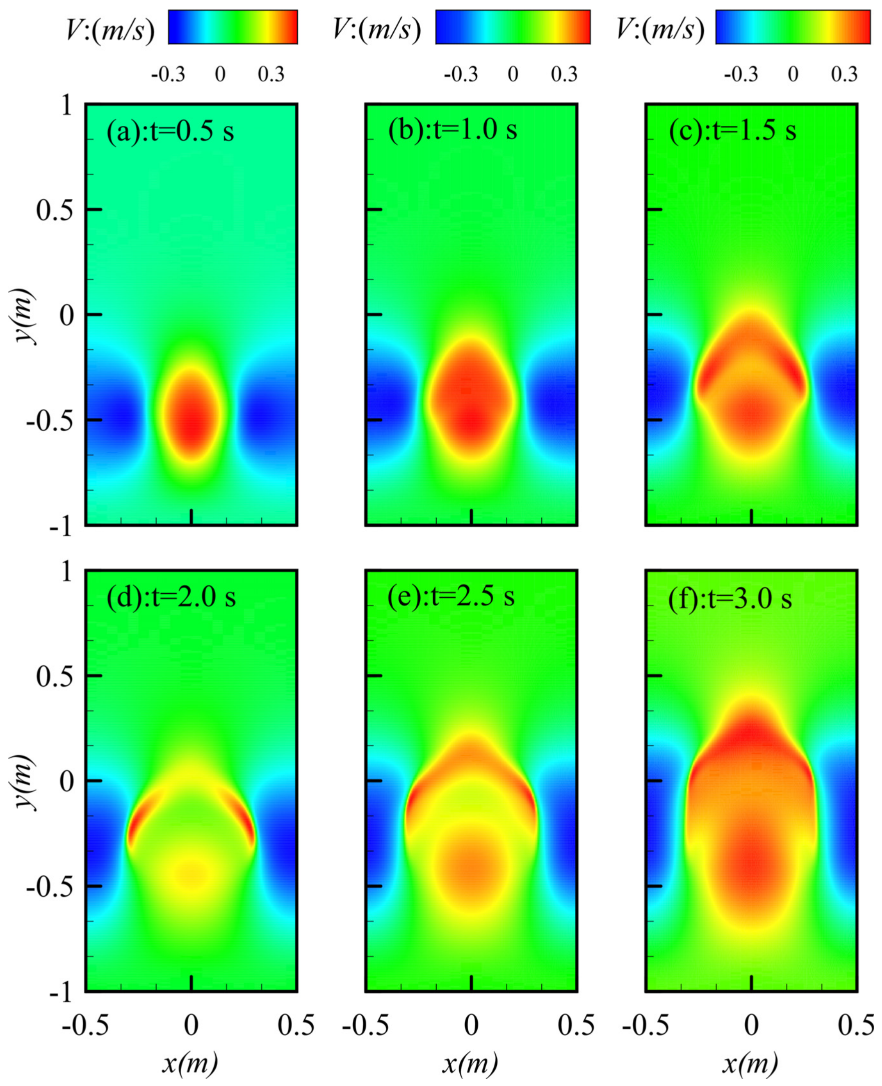

FIG. 10. Contours from the present result of vertical velocity field v from 0.5 to 3 s: (a) $t = 0.5$, (b) $t = 1.0$, (c) $t = 1.5$, (d) $t = 2.0$, (e) $t = 2.5$, and (f) $t = 3.0$ s.

