RESEARCH ARTICLE | MARCH 24 2022

Predicting micro-bubble dynamics with semi-physics-informed deep learning

Hanfeng Zhai ; Quan Zhou ; Guohui Hu

AIP Advances 12, 035153 (2022)
https://doi.org/10.1063/5.0079602

Predicting micro-bubble dynamics with semi-physics-informed deep learning

Cite as: AIP Advances 12, 035153 (2022); doi: 10.1063/5.0079602
Submitted: 23 November 2021 • Accepted: 8 March 2022 •
Published Online: 24 March 2022

Hanfeng Zhai,[1,2] Quan Zhou,[1] and Guohui Hu[1,a)]

AFFILIATIONS

[1] Shanghai Key Laboratory of Mechanics in Energy Engineering, School of Mechanics and Engineering Science, Shanghai Institute of Applied Mathematics and Mechanics, Shanghai University, Shanghai 200072, People's Republic of China
[2] Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, New York 14850, USA

[a)]Author to whom correspondence should be addressed: ghhu@staff.shu.edu.cn

ABSTRACT

Utilizing physical information to improve the performance of the conventional neural networks is becoming a promising research direction in scientific computing recently. For multiphase flows, it would require significant computational resources for neural network training due to the large gradients near the interface between the two fluids. Based on the idea of the physics-informed neural networks (PINNs), a modified deep learning framework `BubbleNet` is proposed to overcome this difficulty in the present study. The deep neural network (DNN) with separate sub-nets is adopted to predict physics fields, with the semi-physics-informed part encoding the continuity equation and the pressure Poisson equation $\mathcal{P}$ for supervision and the time discretized normalizer to normalize field data per time step before training. Two bubbly flows, i.e., single bubble flow and multiple bubble flow in a microchannel, are considered to test the algorithm. The conventional computational fluid dynamics software is applied to obtain the training dataset. The traditional DNN and the `BubbleNet`(s) are utilized to train the neural network and predict the flow fields for the two bubbly flows. Results indicate the `BubbleNet` frameworks are able to successfully predict the physics fields, and the inclusion of the continuity equation significantly improves the performance of deep NNs. The introduction of the Poisson equation also has slightly positive effects on the prediction results. The results suggest that constructing semi-PINNs by flexibly considering the physical information into neural networks will be helpful in the learning of complex flow problems.

I. INTRODUCTION

Machine learning (ML) has achieved tremendous success in the last decade due to the availability of big data and computer resources. ML is the study of algorithms that allow computer programs to automatically improve their performance through experiences.[1] AlphaGo bursts the public's interest by showing the huge potential of machine learning and artificial intelligence.[2,3] The ML techniques are becoming a promising research method in diverse scientific fields, specifically in genomics,[4,5] public health,[6–10] and medicine.[11–14]

Deep neural network (DNN), as one of the most prominent tools of ML, has been adopted to tackle various physics problems, including turbulence,[15] flow control,[16] heat transfer,[17] and combustion.[18] These deep learning applications have grown drastically in recent years, mainly on learning physical equations and inferring dynamics. Numerous frameworks have henceforth been proposed, such as SINDy[19] and PDE-FIND[20] using sparse regression to identify the governing equations for nonlinear dynamic systems; graph kernel network,[21] Fourier neural operator,[22] and MeshfreeFlowNet[23] using convolutional neural networks to learn image mapping the physics fields; and deep potential,[24] DeePMD,[25] and DeePCG[26] using deep neural nets to map the molecular potentials at the microscale. In 2018, Raissi *et al.*[27–29] proposed a deep learning framework, called physics-informed neural networks (PINNs), for identifying and inferring dynamics of physical systems governed by partial differential equations (PDEs). The strategy of PINN can be simplified by encoding governing PDEs into the loss function as a soft physics constraint, namely, the "physics-informed" part. Regarding PINN, Lu *et al.* proposed DeepXDE,[30] a deep learning library for convenient implementation of PINNs; and later DeepONet,[31] numerical implementation of nonlinear neural networks operators,[32] for learning and inferring nonlinear operators, which were later applied to electroconvection multiphysics[33] and hypersonics.[34]

The PINN series has been developed to solve numerous problems, including the following: fractional PINNs for predicting fractional PDEs; conservative PINNs for nonlinear conservation laws;[35] extended PINNs, a PINN approach for space–time domain decomposition;[36] and parareal PINNs, a PINN solver decomposing a long-time problem into many independent short-time problems supervised by an inexpensive/fast coarse-grained (CG) solver.[37] PINNs have achieved great success for predicting laminar flows,[38] high speed flows,[39] and turbulence.[40] Lin *et al.*[41] applied the previously mentioned DeepONet to predict bubble growth dynamics described by the Rayleigh–Plesset (R–P) equation in the continuum regime and dissipative particles dynamics (DPD) for the microscale.

Despite the significant development of the informed machine learning[42] architecture, the consideration of physical equations in the loss function usually requires high order differentiation of physical quantities. Specifically, in two-phase flows, the phase value at the interface between different fluids exhibits a drastic variation from 0 to 1, making the calculation of the gradient highly difficult. Therefore, high-resolution training data would be a prior condition for the success of the algorithm, especially for variables with high gradients. This will greatly increase the amount of deep learning computation. Furthermore, high-resolution data can hardly be obtained in many experiments.

In the present study, we provide an engineering-orientated idea to overcome this difficulty. Bubbly flows are considered to test the algorithm as they are a kind of classic fluid mechanics problems with a high gradient of density. Bubbly flows have widely been applied in biomedical engineering, such as blood–brain barrier[43–45] and drug delivery.[46,47] The bubble pinch-off effect confined in a microchannel is one of the most studied phenomena in fluid mechanics,[48–52] depicting deformation and movement of single bubble dynamics. The flow with multiple bubbles displays complexity due to the interactions between bubbles.[53,54]

Our work is inspired by the ideas of PINNs[27] and the subsequent work using DeepONet to infer bubble dynamics by Lin *et al.*[41] Instead of applying the full hydrodynamic equations to supervise the training process, we aim to achieve satisfactory results based on partial physical information. Specifically, we inserted only the fluid continuity condition (divergence free of velocity) and the pressure Poisson equation (denoted by $\mathcal{P}$) into the loss function, which can be described as a neural network with semi-physical information. A modified version of PINNs, nominated as `BubbleNet` in this paper, is proposed for this purpose.

The `BubbleNet`(s) with/without the Poisson equation are considered and trained in the present study. The advantages of the algorithms are as follows: (1) To save computer resources, the gradient computation from automatic differentiation[55] of the phase function is avoided to some extent. (2) Both physical information and conventional NN losses are encoded in the semi-physics-informed network, allowing the framework to take advantage of deep learning and physical equations during training. (3) In 2D or axisymmetric flow, the velocities are inferred from a latent function (stream function), saving computation resources compared with inferring two components of velocities separately. Another point that differs from the traditional approach is the time discretized normalizer (TDN) utilized for normalizing variables per time step to capture physics information more accurately for the NN training.

This paper is arranged as follows: in Sec. II, to obtain the training data, the conventional computational fluid dynamics (CFD) software is utilized to simulate numerically two cases of the bubble flow, i.e., the single bubble flow and multiple bubble flow through a microchannel. Then, the numerical results are briefly discussed. In Sec. III, we introduce the traditional DNNs and the `BubbleNet` algorithms, respectively. Both networks are trained based on the numerical results. We, hence, obtain the predictions of physical quantities in bubble flows and analyze the absolute errors from machine learning in Sec. IV. Finally, some conclusions of this study are drawn in Sec. V.

II. NUMERICAL IMPLEMENTATION OF BUBBLE FLOWS

A. Problem formulation

Bubble flows are commonly encountered in numerous biological applications. Two-phase flows (air and water) are governed by the Navier–Stokes equation,

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u}, \tag{1}$$

where ρ is the density of the fluid, $\mathbf{u} = (u, v)$ is the 2D velocity vector, p is the fluid pressure, and μ is the dynamic viscosity.

In the present study, the level set algorithm is applied to bubble flows in the microchannel to obtain the dataset for machine learning training. In the level set method, the interface between air and water is represented by a certain level set or isocontours of a globally defined function, e.g., the level set function $\phi = \phi(x, y, t)$ in 2D spaces.[56] In our system, ϕ is a smooth step function that equals zero for water and one for air. Across the interface, there is a smooth transition from zero to one. Thus, the interface is defined by 0.5 in level set ϕ.

The level set phase function ϕ takes the form

$$\frac{\partial\phi}{\partial t} + \nabla\phi\cdot\mathbf{u} = F, \tag{2}$$

where F includes terms with higher-order derivatives of ϕ, designed to keep the interface compact.

Bubble flows involve interactions between two fluids with different physical properties. Here, we set ρ_l as the water (liquid) density, ρ_g as the air (gas) density, μ_l as the water viscosity, and μ_g as the air viscosity. The density and viscosity in the flow can be connected through the level set function as follows:

$$\begin{aligned}\rho &= \rho_l + \phi(\rho_g - \rho_l),\\ \mu &= \mu_l + \phi(\mu_g - \mu_l).\end{aligned} \tag{3}$$

To simulate the interfaces between liquid and gas, Eq. (2) can be rewritten in the following form:

$$\frac{\partial\phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = \gamma\nabla\cdot\left(\epsilon_{ls}\nabla\phi - \phi(1-\phi)\frac{\nabla\phi}{|\nabla\phi|}\right), \tag{4}$$

where the terms on the left-hand side describe the motion of the interface, while those on the right-hand side are necessary for

TABLE I. The level set parameter ϵ_{ls} value for both the two bubbly flow cases with dense and coarse meshing, respectively.

ϵ_{ls}	Dense meshing	Coarse meshing
Single bubble flow	0.430	4.382
Multiple bubble flow	0.083	0.299

numerical stability. γ is the reinitialization parameter, which determines the amount of reinitialization or stabilization of the level set function, equal to 1 in our cases. ϵ_{ls} is the parameter controlling the interface thickness, which equals to the mesh largest size,[56] as shown in Table I.