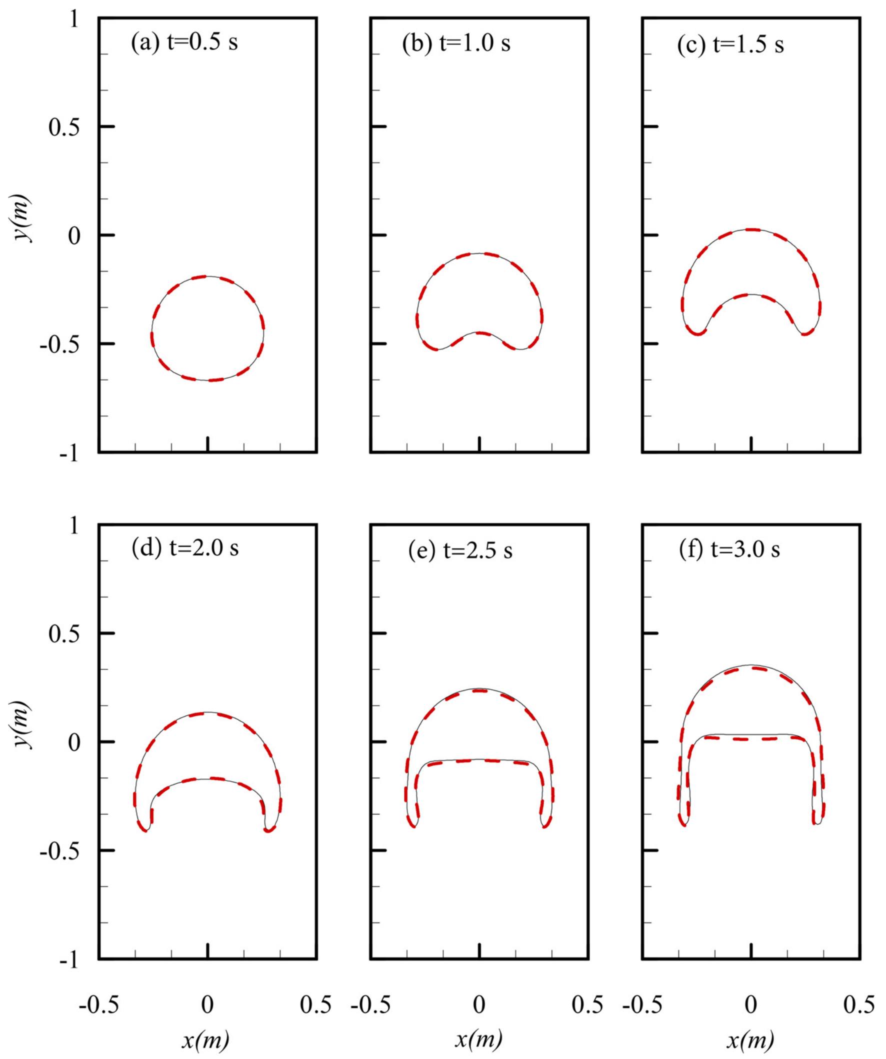

FIG. 11. Interfaces $C = 0$ of the rising bubble from 0.5 to 3 s: (a) $t = 0.5$, (b) $t = 1.0$, (c) $t = 1.5$, (d) $t = 2.0$, (e) $t = 2.5$, and (f) $t = 3.0$ s. Red-dashed lines represent the present result, whereas black solid lines represent the reference result reported by Aland and Voigt.[39]

0.0792, respectively. The maximum error reaches nearly 15%, confirming the effectiveness of the present method. The error in horizontal velocity u is greater than that in the vertical velocity v. In the bubble-rising case, the buoyancy is the main driving force in the bubble-rising procedure, so the loss term in the direction of gravity is crucial when the bubble is rising. Optimizing v can decrease total loss significantly, so the optimizer prefers to suppress the residual terms of the momentum term in the y direction rather than other terms.

In Fig. 11, we also examine the shape of bubble generated by the proposed neural network and reference result.[39] The velocity field near the bubble interface is hard to solve, due to the vast difference of physical parameters between the interface. However, the network still works well in the whole domain. We see that PF-PINNs can correctly express the interface evolution process in two-phase flow at large density ratios.

TABLE IV. Bubble rising at the large density ratio: relative L_2 errors of velocity fields for PF-PINNs at different times.

$\epsilon(t)$	$t = 0.5$ s	$t = 1$ s	$t = 1.5$ s	$t = 2$ s	$t = 2.5$ s	$t = 3$ s
ϵ_u	0.0555	0.0702	0.1347	0.1289	0.1512	0.1348
ϵ_v	0.0356	0.0438	0.0781	0.0650	0.0792	0.0705

To quantitatively assess the result, we apply benchmark quantities related to the bubble as per Huang *et al.*[40] The center of mass y_c and the rising velocity v_c are defined as

$$y_c(t) = \frac{\int_{S\in(x,y)} y \frac{1 - C(x,y,t)}{2} dS}{\int_{S\in(x,y)} \frac{1 - C(x,y,t)}{2} dS}, \tag{32}$$

$$v_c(t) = \frac{\int_{S\in(x,y)} v(x,y,t) \frac{1 - C(x,y,t)}{2} dS}{\int_{S\in(x,y)} \frac{1 - C(x,y,t)}{2} dS}. \tag{33}$$

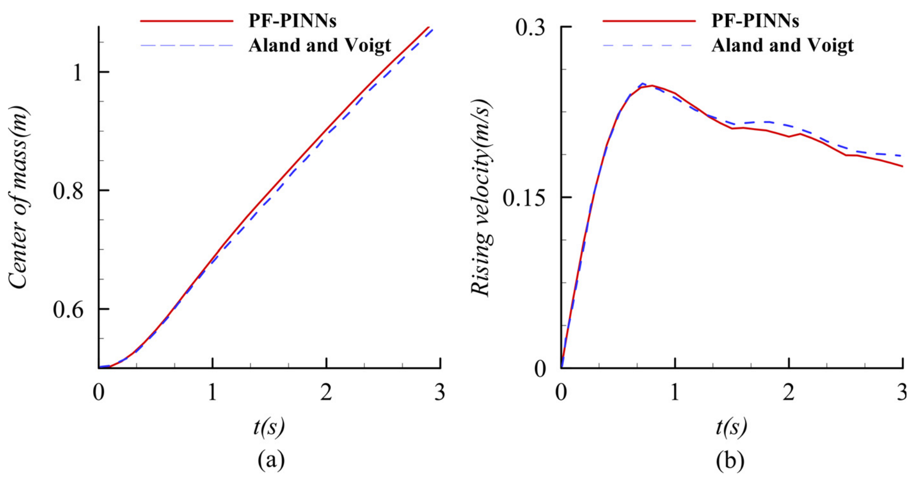

FIG. 12. Temporal evolution of (a) center of mass and (b) rising velocity defined by Eqs. (32) and (33). The red solid lines correspond to the present result, while the blue dashed lines correspond to the reference result obtained from Aland and Voigt.[39]

The center of mass y_c and the rising velocity v_c describe the dynamic behavior of the bubble. These quantities over the entire time sequence are plotted in Fig. 12. We sample from the neural network with $\Delta t = 0.1$ s, and the curve is compared with the reference result. Figure 12(a) shows that the bubble reaches the position of 1.094 4 m when $t = 3$ s using PF-PINNs, which agrees well with the baseline simulation reported by Aland and Voigt,[39] with a magnitude of 1.102 5 m. The maximum relative error of parameter y_c between PF-PINNs and the reference in Fig. 12 appears at $t = 3$ s, which reaches 0.7%. The center of mass is precisely enough, whereas the rising velocity transitions slightly after a period of time. Figure 12(b) summarizes the fluctuation of the rising velocity. The rising velocity of PF-PINNs increases significantly and approaches the maximum value of 0.248 1 m/s, which is close to the baseline maximum of 0.248 8 m/s in the same interface thickness. Then, the rising velocity decreases due to the deformation of the bubble. The estimated rising velocity of PF-PINNs in $t = 3$ s is 0.177 1 m/s, and the baseline result is 0.186 6 m/s. The maximum relative error of parameter v_c is located at $t = 3$ s, as expected, and the error approaches 5.3%. The averaged loss of center of mass and rising velocity over the entire time sequence are 0.48% and 3.7%, respectively. The PF-PINNs predict slower rising velocity and lower center of mass than the baseline result from $t = 0$ to 3 s due to the errors fluctuating in the training process. In the machine learning strategy, the solution to the proposed cases is obtained by training the neural network with a well-designed stochastic optimizer. The stochastic optimizer will train the network to fit the exact solution in the whole computational domain, including the time and space, so the errors will fluctuate in each time step when the time ranges from 1.5 to 3 s and from 0.1 to 1 s. However, an exception occurs when the time ranges from 0.8 to 1.6 s. The bubble deforms quickly at this stage, leading to an increase in relative errors. In conclusion, the relative error is accumulated in long time computation, but the error fluctuates in local time intervals. The accumulated error can be reduced by using a stricter initial constraint (i.e., adding higher weight for the initial condition in the first network). In general, the neural network can provide stable and high-precision simulation results.