The continuity equation for fluids is written as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0,$$

where for incompressible fluids, such an equation can be simplified to $\partial_y u + \partial_x v = 0$, i.e., the divergence free condition of velocity, $\nabla \cdot \mathbf{u} = 0$. As we will discuss later, our goal is to insert such a condition into the neural networks (NNs) for better training and predictions.

The geometric properties of interfaces can be described by the unit vector normal to the interface $\mathbf{n} = \nabla\phi/|\nabla\phi|_{\phi=0.5}$. The curvature of the level set phase function ϕ can be calculated as $\kappa_{ls} = -\nabla \cdot \boldsymbol{n}|_{\phi=0.5}$. To simulate the bubble flows numerically, Eq. (4) is discretized and solved for given boundary conditions (BCs), initial conditions (ICs), and a specific space–time domain with meshing for discretizations.

B. Numerical setup

Two cases are considered for investigations: (1) single bubble flow and (2) multiple bubble flow, confined in a microchannel. For the single bubble case, the initial diameter of the bubble is set to be $d = 4$ μm, and the microchannel has a length of 15 μm and a width of 5 μm. The pressure difference $\Delta p = 10$ Pa is imposed in the axial direction to drive the flow, and the pressure at the end of the channel is kept as constant pressure $p_0 = 799.932$ Pa (6 mmHg), corresponding to the pressure of the interstitial fluid in the human brain[57,58] and in lymph flow.[59] The initial conditions (ICs) are set as the pressure p_0, with room temperature as 293.15 K, as shown in Fig. 1(a). This numerical setup has been designed to simulate the bubble transportation in brain vessels for the investigation of the blood–brain barrier.[43–45] Our model is inspired by the work of Miao *et al.*,[43] where a single bubble is confined in a microchannel with both diameters ~5 μm.

For the multiple bubble flow, 60 micro-bubbles, each micro-bubble of diameter 3 μm, are randomly distributed in a 2D microchannel with a length of 100 μm and a width of 50 μm, as shown in Fig. 1(b). The BCs and ICs are the same as the single bubble case. The moving mesh is adopted for spatial discretization and computations. The single bubble flow case generates 24 182 meshes and the multiple bubble flow generates 75 302 meshes initially. For the single bubble case, the simulation is run for 5000 μs, while for the multiple bubble case, the simulation is run for 3000 μs. $\mathrm{Re} = \rho UL/\mu$, where L is the characteristic length, is equal to the microchannel's width in our cases. The dynamic viscosity of the bubbly flow is denoted by μ. $U = -R^2/3\mu \cdot dp/dx$ is the average velocity of 2D Poiseuille flow. $R = L/2$ is the half width of the channel. The Reynolds number Re is ~0.007 for the single bubble case and 0.010 for the multiple bubble case.

To produce training datasets of the bubble flows in the microchannel, the computational fluid dynamics (CFD) technique is utilized for 2D bubbly flow simulations, adopting the time-dependent level set algorithms for modeling bubbles, using COMSOL Multiphysics®.

In Sec. III, the numerical results obtained in CFD simulation are used for training the NNs. However, as we claimed before, high-resolution data will lead to a huge demand of computer resources in NN training, which may not be available in experiments. For this consideration, we only use a "coarsened" (~1/10) dataset for

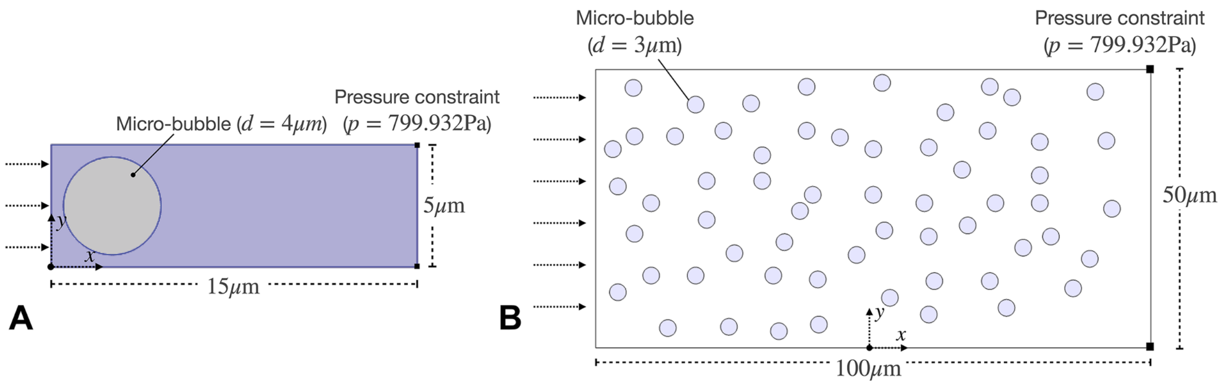
Micro-bubble ($d = 4\mu m$)
Pressure constraint ($p = 799.932$Pa)
5μm
y
x
15μm
A
Micro-bubble ($d = 3\mu$m)
Pressure constraint ($p = 799.932$Pa)
50μm
y
x
100μm
B

FIG. 1. Diagram of the two bubble flow cases. (a) Model for the single bubble flow case. The gray area indicates the initial position of the bubble, with diameter $d = 4$ μm, and the blue area indicates the water. The bubble is constrained in a 2D microchannel with a length of 15 μm and a height of 5 μm. The pressure difference $\Delta p = 10$ Pa is imposed between the two sides of the microchannel. The initial condition is given as a constant pressure of 799.932 Pa. The coordinate is centered at the left bottom point of the microchannel. (b) Model for the multiple bubble flow case. The blue circles indicate 60 bubbles each with diameter $d = 4$ μm constrained in a 2D microchannel with a length 100 μm and a height of 50 μm, with surrounding fluid of water. The BCs and ICs are the same as the single bubble flow case. The coordinates' origin is located at the central bottom point in (b).

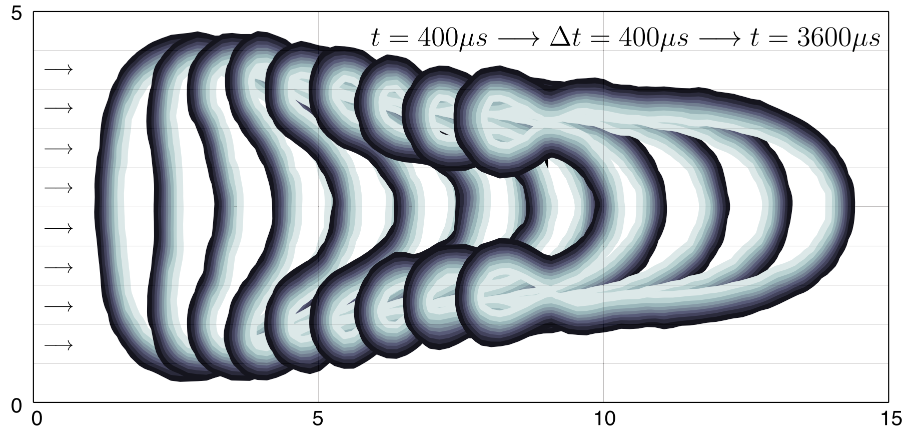

FIG. 2. Simulation results depicting the bubble movement (liquid–gas interface) of the single bubble flow. The bubble deformation is illustrated at nine different time steps, from 400 to 3600 μs with a time interval of 400 μs.

training NNs. Therefore, we also perform CFD simulations for the two bubble flows with coarse meshing (also ~1/10 of the dense meshing) for comparison, i.e., 2419 meshes for the single bubble flow and 3766 meshes for the multiple bubble flow. In solving Eq. (4), ϵ_{ls} is related to the meshing size. Hence, ϵ_{ls} is listed for the four cases in Table I for reference.

C. CFD results

The single bubble motion in a microchannel is depicted in Fig. 2, where the snapshots of bubbles are illustrated at nine different time steps, from $t = 400$ μs to $t = 3600$ μs with an interval of $\Delta t = 400$ μs. It shows that the front side of the bubble flows forward in a parabolic shape since the flow velocity is relatively higher in the middle of the channel, while the rear side of the bubble is significantly stretched due to viscous shear. The configuration of the bubble is similar to the deformation of red blood cells traveling in a microchannel reported by Tomaiuolo *et al.*[60]

The multiple bubbles' motion indicates that bubbles tend to collide and ruptured with each other, as shown in Fig. 3. The collision of two "daughter" bubbles, which is demonstrated by the contact of bubbles' outer interface (light blue part), results in the formation of bigger bubbles, as can be observed in every consecutive inset. Such results are consistent with the numerical study of bubble behaviors reported in Ref. 61.

In multiphase flow simulations, some numerical factors (i.e., meshing, BCs, ICs, and solvers) could cause losses of components, leading to inaccurate results. To validate our simulations, the liquid–gas volume ratio during the whole computation process, signifying the conservation of the components, is shown in Fig. 4 for both the single and multiple bubble cases, together with the comparison with the simulations with coarse meshing. In our

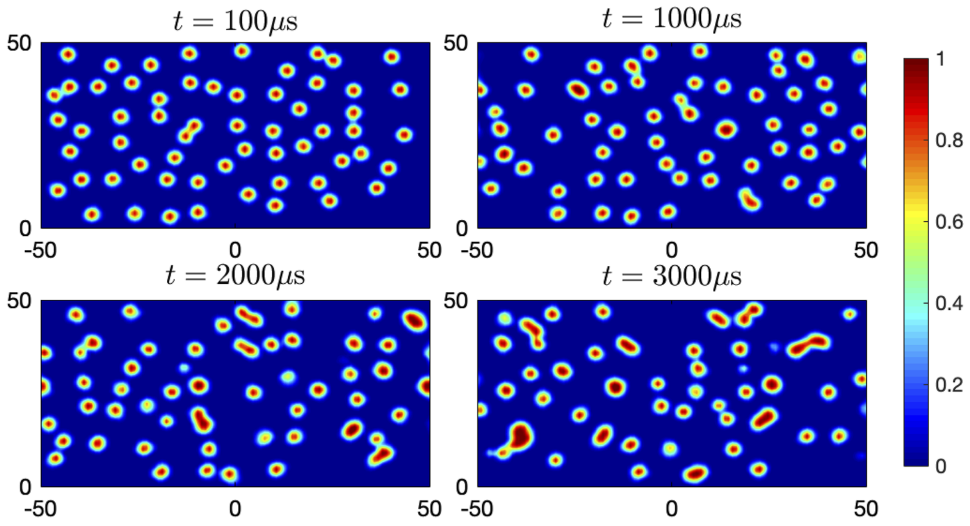

FIG. 3. Snapshots of the multiple bubble flow at different time steps, from 100 to 3000 μs. The two-phase flow is depicted by the phase function ϕ, where $\phi = 0$ for water and $\phi = 1$ for air.