IV. CONCLUSIONS AND FUTURE WORKS

This paper investigates the physics-informed neural networks by using the phase-field method (PF-PINNs) to directly simulate two-dimensional incompressible immiscible two-phase flow. In particular, the Cahn–Hillard equation is used as a high-precision interface-capturing method for two-phase flow at large density ratios. Physics-informed neural networks are introduced to tackle the disadvantages of the phase-field method, including the calculation of high-order derivates and randomly sampling methods near the interface without losing the ability to capture the interface.

We first validate this issue in the reversed single vortex. This case proves the interface tracking ability of PF-PINNs in two-phase flow, and the volume of the bubble in the whole process remains unchanged. Then, simply applying PF-PINNs to all time sequences may cause divergence of training, so we propose the time-marching strategy to help the network converge. A comparison test between a single network in the whole time domain and multiple networks with the time-marching strategy is proposed, and the results reveal that the time-marching strategy is required. Moreover, we need to confirm the performance of the proposed method in complex flow, so the bubble-rising case at the large density ratio is investigated to compare the accuracy with the reference result. Inspired by the existing numerical method, an auxiliary variable is defined to truncate the density and viscosity for improving convergence. PF-PINNs can precisely predict the interface evolution and recover the velocity fields around the bubble. Finally, several quantitative parameters are defined to assess the proposed method, including the relative L_2 error of velocities, the center of mass, and the rising velocity. The presented results are consistent with baseline simulation and with published results, indicating that PF-PINNs are adequate to describe bubble dynamics and to deal with two-phase flow at the large density ratio.

The presented methodology offers a way to simulate two-phase flow at the large density ratio, and it may be applied to solve other systems. However, it is still necessary to accelerate the training process of PF-PINNs and expand the ability of the proposed network. First, the structure of PF-PINNs is highly parallelized, so the distributed parallel in multiple GPUs shows a great potential to promote the practical application of PF-PINNs. In addition, decreasing the order of governing equations is helpful to save the cost of training a neural network. Finally, this paper does not include the application of the inverse problem for two-phase flow, which is essential for data assimilation. The estimation of physical parameters based on experimental data will be an important issue.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant Nos. 12122214, 11872065, 52006232, and 12002346) and the Youth Innovation Promotion Association CAS (Grant Nos. Y201906 and Y2022019).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

APPENDIX A: THE VALIDATION OF MASS CONSERVATION

The volume of the bubble reflects the mass conservation quantitively. The volume of the bubble in this manuscript is defined as follows:

$$V(t) = \int_{S\in(x,y)} \frac{1 - C(x,y,t)}{2} dS. \tag{A1}$$

The definition of Eq. (A1) is the same as the submitted manuscript. The time vs the volume of bubble is presented in Table V. The volume of bubble varies from 0.196 5 to 0.194 2 m^2. The result indicates that the volume of bubble hardly changes in each time, confirming the reliability of the proposed method in mass conservation.

APPENDIX B: PARAMETER STUDY ABOUT INTERFACE THICKNESS AND INTERFACE DIFFUSION

In the conventional phase-field method, interface thickness and interface diffusion are two representative parameters to assess the convergence behavior of a specific scheme in two-phase flow. The mobility M_0 is considered as a parameter that affects the diffusion of the interface, while the interface thickness is defined as ε directly. Before we choose the interface thickness with 0.01 m and the mobility with $10^{-4}\,\mathrm{m}/(\mathrm{N\,s})$, two comparative studies are employed to determine which parameter is acceptable. We will introduce these studies.

TABLE V. The volume of bubble in various times.

$t(\mathrm{s})$	0.0	0.2	0.4	0.6	0.8	1.0
$V(\mathrm{m}^2)$	0.1965	0.1962	0.1961	0.1962	0.1961	0.1963
$t(\mathrm{s})$	1.2	1.4	1.6	1.8	2.0	2.2
$V(\mathrm{m}^2)$	0.1959	0.1959	0.1961	0.1955	0.1956	0.1954
$t(\mathrm{s})$	2.4	2.6	2.8	3.0	⋯	⋯
$V(\mathrm{m}^2)$	0.1949	0.1947	0.1947	0.1943	⋯	⋯

TABLE VI. The maximum and minimum phase-field variables in different thicknesses at t=1 s.

ε	0.005 m	0.01 m	0.02 m
C_{min}	−1.0364	−1.0514	−1.0554
C_{max}	1.0288	1.0414	1.0380

A. Influence of the interface thickness

Three cases with various interface thicknesses, including 0.005, 0.01, and 0.02 m, are tested to investigate the influence of interface thickness. We train the neural network from $t = 0$ to $t = 1$ s and give the contour of phase-field variable. The maximum and minimum values of phase-field variable are presented in Table VI. The comparison of the interface profile in different interface thicknesses is also shown in Fig. 13. The results show that smaller interface thickness will suppress the out of range of phase-field variable C, but the interface profile deviates significantly at the same time. Decreasing the size of the spatial interval $(\Delta x, \Delta y)$ is helpful to ensure the convergence of PF-PINNs when $\varepsilon = 0.005$ m. If we increase the interface thickness to $\varepsilon = 0.02$ m, the dispersion of the interface will become more serious, which affects the accuracy of interface capturing, as shown in Fig. 14. Consequently, the interface thickness of 0.01 m is chosen as the parameter in PF-PINNs.

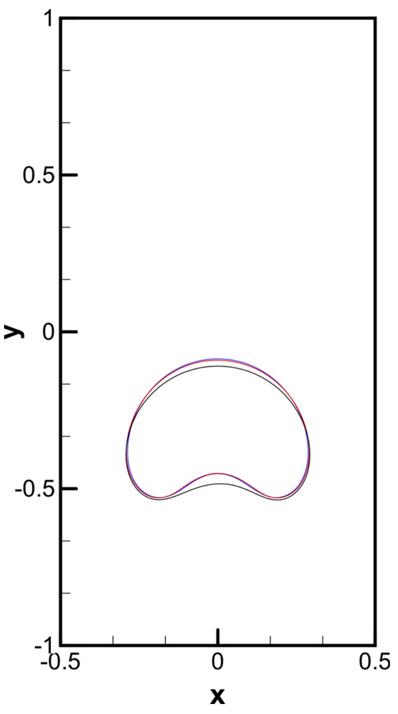

FIG. 13. Comparison of bubble shape between different ϵ. Blue: $\epsilon = 0.02$; red: $\varepsilon = 0.01$; black: $\varepsilon = 0.005$ m.

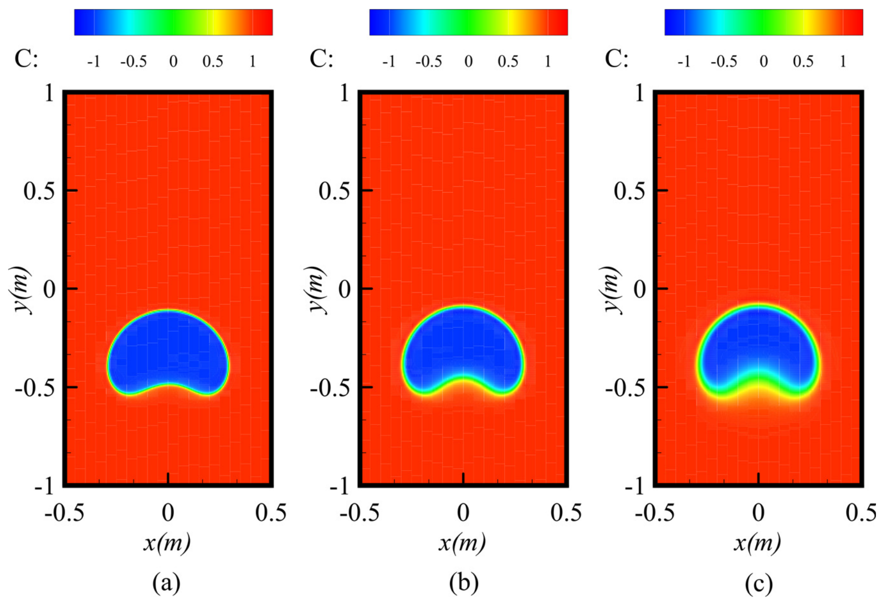

FIG. 14. The contours of phase-field variable C in various interface thicknesses: (a) $\varepsilon = 0.005$; (b) $\varepsilon = 0.01$; (c) $\varepsilon = 0.02$ m.