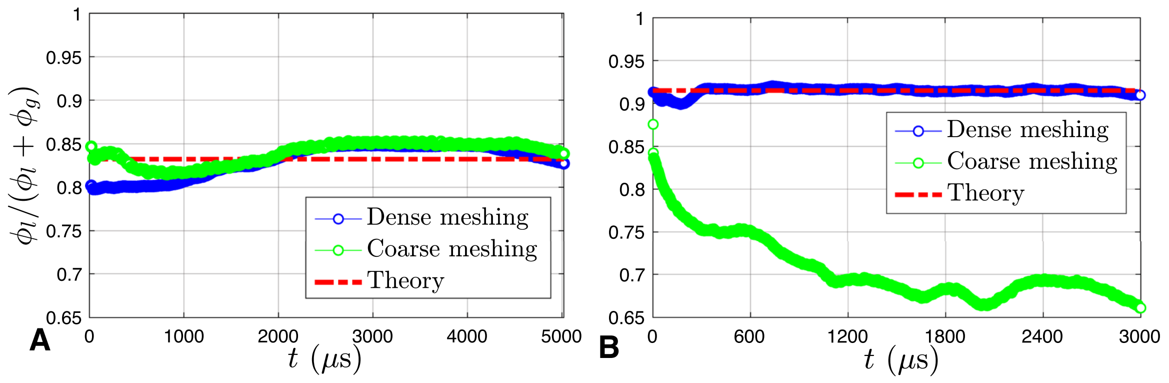

FIG. 4. Comparison of the liquid ratio $[\phi_l/(\phi_l+\phi_g)]$ between the CFD computation results of dense meshing (blue dots), those of coarse meshing (green dots), the theoretical value (red dotted line) with respect to time t. Note that the vertical coordinates are plotted in the range of $[0.65, 1]$ for comparison. (a) The liquid ratio of the single bubble flow case. (b) The liquid ratio of the multiple bubble flow case.

cases, if A stands for the area (2D volume) of the bubbly flows, the theoretical liquid ratio (or water ratio) can be estimated by $\phi_0 = A_{water}/(A_{water}+A_{air})$. Initially, it has $\phi_0 = 0.8324$ for the single bubble flow and $\phi_0 = 0.915$ for the multiple bubble flow case. Based on Eq. (3), the liquid ratio of simulation can be represented as $\phi_l/(\phi_l+\phi_g)$. Figure 4 shows that the simulations with dense meshing generally agree with the theoretical results (blue dotted lines), with only subtle fluctuation around the theoretical value LG, validating the accuracy of our simulations. It is also found that the mass conservation is almost kept for coarse meshing simulation of the single bubble flow, yet an apparent liquid component loss is observed for the multiple bubble flow. Therefore, fine meshing densities are required for complex two-phase flow systems, especially for multiple bubble flows.

III. DEEP LEARNING ALGORITHMS

In this section, we briefly introduce the basis of deep learning and NNs. Then, we present our approach for using DNNs and our framework `BubbleNet` to predict physical fields of bubbly flows. Both the DNN and `BubbleNet` trained on the coarsened simulation data and the bubble motion at a specific time are predicted.

A. Traditional DNNs

A neural network (NN) consists of input layers, hidden layers, and output layers, fully connected within. DNNs are NNs with multiple hidden layers that are able to approximate nonlinear mapping between function spaces.

Here, we apply a DNN with four sub-nets: Net_u, Net_v, Net_p, and Net_ϕ (see the Appendix) for predicting the physics fields u, v, p, and ϕ, respectively. Each sub-net consists of 9 layers with 30 neurons for each layer. The input quantities are the field data in the space–time domain, namely, x, y, t. The Adam optimizer and the "L-BFGS-B" optimization method are adopted in the training process. Each neuron is activated by the tanh function. The maximum iteration for "L-BFGS-B" optimization is 500 000.

To reduce the computational resources, the training data are obtained by coarsening the CFD results with fine meshes, which is given by interpolation

$$\begin{aligned}&[x_{train},\ y_{train},\ t_{train}]\\&\quad=\left[x(1:\Delta^s:end),\ y(1:\Delta^s:end),\ t(1:\Delta^t:end)\right],\end{aligned}$$

where Δ^s and Δ^t are the spatial and temporal intervals, respectively. For the single bubble case, we obtain 2419×126 (space $\times$ time) training data and 3766×101 for the multiple bubble case on the spatial–temporal domain.

The coarsened dataset is then normalized using the `mapminmax` function of MATLAB® before training. If the coarsened data obtained from the simulation of the bubbly flow are defined as $\mathcal{U}=(u,\ v,\ p,\ \phi)$, the training data $\mathcal{W}$ can be obtained through normalizing $\mathcal{U}$. The function `mapminmax` normalization operates the data by

$$\mathcal{W}=\frac{\mathcal{U}-\mathcal{U}_{min}}{\mathcal{U}_{max}-\mathcal{U}_{min}},\tag{5}$$

where $\mathcal{U}_{max}$ and $\mathcal{U}_{min}$ denote the maximum and minimum value of the coarsened simulation data in the whole spatial–temporal domain, respectively. Such a process can be simplified to a normalization function $\mathbb{N}$, written as $\mathcal{W}=\mathbb{N}(\mathcal{U})$.

For the single bubble case, the DNNs are trained for 10000 iterations on the normalized data, and we aim to predict the physics fields at $t = 2000\ \mu$s. For the single bubble case, the DNNs are trained for 200 000 iterations on the normalized data, and the target time is $t = 1500\ \mu$s.

B. Semi-physics-informed neural networks

Physics-informed neural networks (PINNs), as introduced, encode physics information into the loss function, imposing the NN to approximate the real physics equations during training. The original PINNs[27–29] and their modified versions[30,31] prefer to encode the whole physics governing equations together with related BCs and

ICs into the NN's loss function. Such approaches have been extensively studied and successfully applied to many physical systems, as introduced in Sec. I. However, encoding full equations into the losses might consume considerable computational resources in automatic differentiation. Furthermore, for problems such as two-phase flows, there essentially exists a drastic variation of level set function ϕ and density at the interfaces between the two fluids, placing high demand on the accuracy of the calculation of the gradient at and around the interfaces. The inaccuracy of gradient calculation will lead to difficulties in neural network training. In traditional CFD, this problem is usually solved by increasing the grid density, which will inevitably rise the training time of the neural network substantially. As Karniadakis *et al.*[62] claimed, if plenty of data are trained on a NN, PINN can be employed to approximate the time and spatial gradient so that one can apply the model on such data to get a PDE expression. In such a regime, the NNs are able to approximate data with good accuracy even with no physical guidance. In comparison, full physics are preferred only at the regime of small data. In the present study, we aim to predict the bubble flow with satisfactory results based on partial physical information to reduce the computer resource required for the flow field with high gradients.

Our algorithm `BubbleNet` encodes the continuity equation of incompressible fluids and the pressure Poisson equation in the inference process, namely, the semi-physics-informed neural networks, eliciting the latent function ψ for predicting the velocity fields u, v,

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}.$$

Thus, the continuum condition (divergence free of velocity), $\nabla \cdot \mathbf{u} = 0$, is automatically satisfied. Meanwhile, this reduces one sub-net for 2D or axisymmetric flows, which saves considerable computation resources for initialization and training. Furthermore, the introduction of stream function avoids gradient calculation of velocity vectors in the loss function, improving the efficiency of the neural networks.

Simultaneously, the pressure Poisson equation is also included within the losses to improve the accuracy of prediction, which writes as

$$\nabla^2 p = \rho \frac{\nabla \cdot \mathbf{u}}{\Delta t} - \rho \nabla \cdot (\mathbf{u} \cdot \nabla \mathbf{u}) + \mu \nabla^2 (\nabla \cdot \mathbf{u}).$$

Both the processes are achieved through automatic differentiation on the output physics fields.

1. Time discretized normalization (TDN)

In the present semi-PINN algorithm, Eq. (5) is still used as normalized operator $\mathbb{N}$ acting on dataset $\mathcal{U}$. However, different from the traditional method, $\mathcal{U}_{max}$ and $\mathcal{U}_{min}$ are considered as the maximum and minimum of the coarsened CFD data at each time step, respectively, which is called time discretized normalization (TDN). The reason for this treatment lies in the significant variation of physical quantities in the flow field, as shown in Fig. 5. In Fig. 5, the variations of the maximum values of velocity magnitude, $U = \sqrt{u^2 + v^2}$, and pressure, p, are depicted with respect to time. The TDN will be helpful to eliminate the inaccuracy caused by these variations. The training data, the optimization method, and the iterations are the same for `BubbleNet` as in DNN.

2. Loss function

In the present framework, the mean squared error (MSE) is used for computing the deviation of predictions and training data in the loss function. If we use $\mathcal{W}$ to represent the normalized dataset and $\mathcal{W} = (u,\ v,\ p,\ \phi)$, then the loss function $\mathcal{L}$ takes the form

$$\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m}\left(\mathcal{W}_{pred(i)} - \mathcal{W}_{train(i)}\right)^2 + \frac{1}{m}\sum_{i=1}^{m}\left(\nabla^2 p_{(i)}\right)^2,$$

where $\mathcal{W}_{pred}$ is the predictions of the NN training and $\mathcal{W}_{train}$ is the normalized training data obtained from CFD simulations. $\nabla^2 p_{(i)}$ denotes the pressure field in the training sets i. m is the training data number.