B. Influence of the mobility

Similar to the former cases, we adjust the mobility, while the other parameters remain constant. The values of mobility are set to 10^{-3}, 10^{-4}, and 10^{-5} in three different test cases, respectively. The shapes of the bubble in various mobility when $t = 1$ s are plotted in Fig. 15. We observe that the choice of mobility does not affect the capturing accuracy significantly. The position of the interface coincides well in all cases. To ensure the stability of the proposed network, we choose 10^{-4} as the value of mobility.

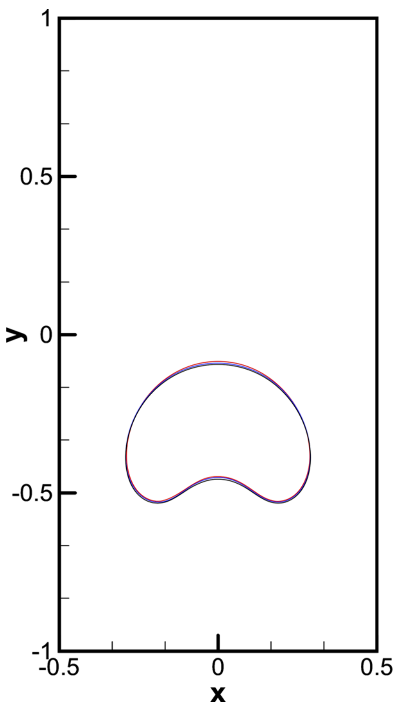

FIG. 15. Comparison of interfaces in various mobilities. Red: 10^{-3}; blue: 10^{-4}; black: 10^{-5}.

REFERENCES

[1]P. J. Schmid and J. Sesterhenn, *Dynamic Mode Decomposition of Numerical and Experimental Data* (Cambridge University Press, 2010), pp. 5–28.
[2]T. Murata, K. Fukami, and K. Fukagata, "Nonlinear mode decomposition with convolutional neural networks for fluid dynamics," J. Fluid Mech. **882**, A13 (2020).
[3]C. W. Rowley and S. T. M. Dawson, "Model reduction for flow analysis and control," Annu. Rev. Fluid Mech. **49**, 387–417 (2017).
[4]S. E. Ahmed, S. Pawar, O. San, A. Rasheed, T. Iliescu, and B. R. Noack, "On closures for reduced order models—A spectrum of first-principle to machine-learned avenues," Phys. Fluids **33**, 091301 (2021).
[5]P. Wu, S. Gong, K. Pan, F. Qiu, W. Feng, and C. Pain, "Reduced order model using convolutional auto-encoder with self-attention," Phys. Fluids **33**, 077107 (2021).
[6]R. Maulik, B. Lusch, and P. Balaprakash, "Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders," Phys. Fluids **33**, 037106 (2021).
[7]B. Colvert, M. Alsalman, and E. Kanso, "Classifying vortex wakes using neural networks," Bioinspiration Biomimetics **13**, 025003 (2018).
[8]B. L. Li, Z. X. Yang, X. Zhang, G. W. He, and L. Shen, "Using machine learning to detect the turbulent region in flow past a circular cylinder," J. Fluid Mech. **905**, A10 (2020).
[9]M. Y. Wang and M. S. Hemati, "Detecting exotic wakes with hydrodynamic sensors," Theor. Comput. Fluid Dyn. **33**, 235–254 (2019).
[10]H. Li and J. Tan, "Cluster-based Markov model to understand the transition dynamics of a supersonic mixing layer," Phys. Fluids **32**, 056104 (2020).
[11]Z. Zhang, X. D. Song, S. R. Ye, Y. W. Wang, C. G. Huang, Y. R. An, and Y. S. Chen, "Application of deep learning method to Reynolds stress models of channel flow based on reduced-order modeling of DNS data," J. Hydrodyn. **31**, 58–65 (2019).
[12]H. Xiao, J. L. Wu, J. X. Wang, R. Sun, and C. Roy, "Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach," J. Comput. Phys. **324**, 115–136 (2016).

[13]A. P. Singh, S. Medida, and K. Duraisamy, "Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils," AIAA J. **55**, 2215–2227 (2017).
[14]D. Schmidt, R. Maulik, and K. Lyras, "Machine learning accelerated turbulence modeling of transient flashing jets," Phys. Fluids **33**, 127104 (2021).
[15]L. Zhu, W. Zhang, J. Kou, and Y. Liu, "Machine learning methods for turbulence modeling in subsonic flows around airfoils," Phys. Fluids **31**, 015105 (2019).
[16]S. Taghizadeh, F. D. Witherden, Y. A. Hassan, and S. S. Girimaji, "Turbulence closure modeling with data-driven techniques: Investigation of generalizable deep neural networks," Phys. Fluids **33**, 115132 (2021).
[17]I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," IEEE Trans. Neural Networks **9**, 987–1000 (1998).
[18]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," J. Comput. Phys. **378**, 686–707 (2019).
[19]L. Lu, X. H. Meng, Z. P. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," SIAM Rev. **63**, 208–228 (2021).
[20]H. Gao, L. Sun, and J.-X. Wang, "Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels," Phys. Fluids **33**, 073603 (2021).
[21]H. Wang, Y. Liu, and S. Wang, "Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network," Phys. Fluids **34**, 017116 (2022).
[22]A. Arzani, J.-X. Wang, and R. M. D'Souza, "Uncovering near-wall blood flow from sparse data with physics-informed neural networks," Phys. Fluids **33**, 071905 (2021).
[23]S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," Annu. Rev. Fluid Mech. **52**, 477–508 (2020).
[24]G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. F. Wang, and L. Yang, "Physics-informed machine learning," Nat. Rev. Phys. **3**, 422–440 (2021).
[25]X. W. Jin, S. Z. Cai, H. Li, and G. E. Karniadakis, "NSFnets (Navier–Stokes flow nets): Physics-informed neural networks for the incompressible Navier–Stokes equations," J. Comput. Phys. **426**, 109951 (2021).
[26]N. Geneva and N. Zabaras, "Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks," J. Comput. Phys. **403**, 109056 (2020).
[27]R. Laubscher, "Simulation of multi-species flow and heat transfer using physics-informed neural networks," Phys. Fluids **33**, 087101 (2021).
[28]H. Xu, W. Zhang, and Y. Wang, "Explore missing flow dynamics by physics-informed deep learning: The parameterized governing systems," Phys. Fluids **33**, 095116 (2021).
[29]S. Z. Cai, Z. C. Wang, F. Fuest, Y. J. Jeon, C. Gray, and G. E. Karniadakis, "Flow over an espresso cup: Inferring 3D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks," J. Fluid Mech. **915**, A102 (2021).
[30]Z. P. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," Comput. Methods Appl. Mech. Eng. **360**, 112789 (2020).
[31]S. F. Wang and P. Perdikaris, "Deep learning of free boundary and Stefan problems," J. Comput. Phys. **428**, 109914 (2021).
[32]A. B. Buhendwa, S. Adami, and N. A. Adams, "Inferring incompressible two-phase flow fields from the interface motion using physics-informed neural networks," Mach. Learn. Appl. **4**, 100029 (2021).
[33]C. W. Hirt and B. D. Nichols, "Volume of fluid (VoF) method for the dynamics of free boundaries," J. Comput. Phys. **39**, 201–225 (1981).
[34]M. Sussman, P. Smereka, and S. Osher, "A level set approach for computing solutions to incompressible two-phase flow," J. Comput. Phys. **114**, 146–159 (1994).
[35]D. Adalsteinsson and J. A. Sethian, "A fast level set method for propagating interfaces," J. Comput. Phys. **118**, 269–277 (1995).
[36]G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y. J. Jan, "A front-tracking method for the computations of multiphase flow," J. Comput. Phys. **169**, 708–759 (2001).