The schematic for our proposed semi-PINN architecture `BubbleNet` is shown in Fig. 6, and the details of the algorithm are

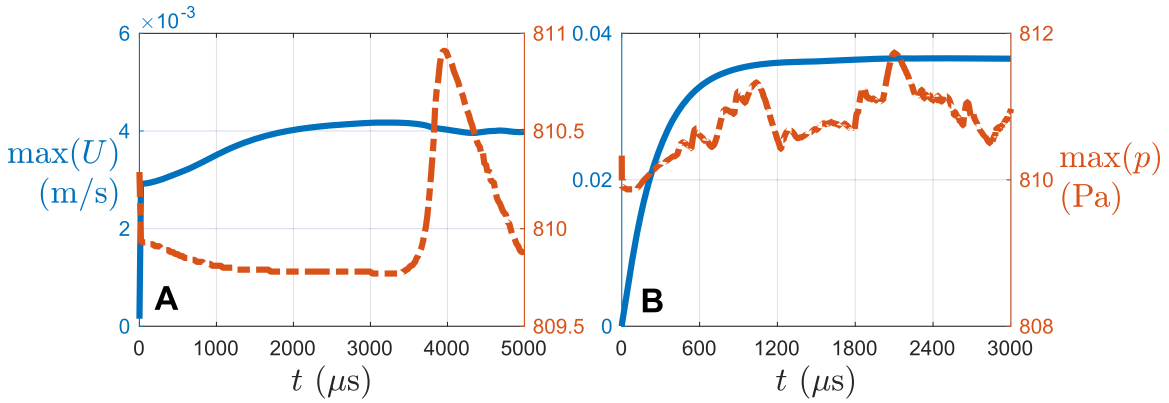

FIG. 5. The variations of the maximum value of velocity magnitude U and pressure p with time for the (a) single bubble flow (solid line) and (b) multiple bubble flow (dashed line).

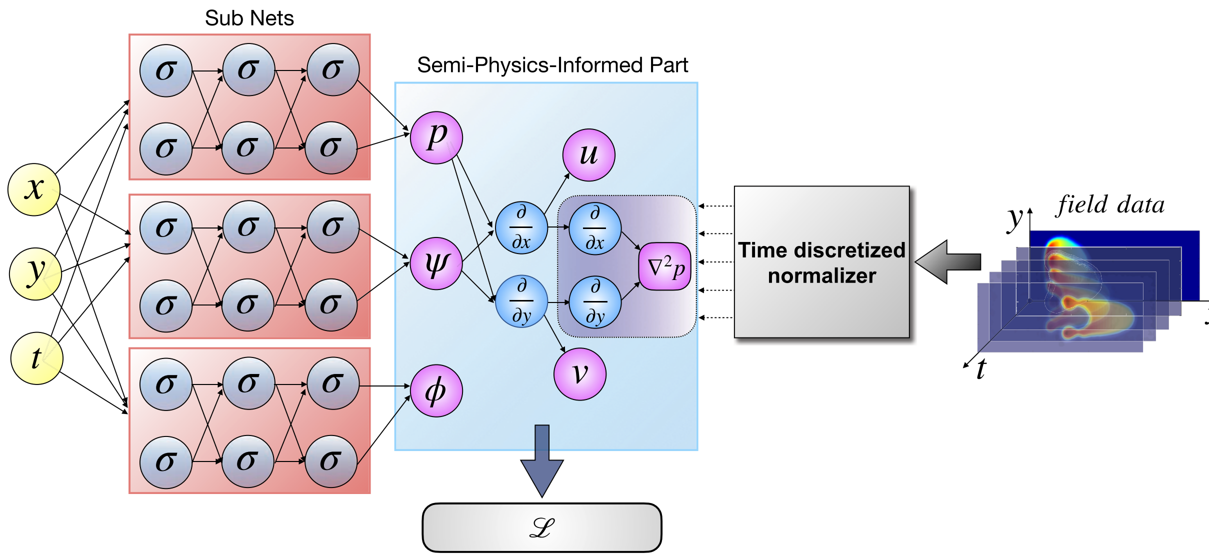

FIG. 6. Schematic diagram of the deep learning framework `BubbleNet`, consisting of three sub-nets for inferring p, ψ, ϕ, respectively, each having 9 hidden layers and 30 neurons. The semi-physics-informed part infers velocities u, v with the automatic differentiation through the fluid continuity equation, and the pressure Poisson equation is also inserted in the loss function. The time discretized normalizer is applied to normalize the training data per time step. The Poisson equation is represented by $\mathcal{P}$ (the shaded purple part in the Diagram). The loss function consists of the residual of inferred p, u, v, ϕ, and the pressure Poisson equation $\mathcal{P}$.

described in the Appendix, where we use three sub-nets: Net$_\psi$, Net$_p$, and Net$_\phi$, to predict ψ, p, and ϕ, respectively, and compute the velocities through automatic differentiations from ψ.

The variations of losses $\mathcal{L}$ in iterations of the DNNs and `BubbleNet` for both the single bubble and multiple bubble simulations are shown in Fig. 7. Figure 7(a) indicates traditional DNN exhibits lower losses and with longer iterations for single bubble case, while both `BubbleNet`(s) stop training at an earlier stage with higher losses. The higher losses may be accounted for the additional errors resulted from the physical information. Figure 7(b) shows that `BubbleNet` exhibits lower losses with more iterations and training. Yet, both DNNs display similar fluctuating losses (blue solid line in Fig. 7). Both the `BubbleNet`(s) have similar magnitude and trends in losses as indicated from the red solid lines and black dotted lines. The consideration of the Poisson equation $\mathcal{P}$ has only minor effects on the variation of the losses with time.

IV. RESULTS OF MACHINE LEARNING

A. Single bubble flow

For the single bubble flow case, both DNN and `BubbleNet`(s) are used to predict its physical variables, with comparison to the

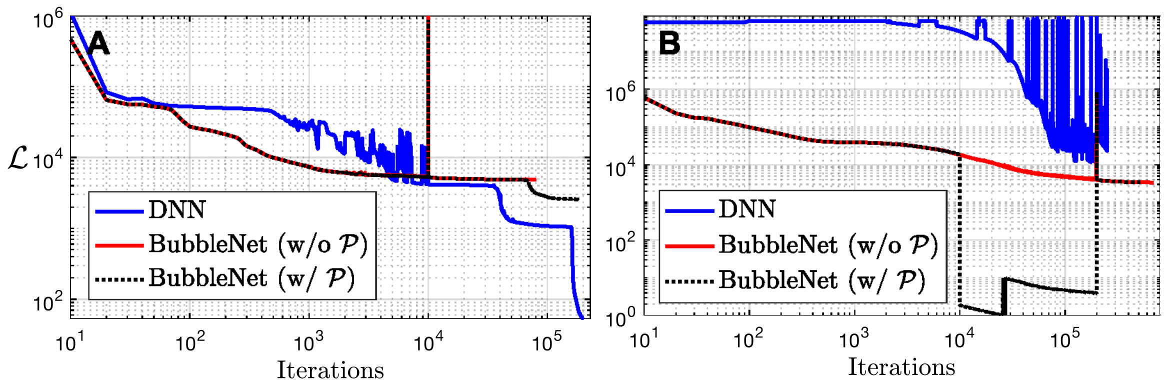

FIG. 7. The variations of losses $\mathcal{L}$ in iterations of the DNNs and `BubbleNet` for (a) the single bubble and (b) multiple bubble simulations. The blue solid line represents the losses of DNN, the red solid line represents for the losses of `BubbleNet` with no Poisson equation inserted, and the black dotted line denotes the losses of the `BubbleNet` with the Poisson equation.

2D CFD simulations results at t = 2000 μs. The predicted physics fields u, v, p, ϕ are shown in Fig. 8. `BubbleNet`(s) outperforms traditional DNN on approximating the physical trends on the velocity fields. One advantage of `BubbleNet`(s) origins from the application of TDN. To eliminate the negative effects during learning induced by the significant variation of the physical quantities with time, as shown in Fig. 5, enforcing a normalization on the time domain is obviously helpful. In comparison, normalization on the whole temporal–spatial domain results in inaccuracy in capturing the features of velocities at a specific time.

The consideration of physical information has benefited the prediction from the neural networks, even with only a coarse dataset. The `BubbleNet` with $\mathcal{P}$ approximates quantities more accurately than `BubbleNet` without $\mathcal{P}$ for the two velocities fields, as compared with the third and fourth columns in Fig. 8, which will be further discussed through analyzing the absolute errors of the predictions. This can be ascribed to the fact that the Poisson equation in the losses serves as an "inner supervision" on Net$_p$, allowing the NN to train on the other physical variables (u, v, ϕ) on their corresponding sub-nets more comprehensively. Including more physics is possible to further improve the performance of the networks. This result supports our idea that the semi-PINN, i.e., the combination of physical information and traditional neural networks, could be flexible in the construction of the network framework, obtaining satisfying results meeting engineering needs, especially when acquiring a huge amount of training data is impossible.

Both the DNN and `BubbleNet` (without $\mathcal{P}$) display good accuracy on the phase function ϕ and also on the overall numerical magnitudes of the pressure gradient. However, they do not successfully capture the bubble shape feature details in the pressure field, as shown in the third row in Fig. 8. To overcome this defect, the Poisson equation is included in the `BubbleNet` for supervision. However, this has only a negligible influence on the results. This is because the subtle differences in pressure magnitude depicting the bubble shape are too small compared with the large pressure range ([800, 810] shown in the third row of Fig. 8). Such detailed features missing in pressure will be discussed further. As for the prediction regarding ϕ, the relatively large variation from 0 to 1 is easier to be detected by the NN, as shown in the fourth column of Fig. 8.

To quantitatively compare the performance of the algorithms, the absolute error $|\epsilon|$ of predictions are estimated in the present study. $\mathcal{U}$ is the coarsened CFD simulation results $[u, v, p, \phi]$. $\mathcal{U}_{pred}$ is the prediction obtained from deep learning, and the absolute error is defined by

$$|\epsilon_{\mathcal{U}}| = |\mathcal{U}_{pred} - \mathcal{U}|. \tag{6}$$

The averaged error $\overline{|\epsilon_{\mathcal{U}}|} = \frac{\sum_{i=1}^{m}(|\epsilon_{\mathcal{U}}|)}{m}$ is also calculated to evaluate the algorithms.