[37]J. U. Brackbill, D. B. Kothe, and C. Zemach, "A continuum method for modeling surface tension," J. Comput. Phys. **100**, 335–354 (1992).
[38]H. Ding, P. D. Spelt, and C. Shu, "Diffuse interface model for incompressible two-phase flows with large density ratios," J. Comput. Phys. **226**, 2078–2095 (2007).
[39]S. Aland and A. Voigt, "Benchmark computations of diffuse interface models for two-dimensional bubble dynamics," Int. J. Numer. Methods Fluids **69**, 747–761 (2012).
[40]Z. Y. Huang, G. Lin, and A. M. Ardekani, "Consistent, essentially conservative and balanced-force phase-field method to model incompressible two-phase flows," J. Comput. Phys. **406**, 109192 (2020).
[41]T. W. Zhang, J. Wu, and X. J. Lin, "An interface-compressed diffuse interface method and its application for multiphase flows," Phys. Fluids **31**, 122102 (2019).
[42]Y. Xiao, Z. Zeng, L. Q. Zhang, J. Z. Wang, Y. W. Wang, H. Liu, and C. G. Huang, "A spectral element-based phase field method for incompressible two-phase flows," Phys. Fluids **34**, 022114 (2022).
[43]M. E. Gurtin, D. Polignone, and J. Vinals, "Two-phase binary fluids and immiscible fluids described by an order parameter," Math. Models Methods Appl. Sci. **06**, 815–831 (1996).
[44]C. Ma, J. Wu, and T. Zhang, "A high order spectral difference-based phase field lattice Boltzmann method for incompressible two-phase flows," Phys. Fluids **32**, 122113 (2020).
[45]J. Kou, X. Wang, M. Zeng, and J. Cai, "Energy stable and mass conservative numerical method for a generalized hydrodynamic phase-field model with different densities," Phys. Fluids **32**, 117103 (2020).
[46]A. Dadvand, M. Bagheri, N. Samkhaniani, H. Marschall, and M. Worner, "Advected phase-field method for bounded solution of the Cahn–Hilliard Navier–Stokes equations," Phys. Fluids **33**, 053311 (2021).
[47]A. De Rosis and E. Enan, "A three-dimensional phase-field lattice Boltzmann method for incompressible two-components flows," Phys. Fluids **33**, 043315 (2021).
[48]Y. Wang, C. Shu, J. Y. Shao, J. Wu, and X. D. Niu, "A mass-conserved diffuse interface method and its application for incompressible multiphase flows with large density ratio," J. Comput. Phys. **290**, 336–351 (2015).
[49]J. D. Van der Waals, "The thermodynamic theory of capillarity under the hypothesis of a continuous variation of density," J. Stat. Phys. **20**, 200–244 (1979).
[50]J. W. Cahn and J. E. Hilliard, "Free energy of a nonuniform system. I. Interfacial free energy," J. Chem. Phys. **28**, 258–267 (1958).
[51]J. W. Cahn and J. E. Hilliard, "Free energy of a nonuniform system. III. Nucleation in a two-component incompressible fluid," J. Chem. Phys. **31**, 688–699 (1959).
[52]A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," J. Mach. Learn. Res. **18**, 1 (2018).
[53]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980 (2014).
[54]X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (JMLR Workshop and Conference Proceedings, 2010), pp. 249–256.
[55]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Curran Associates, Inc, 2019), Vol. 32; available at https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.
[56]C. L. Wight and J. Zhao, "Solving Allen–Cahn and Cahn–Hilliard equations using the adaptive physics informed neural networks," arXiv:2007.04542 (2020).
[57]S. R. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, and L. Tobiska, "Quantitative benchmark computations of two-dimensional bubble dynamics," Int. J. Numer. Methods Fluids **60**, 1259–1288 (2009).
[58]S. C. Dong and J. Shen, "A time-stepping scheme involving constant coefficient matrices for phase-field simulations of two-phase incompressible flows with large density ratios," J. Comput. Phys. **231**, 5788–5804 (2012).