The absolute errors corresponding to the four physical quantities $|\epsilon_u|$, $|\epsilon_v|$, $|\epsilon_p|$, $|\epsilon_\phi|$ for the single bubble flow are shown in Fig. 9. It can be found that both `BubbleNet`(s) display higher accuracy on the predictions of all the variables from the absolute errors, which confirms our observations in Fig. 8. The `BubbleNet` with $\mathcal{P}$ exhibits lower errors than `BubbleNet` without $\mathcal{P}$ on the velocities,

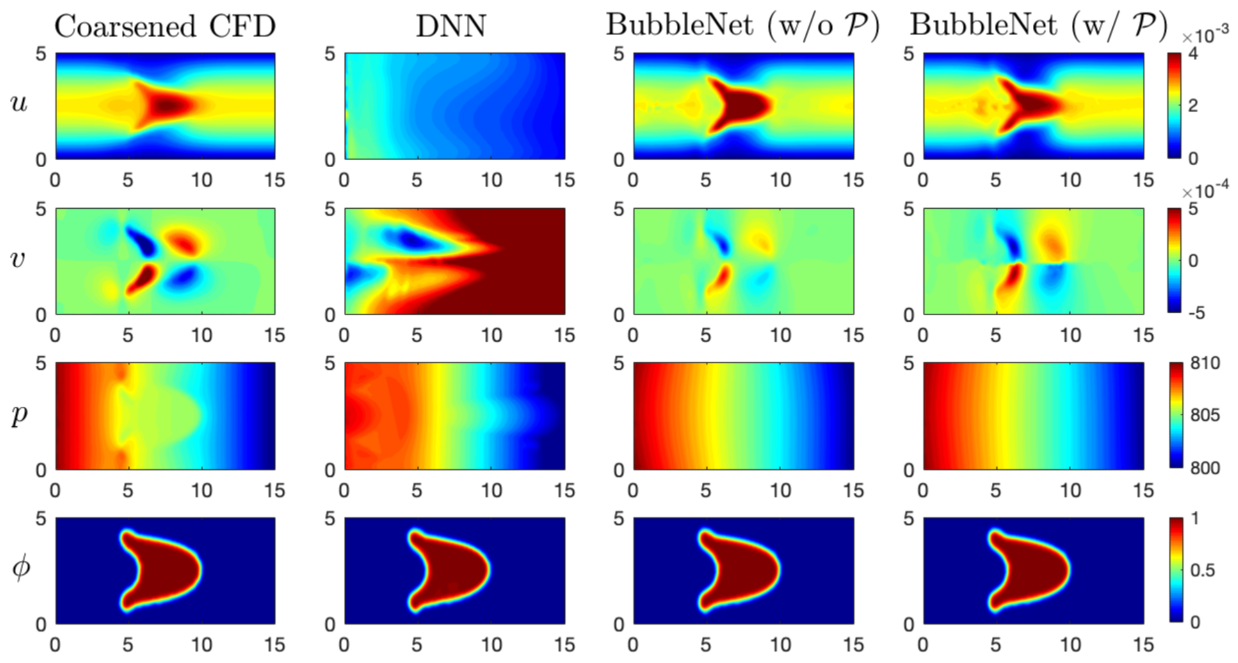

FIG. 8. The comparison of the physical quantities u, v, p, ϕ obtained from the CFD simulation results (coarsened), DNN, and `BubbleNet`(s). The four rows illustrate the four physical fields, respectively. The first column is the coarse training data based on the CFD simulation results. The other columns illustrate the predicted physical quantities from the traditional DNN and `BubbleNet`(s).

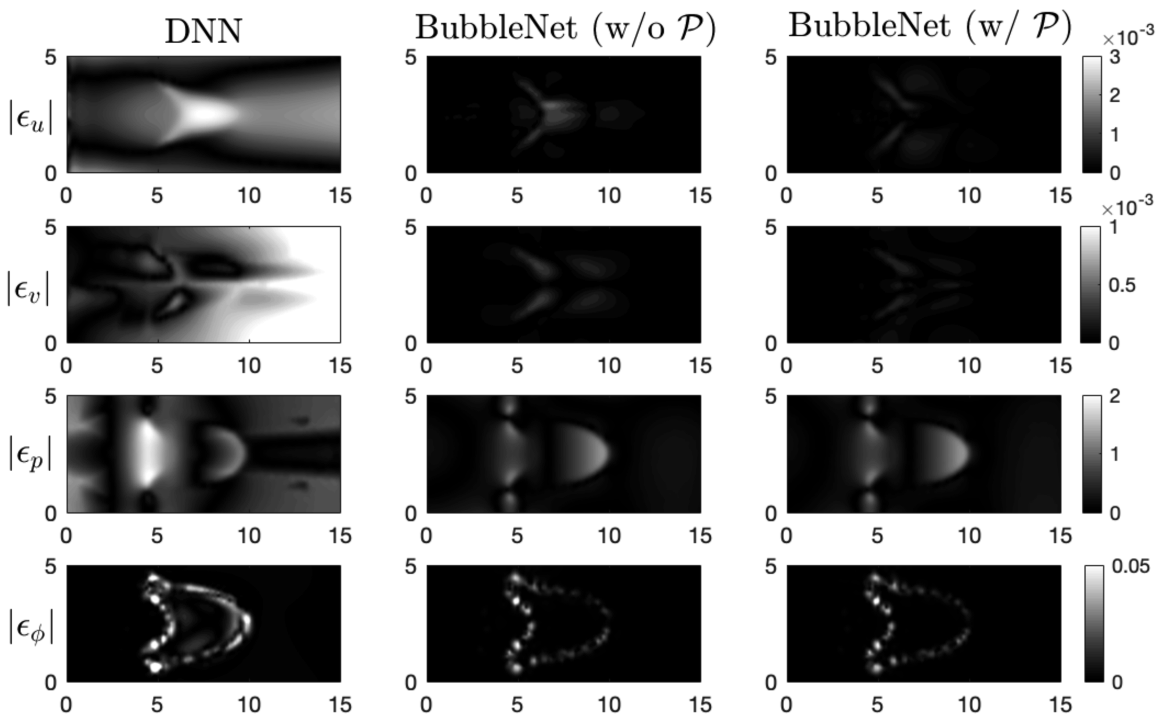

FIG. 9. The comparison of the absolute errors $|\epsilon_u|$, $|\epsilon_v|$, $|\epsilon_p|$, $|\epsilon_\phi|$ between DNN and `BubbleNet` (with and without $\mathcal{P}$) for the single bubble flow case. The four rows describe the errors of the predictions for the four physical quantities u, v, p, ϕ.

as indicated in the third and fourth columns of Fig. 9. However, significant improvement is not observed on the pressure and the phase function. To quantitatively demonstrate this result, the average absolute errors $|\overline{\epsilon_U}|$ are computed from Eq. (6), as shown in Table II. It indicates that $|\overline{\epsilon_u}|$ and $|\overline{\epsilon_v}|$ are remarkably reduced for `BubbleNet` with $\mathcal{P}$, whereas $|\overline{\epsilon_p}|$ and $|\overline{\epsilon_\phi}|$ remain nearly unchanged for both `BubbleNet`(s) in Table II. Together with the visualization in Fig. 9, it can be deduced that the two `BubbleNet`(s) display approximately the same accuracy on the two physical quantities. This implies that the auxiliary physics-informed part $\mathcal{P}$ mainly improves the accuracy on the velocities yet not directly on the pressure field. It might be inferred more accurately if we increase the density of the training dataset since the second-order differentiation is requested in the Poisson equation.

TABLE II. The mean value of the absolute errors of the three deep learning frameworks for the single bubble flow case.

	$\lvert\overline{\epsilon_u}\rvert$	$\lvert\overline{\epsilon_v}\rvert$	$\lvert\overline{\epsilon_p}\rvert$	$\lvert\overline{\epsilon_\phi}\rvert$
DNN	9.8832×10^{-4}	5.1092×10^{-4}	0.5817	0.0113
`BubbleNet` without $\mathcal{P}$	1.2869×10^{-4}	3.9572×10^{-5}	0.2105	0.0019
`BubbleNet` with $\mathcal{P}$	8.5603×10^{-5}	3.0938×10^{-5}	0.2167	0.0018

In summary, considering the effect of $\mathcal{P}$ on the results of the prediction, we can further expand our previous hypothesis that the "physics-informed" part serves as inner supervision to that the additional inner supervision may not have direct influences on its corresponding values or sub-net (Net$_p$ in our case) but is helpful to improve the overall accuracy of the predictions.

B. Multiple bubble flow

The multiple bubble flow is more complicated to be predicted due to its significant variation in physical variables, especially for small training sets. From predictions of the deep learning frameworks shown in Fig. 10, it can be found that generally `BubbleNet`(s) performs more accurately than the traditional NN, particularly on the level set function. It is not surprising that DNN fails to estimate the level set function due to its remarkable variation in the training set.

As for the prediction of the velocity vector, `BubbleNet`(s) basically predict the general trend of its change, especially for the y-component, but apparently with undesired fluctuations in its horizontal component. The introduction of the continuity condition greatly improves the prediction of the velocity in y-direction. However, due to the small magnitude of velocity in this direction, the error caused by the derivation of the stream function on the coarse grid also leads to some deviation in the estimation of the velocity in x-direction. In contrast, the DNN is better at estimating the horizontal velocity but displays a larger deviation in the vertical velocity,

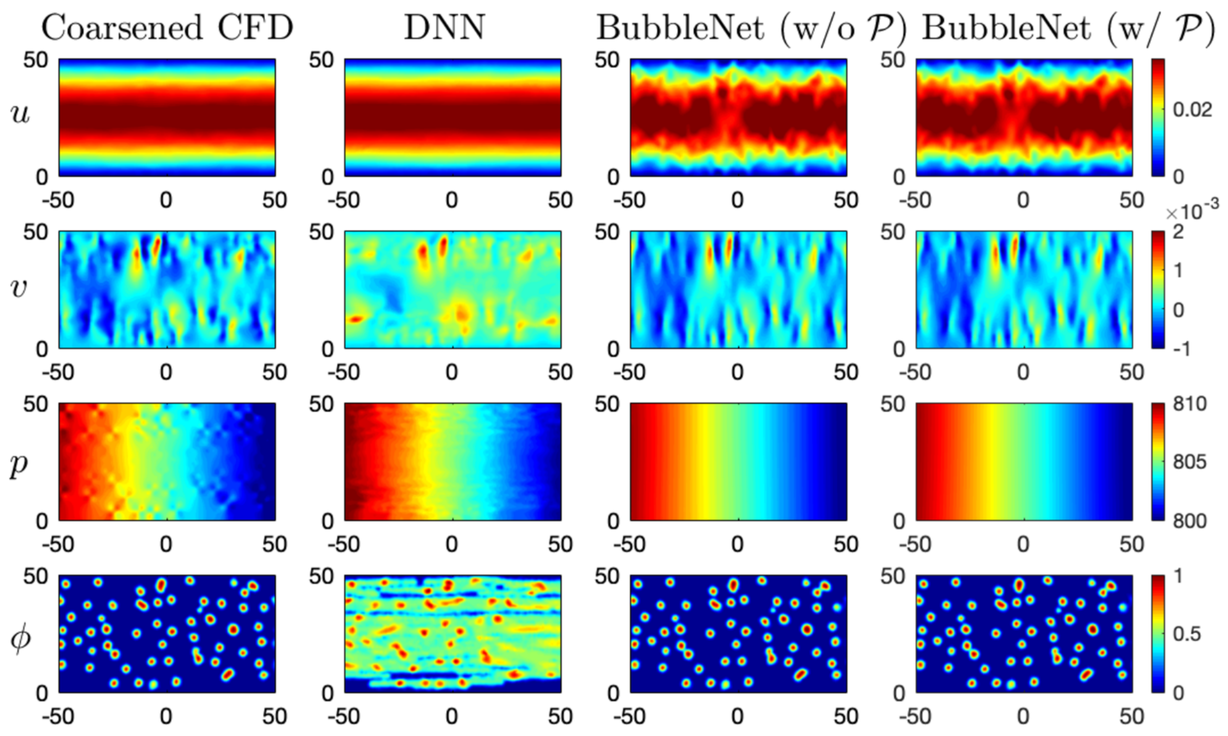

FIG. 10. The comparison between the predicted physical quantities u, v, p, ϕ from the CFD simulation results (coarsened), DNN, and `BubbleNet` for the multiple bubble flow. The four rows indicate the results for velocity u, v, p, ϕ, respectively. The first column indicates the coarsened training data and the CFD simulation results. The other columns illustrate physical quantities from the traditional DNN and `BubbleNet`(s).

which reveals the advantages of the adoptions of TDN and physical equations in the inner supervision. Consideration of the pressure Poisson equation for supervision in the `BubbleNet` does not lead to significant improvement in the results, which can be observed in the third and fourth columns in Fig. 10.

Quantitative error analyses for the multiple bubble flow are provided in Table III, which indicates that the inclusion of the physical equations in `BubbleNet`(s) is beneficial to improve the accuracy of prediction of the vertical velocity, pressure, and level set function. However, the error in the estimation of the horizontal velocity increases due to the reason we mentioned above. Moreover, different from the single bubble case, there are only negligible differences between the two `BubbleNet`(s). This implies that to get results that meet engineering needs, we may only need to consider part of the physical information in the neural network instead of full Navier–Stokes equations. This will profoundly reduce the computational resources requested for neural network training. Further quantitative analysis of the effect of the quantities of required physics information for the training of PINNs might be a promising topic.

The level set function is crucial to identify the structures and dynamics of bubbles. It is impressive that `BubbleNet`(s) present remarkable accuracy in the prediction of ϕ for the complex flow, in which $|\epsilon_\phi|$ reduces by several orders after the continuous equation is imposed. It could be difficult for traditional NNs due to the large gradient on the surface and the small training dataset, especially on coarse grids. This manifests the advantage of the present algorithm in its simplicity and accuracy.

Both the DNN and `BubbleNet`(s) succeed in approximating the horizontal variation of the pressure field, and the results obtained by the `BubbleNet`(s) are more accurate, as shown in the third row in Fig. 10 and Table III. However, both algorithms miss the subtle variations in pressure field caused by the bubble(s) movement, as in Figs. 8 and 10, although the pressure Poisson equation is utilized for supervision.

To explain pressure information deprivation, the distribution of p at a targeted time is depicted in Fig. 11, where the original data of CFD (with dense meshing), the coarsened CFD data for NN training (training), DNN, and `BubbleNet`(s) predictions are plotted on the normalized space domain $\mathcal{X}$ for the two bubbly flow

TABLE III. The mean value of the absolute errors fields of the three deep learning frameworks for the multiple bubble flow case.

	$\overline{\lvert\epsilon_u\rvert}$	$\overline{\lvert\epsilon_v\rvert}$	$\overline{\lvert\epsilon_p\rvert}$	$\overline{\lvert\epsilon_\phi\rvert}$
DNN	3.0235×10^{-4}	3.8237×10^{-4}	0.5725	0.3492
`BubbleNet` without $\mathcal{P}$	0.0015	7.6516×10^{-5}	0.2525	0.0061
`BubbleNet` with $\mathcal{P}$	0.0015	7.7402×10^{-5}	0.2481	0.0075

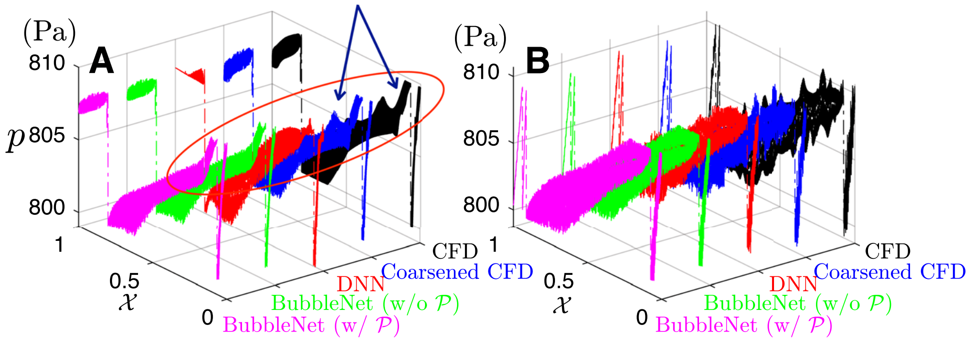

FIG. 11. The comparison of the pressure distribution for both the two bubbly flow cases. The black dotted lines illustrate the original CFD numerical results, with dense meshing. The blue dotted lines represent the coarsened CFD data used for neural network's training. The red dotted lines stand for the results of the traditional deep neural networks. The blue dotted lines and the pink dotted lines correspond to the `BubbleNet`'s predictions without and with the Poisson equation, respectively. (a) Single bubble flow. (b) Multiple bubble flow.

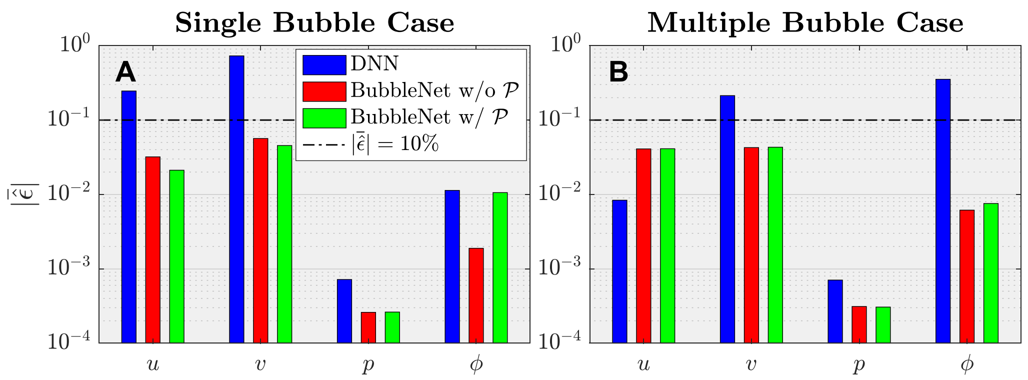

FIG. 12. The overall mean relative errors for both the single and multiple bubble flow cases [shown in (a) and (b)] predicted by the three different deep learning models (DNN and BubbleNets with and without $\mathcal{P}$), respectively. The black dashed line denotes the relative error of 10%.

cases. Here, $\mathcal{X}$ represents for the normalized form of the space field $\mathbb{X} = [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$, and n is the number of meshing elements. In the right-top corner of Fig. 11(a) (marked with the red circle), a crater-like shape (pointed out by blue arrows), which describes the structure of the single bubble, can be found for the two CFD datasets, whereas it is absent in all the predicting results. The present DNN and `BubbleNet`(s) fail to capture this subtle feature containing important physical information in pressure. A similar phenomenon is also observed in Fig. 11(b) for multiple bubble flow. The data coarsening might be partly accounted for this "feature losing," yet increasing data density will inevitably lead to a rise in computational resources. Another more promising strategy might be improving the network structure in the present study, i.e., using U-Net, GAN, ConvLSTM, etc., which have been successfully applied to turbulence investigations.[63]

To specify our objective for providing the engineering-oriented predicting models, the overall relative errors of the two bubble flow cases are analyzed, which is defined as

$$|\bar{\epsilon}| = \frac{1}{m}\sum_{i=1}^{m}\left|\frac{\mathcal{U}_{pred} - \mathcal{U}}{\mathcal{U}}\right|,$$

where $\mathcal{U}$ denotes the coarsened CFD simulation results and m is the total number of extracted data points. As shown in Fig. 12, it can be found that both BubbleNet(s) predictions errors are less than 10% (the black dashed lines in Fig. 10), indicating that BubbleNet(s) can be adopted as general acceptable engineering deep learning models.

V. CONCLUSIONS

In the present study, both traditional DNN and semi-PINN framework `BubbleNet` are applied for inferring and predicting the physics information of bubbly flow. To obtain the training dataset, two-dimensional simulations by the conventional CFD software are conducted for the single and multiple bubble flows. We briefly analyze their flow fields and find that the liquid mass ratio remains nearly constant during the whole computation, which verifies the reliability of our computations.

The deep learning framework `BubbleNet` proposed in this investigation consists of the DNN with three sub-nets for predicting different physics fields (specifically ψ, p, ϕ): the semi-physics-informed part with the fluid continuum condition, the pressure Poisson equation $\mathcal{P}$ encoded, and the time discretized normalizer (TDN), which are adopted to normalize physical variables per time step before training. The purpose of constructing this architecture is to avoid high order differentiation if the full hydrodynamic equations are encoded into the loss function, which will lead to difficulty in reducing the differential error due to the large gradient near the liquid–gas interface in bubbly flows. The `BubbleNet` framework with and without $\mathcal{P}$ are considered separately to reveal how physics information works in neural networks. Considering the high resolution of flow fields can sometimes hardly be obtained, and the training dataset is intentionally coarsened through interpolation from the original CFD simulation results to reduce the computational consumption of machine learning.

The effectiveness of the `BubbleNet`(s) is demonstrated from training the coarsened data obtained for the two bubbly flow cases. The results indicate that the `BubbleNets` are of ability to predict the physics more accurately than the traditional DNNs in which the absolute errors of the physical quantities $|\epsilon|$ decrease profoundly, especially for multiple bubble flow. TDN is also helpful to improve the accuracy of the algorithm, which indicates that a proper characteristic scale is crucial in machine learning, especially for small datasets. The inclusion of the Poisson equation has a limited effect on reducing errors of machine learning.

In summary, although the deep neural network encoding full hydrodynamic equation might be more accurate in prediction, the present `BubbleNet`, which is essentially an engineering-orientated semi-PINN, has the advantages in simplicity, computational efficiency, and flexibility. This raises an intriguing question that deserves to be pursued in the future, namely, that we can optimize the network performance by selectively introducing physical information into the neural network.

ACKNOWLEDGMENTS

This research was supported by the Natural Science Foundation of China (Grant Nos. 11772183, 11832017, and 12172209) and the Key Research Project of Zhejiang Laboratory (Grant No. 2021PE0AC02).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

DATA AVAILABILITY

The data that support the findings of this study are openly available in Zenodo at https://doi.org/10.5281/zenodo.4769081.[64]

ALGORITHM 1. DNN for predicting bubble dynamics.

```
1: class DeepNeuralNet($self, x, y, t, u, v, p, \phi, layers$)
2:     $(\hat{x},\ \hat{y},\ \hat{t},\ \hat{u},\ \hat{v},\ \hat{p},\ \hat{\phi}) = \text{UPDATE}(x, y, t, u, v, p, \phi)$
3:     $(\hat{weights},\ \hat{biases},\ \hat{layers}) = self.\text{INITIALIZENN}(weights, biases, layers)$
4:     $self.\text{Loss} = \texttt{MSE}[(u - u_{pred}) + (v - v_{pred}) + (p - p_{pred}) + (\phi - \phi_{pred})]$
5:     $u_{pred} = self.\text{Net}_{\text{u}}(x, y, t)$
6:     $v_{pred} = self.\text{Net}_{\text{v}}(x, y, t)$
7:     $p_{pred} = self.\text{Net}_{\text{p}}(x, y, t)$
8:     $\phi_{pred} = self.\text{Net}_{\phi}(x, y, t)$
9:     Optimization method 'L-BFGS-B' and optimizer: Adam
10:    def INITIALIZENN($self, layers$)
11:        Initialize all the $weights\ and\ biases$ for $\text{Net}_{\text{u}}, \text{Net}_{\text{v}}, \text{Net}_{\text{p}}, \text{Net}_{\phi}$.
12:    def NEURALNET($self, weights, biases$)
13:        Build NN for $u, v, p, \phi$ with four sets of $weights\ and\ biases$.
14:    def $\{\text{Net}_{\text{u}}, \text{Net}_{\text{v}}, \text{Net}, \text{Net}_{\phi}\}$ ($self, x, y, t$)
15:        $\{u, v, p, \phi\} = self.\text{NEURALNET}(x, y, t, weights, biases)$
16:    def TRAIN($self, iterations$)
17:        Obtain training time and Losses; train the NN with Adam optimizer.
18:    def PREDICT $\{u, v, p, \phi\}$ ($self, iterations$)
19:        $\{u_{pred}, v_{pred}, p_{pred}, \phi_{pred}\} = self.\texttt{sess}.\text{run}(x, y, t)$
20: Input = $\{x, y, t\}$, output = $\{u, v, p, \phi\}$
21: Hidden layers = [30 neurons × nine layers]
22: Load fields data of micro-bubble system dynamics simulation.
23: Set training sets = $\{x_{train}, y_{train}, t_{train}, u_{train}, v_{train}, p_{train}, \phi_{train}, layers\}$ = MaxMinScaler(Simulation data)
24: model = DEEPNEURALNET(training sets)
25: model.TRAIN($\#Iterations$)
26: Set target prediction time as $t_{pred}$
27: Obtain $\{u_{pred}, v_{pred}, p_{pred}, \phi_{pred}\}$ = model.PREDICT$(x, y, t)$ at $t_{pred}$.
28: Save all the data and post-processing.
```

ALGORITHM 2. BubbleNet: Semi-PINN for microbubble dynamics.

```
1:  class BubbleNet($self, x, y, t, u, v, p, \phi, layers$)
2:      $(\hat{x},\ \hat{y},\ \hat{t},\ \hat{u},\ \hat{v},\ \hat{p},\ \hat{\phi})$ = UPDATE($x, y, t, u, v, p, \phi$)
3:      $(\widehat{weights},\ \widehat{biases},\ \widehat{layers})$ = $self$.INITIALIZENN($weights, biases, layers$)
4:      $self$.Loss = MSE$[(u - u_{pred}) + (v - v_{pred}) + (p - p_{pred}) + (\phi - \phi_{pred})]$ + MSE$[Poisson]$
5:      $\{u_{pred}, v_{pred}, p_{pred}, \phi_{pred}\} = self.\backslash\{\mathrm{Net}_\psi, \mathrm{Net}_\mathrm{p}, \mathrm{Net}_\phi\backslash\}(x, y, t)$
6:      Optimization method 'L-BFGS-B' & optimizer: Adam
7:      def INITIALIZENN($self, layers$)
8:          Initialize all the $weights$ & $biases$ for $\mathrm{Net}_\psi$, $\mathrm{Net}_\mathrm{p}$, $\mathrm{Net}_\phi$.
9:      def NEURALNET($self, weights, biases$)
10:         Build NN for $\psi, p, \phi$ with four sets of $weights\ and\ biases$.
11:     def $\{\mathrm{Net}_\psi, \mathrm{Net}_\mathrm{p}, \mathrm{Net}_\phi\}$ ($self, x, y, t$)
12:         $\{\psi, p, \phi\}$ = $self$.NEURALNET($x, y, t, weights, biases$)
13:         $u = \partial_y\psi$ & $v = -\partial_x\psi$, $Poisson = \partial_{xx}p + \partial_{yy}p$
14:     def TRAIN($self, iterations$)
15:         Obtain training time and Losses; train the NN with Adam optimizer.
16:     def PREDICT $\{u, v, p, \phi\}$ ($self, iterations$)
17:         $\{u_{pred}, v_{pred}, p_{pred}, \phi_{pred}\}$ = $self$. sess.run($x, y, t$)
18: Set training sets = $\{x_{train}, y_{train}, t_{train}, u_{train}, v_{train}, p_{train}, \phi_{train}, layers\}$ = TimeDiscretizedNormalization(Simulation data, $time\ step$)
19: model = BUBBLENET(training sets)
20: model.TRAIN($\#Iterations$)
21: Rest procedures same as Algorithm 1
```

APPENDIX: ALGORITHMS FOR DNN AND PHYSICS-INFORMED NEURAL NETWORK

The algorithms for DNN and Physics-Informed Neural Network used in this paper are here attached as Algorithms 1 and 2. In Algorithm 1, we use the MaxMinScaler to represent the usual normalization method. For the single bubble case, the number of iterations is equal to 10^4, and the number of iterations 2×10^5 for the multiple bubble case. For the single bubble case, $t_{pred} = 2000\ \mu$s; for the multiple bubble case, $t_{pred} = 1500\ \mu$s. The four sub-nets $\mathrm{Net}_u, \mathrm{Net}_v, \mathrm{Net}_p, \mathrm{Net}_\phi$ are executed on four separate def functions. The codes are run on `tensorflow 1.15.0`.

REFERENCES

[1]T. Mitchell, *Machine Learning* (McGraw-Hill, New York, 1997), ISBN: 0-07-042807-7.
[2]D. Silver, A. Huang, C. J. Maddison *et al.*, "Mastering the game of Go with deep neural networks and tree search," Nature **529**, 484–489 (2016).
[3]D. Silver, J. Schrittwieser, K. Simonyan *et al.*, "Mastering the game of Go without human knowledge," Nature **550**, 354–359 (2017).
[4]A. W. Senior, R. Evans, J. Jumper *et al.*, "Improved protein structure prediction using potentials from deep learning," Nature **577**, 706–710 (2020).
[5]L. Koumakis, A. Kanterakis, E. Kartsaki *et al.*, "MinePath: Mining for phenotype differential sub-paths in molecular pathways," PLoS Comput. Biol. **12**(11), e1005187 (2016).
[6]J. Waring, C. Lindvall, and R. Umeton, "Automated machine learning: Review of the state-of-the-art and opportunities for healthcare," Artif. Intell. Med. **104**, 101822 (2020).
[7]K. Benke and G. Benke, "Artificial intelligence and big data in public health," Int. J. Environ. Res. Public Health **15**(12), 2796 (2018).
[8]T. Panch, J. Pearson-Stuttard, F. Greaves *et al.*, "Artificial intelligence: Opportunities and risks for public health," Lancet Digital Health **1**(1), E13–E14 (2019).
[9]Y. Li, K. Shang, W. Bian *et al.*, "Prediction of disease progression in patients with COVID-19 by artificial intelligence assisted lesion quantification," Sci. Rep. **10**, 22083 (2020).
[10]N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "COVID-19 epidemic analysis using machine learning and deep learning algorithms," medRxiv:2020.04.08.20057679v1 (2020).
[11]T. Jo, K. Nho, and A. J. Saykin, "Deep learning in Alzheimer's disease: Diagnostic classification and prognostic prediction using neuroimaging data," Front. Aging Neurosci. **11**(40), 220 (2019).
[12]A. Z. Woldaregay, E. Årsand, S. Walderhaug *et al.*, "Data-driven modeling and prediction of blood glucose dynamics: Machine learning applications in type 1 diabetes," Artif. Intell. Med. **98**, 109–134 (2019).
[13]S. A. Harmon, T. H. Sanford, S. Xu *et al.*, "Artificial intelligence for the detection of COVID-19 pneumonia on chest CT using multinational datasets," Nat. Commun. **11**, 4080 (2020).
[14]G. Chassagnon, M. Vakalopoulou, E. Battistella *et al.*, "AI-driven quantification, staging and outcome prediction of COVID-19 pneumonia," Med. Image Anal. **67**, 101860 (2021).
[15]A. Corbetta, V. Menkovski, R. Benzi *et al.*, "Deep learning velocity signals allow quantifying turbulence intensity," Sci. Adv. **7**, 12 (2021).
[16]K. Bieker, S. Peitz, S. L. Brunton *et al.*, "Deep model predictive flow control with limited sensor data and online learning," Theor. Comput. Fluid Dyn. **34**, 577–591 (2020).
[17]M. Edalatifar, M. B. Tavakoli, M. Ghalambaz *et al.*, "Using deep learning to learn physics of conduction heat transfer," J. Therm. Anal. Calorim. **146**, 1435 (2020).
[18]Z. Wang, C. Song, and T. Chen, "Deep learning based monitoring of furnace combustion state and measurement of heat release rate," Energy **131**, 106–112 (2017).
[19]S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of nonlinear dynamics," Proc. Natl. Acad. Sci. U. S. A. **113**(15), 3932–3937 (2016).

[20]S. H. Rudy, S. L. Brunton, J. L. Proctor *et al.*, "Data-driven discovery of partial differential equations," Sci. Adv. **3**, e1602614 (2017).
[21]Z. Li, N. Kovachki, K. Azizzadenesheli *et al.*, "Neural operator: Graph Kernel network for partial differential equations," ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations (2020).
[22]Z. Li, N. Kovachki, K. Azizzadenesheli *et al.*, "Fourier neural operator for parametric partial differential equations," International Conference on Learning Representations, 2021.
[23]C. Jiang, S. Esmaeilzadeh, K. Azizzadenesheli *et al.*, "MESHFREEFLOWNET: A physics-constrained deep continuous space-time super-resolution framework," SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, 2020, 1–15.
[24]L. Zhang, J. Han, H. Wang *et al.*, "Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics," Phys. Rev. Lett. **120**, 143001 (2018).
[25]H. Wang, L. Zhang, J. Han *et al.*, "DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics," Comput. Phys. Commun. **228**, 178–184 (2018).
[26]L. Zhang, J. Han, H. Wang *et al.*, "DeePCG: Constructing coarse-grained models via deep neural networks," J. Chem. Phys. **149**, 034101 (2018).
[27]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).
[28]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations," arXiv:1711.10561 (2019).
[29]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (Part II): Data-driven discovery of nonlinear partial differential equations," arXiv:1711.10566 (2019).
[30]L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," SIAM Review **63**(1), 208–228 (2021).
[31]L. Lu, P. Jin, and G. E. Karniadakis, "DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators," Nat Mach Intell **3**, 218–229 (2021).
[32]T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," IEEE Trans. Neural Networks **6**(4), 911–917 (1995).
[33]S. Cai, Z. Wang, L. Lu *et al.*, "DeepM & Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks," J. Comp. Phys. **436**, 110296 (2021).
[34]Z. Mao, L. Lu, O. Marxen *et al.*, "DeepM & Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators," J. Comp. Phys. **447**, 110698 (2021).
[35]A. D. Jagtap, E. Karazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," Comput. Methods Appl. Mech. Eng. **365**, 113028 (2019).
[36]A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," Commun. Comput. Phys. **28**, 2002–2041 (2020).
[37]X. Meng, Z. Li, D. Zhang *et al.*, "PPINN: Parareal physics-informed neural network for time-dependent PDEs," Comput. Methods Appl. Mech. Eng. **370**, 113250 (2020).
[38]C. Rao, H. Sun, and Y. Liu, "Physics-informed deep learning for incompressible laminar flows," Theor. Appl. Mech. Lett. **10**(3), 207–212 (2020).
[39]Z. Mao, A. D. Jagtap, G. E. Karniadakis *et al.*, "Physics-informed neural networks for high-speed flows," Comput. Methods Appl. Mech. Eng. **360**, 112789 (2020).
[40]R. Wang, K. Kashinath, M. Mustafa *et al.*, "Towards physics-informed deep learning for turbulent flow prediction," in *KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. August 2020* (ACM Digital Library, 2020), pp. 1457–1466.
[41]C. Lin, Z. Li, L. Lu *et al.*, "Operator learning for predicting multiscale bubble growth dynamics," J. Chem. Phys. **154**, 104118 (2020).
[42]L. von Rueden, S. Mayer, K. Beckh *et al.*, "Informed machine learning—A taxonomy and survey of integrating prior knowledge into learning systems," in *IEEE Transactions on Knowledge and Data Engineering* (IEEE, 2021), p. 1.
[43]H. Miao, S. M. Gracewski, and D. Dalecki, "Ultrasonic excitation of a bubble inside a deformable tube: Implications for ultrasonically induced hemorrhage," J. Acoust. Soc. Am. **124**, 2374–2384 (2008).
[44]N. Hosseinkhah, H. Chen, T. J. Matula *et al.*, "Mechanisms of microbubble–vessel interactions and induced stresses: A numerical study," J. Acoust. Soc. Am. **134**, 1875–1885 (2013).
[45]N. Hosseinkhah, D. E. Goertz, and K. Hynynen, "Microbubbles and blood brain barrier opening: A numerical study on acoustic emissions and wall stress predictions," IEEE Trans. Biomed. Eng. **62**(5), 1293–1304 (2015).
[46]S. A. Peyman, R. H. Abou-Saleh, J. R. McLaughlan *et al.*, "Expanding 3D geometry for enhanced on-chip microbubble production and single step formation of liposome modified microbubbles," Lab Chip **12**, 4544–4552 (2012).
[47]V. Papadopoulou, M.-X. Tang, C. Balestra *et al.*, "Circulatory bubble dynamics: From physical to biological aspects," Adv. Colloid Interface Sci. **206**, 239–249 (2014).
[48]B. Dollet, W. van Hoeve, J. P. Raven *et al.*, "Role of the channel geometry on the bubble pinch-off in flow-focusing devices," Phys. Rev. Lett. **100**, 034504 (2008).
[49]M. A. Herrada, J. M. Montanero, C. Ferrera *et al.*, "Analysis of the dripping–jetting transition in compound capillary jets," J. Fluid Mech. **649**, 523–536 (2010).
[50]W. van Hoeve, B. Dollet, M. Versluis *et al.*, "Microbubble formation and pinch-off scaling exponent in flow-focusing devices," Phys. Fluids **23**, 092001 (2011).
[51]E. J. Vega, A. J. Acero, J. M. Montanero *et al.*, "Production of microbubbles from axisymmetric flow focusing in the jetting regime for moderate Reynolds numbers," Phys. Rev. E **89**, 063012 (2014).
[52]B. Zhao, A. Alizadeh Pahlavan, L. Cueto-Felgueroso *et al.*, "Forced wetting transition and bubble pinch-off in a capillary tube," Phys. Rev. Lett. **120**, 084501 (2014).
[53]E. Talu, K. Hettiarachchi, R. L. Powell *et al.*, "Maintaining monodispersity in a microbubble population formed by flow-focusing," Langmuir **24**(5), 1745–1749 (2008).
[54]M. Tenjimbayashi, K. Doi, and M. Naito, "Microbubble flows in superwettable fluidic channels," RSC Adv. **9**, 21220 (2019).
[55]A. G. Baydin, B. A. Pearlmutter, A. A. Radul *et al.*, "Automatic differentiation in machine learning: A survey," J. Mach. Learn. Res. **8**, 1–43 (2018).
[56]MEMS Module User's Guide, COMSOL Multiphysics® v. 5.6. COMSOL AB, Stockholm, Sweden, 2020, pp. 203–211.
[57]W. F. Ganong, *Review of Medical Physiology*, 22nd ed. (McGraw-Hill, New York, 2005), pp. 592–593 and 702.
[58]A. C. Guyton and J. E. Hall, *Human Physiology and Mechanisms of Disease*, 6th ed. (Saunders, Philadelphia, PA, 1997), pp. 115, 116, 131–141, 319–322, and 521–523.
[59]A. C. Guyton, H. J. Granger, and A. E. Taylor, "Interstitial fluid pressure," Physiol. Rev. **51**(3), 527–563 (1971).
[60]G. Tomaiuolo, M. Simeone, V. Martinelli *et al.*, "Red blood cell deformation in microconfined flow," Soft Matter **5**, 3736–3740 (2009).
[61]X. Li, W. Wang, P. Zhang *et al.*, "Interactions between gas—Liquid mass transfer and bubble behaviours," R. Soc. Open Sci. **6**, 190136 (2019).
[62]G. E. Karniadakis, I. G. Kevrekidis, L. Lu *et al.*, "Physics-informed machine learning," Nat. Rev. Phys. **3**, 422–440 (2021).
[63]R. Wang, K. Kashinath, M. Mustafa *et al.*, "Towards physics-informed deep learning for turbulent flow prediction," OpenReview, https://openreview.net/forum?id=Hkg5lAEtvS.
[64]H. Zhai (2021). "Data for BubbleNet code & micro-bubbles system dynamics simulation (v.0.0)," Zenodo. https://doi.org/10.5281/zenodo.4769081