Journal of Computational Physics 496 (2024) 112589

Contents lists available at ScienceDirect

Journal of Computational Physics

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

A deep learning method for the dynamics of classic and conservative Allen-Cahn equations based on fully-discrete operators

Yuwei Geng, Yuankai Teng, Zhu Wang, Lili Ju *

Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA

ARTICLE INFO

Keywords:
Allen-Cahn equations
Mass-conserving
Fully-discrete operator
Convolutional neural network
Bound limiter
Training strategy

ABSTRACT

The Allen-Cahn equation is a well-known stiff semilinear parabolic equation used to describe the process of phase separation and transition in phase field modeling of multi-component physical systems, while the conservative Allen-Cahn equation is a modified version of the classic Allen-Cahn equation that can additionally conserve the mass. As neural networks and deep learning techniques have achieved significant successes in recent years in scientific and engineering applications, there has been growing interest in developing deep learning algorithms for numerical solutions of partial differential equations. In this paper, we propose a novel deep learning method for predicting the dynamics of the classic and conservative Allen-Cahn equations. Specifically, we design two special convolutional neural network models, one for each of the two equations, to learn the fully-discrete operators between two adjacent time steps. The loss functions of the two models are defined using the residual of the fully-discrete systems, which result from applying the central finite difference discretization in space and the Crank–Nicolson approximation in time. This approach enables us to train the models without requiring any ground-truth data. Moreover, we introduce an effective training strategy that automatically generates useful samples along the time evolution to facilitate training of the models. Finally, we conduct extensive experiments in two and three dimensions to demonstrate outstanding performance of our proposed method, including its dynamics prediction and generalization ability under different scenarios.

1. Introduction

The Allen-Cahn (AC) equation was first introduced in [1] as a phenomenological model for antiphase domain coarsening in a binary alloy. Since then it has been used to describe a wide range of phase transition phenomena in science and engineering applications, including crystal growth [3,31], image analysis [2,11,4], and biology [22,33]. The classic AC equation takes the following form:

$$\partial_t u = \epsilon^2 \Delta u + f(u), \quad \boldsymbol{x} \in \Omega, t > 0 \tag{1}$$

with the initial condition

* Corresponding author.
E-mail addresses: ygeng@email.sc.edu (Y. Geng), yteng@email.sc.edu (Y. Teng), wangzhu@math.sc.edu (Z. Wang), ju@math.sc.edu (L. Ju).

https://doi.org/10.1016/j.jcp.2023.112589
Received 17 May 2023; Received in revised form 26 September 2023; Accepted 19 October 2023

Available online 28 October 2023
0021-9991/

$$u(\boldsymbol{x}, t=0) = u_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega,$$

and appropriate boundary conditions such as the periodic or homogeneous boundary condition. Here $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$ is a bounded domain with the Lipschitz boundary $\partial\Omega$, $\epsilon > 0$ is an interfacial parameter, $u(\boldsymbol{x}, t)$ is the unknown scalar function, and $f = -F'$, where $F(u)$ is an nonlinear potential function. There are two commonly used potential functions as considered in this paper: one is the double-well potential in which

$$F(u) = \frac{1}{4}(u^2 - 1)^2, \quad f(u) = u - u^3, \tag{2}$$

and the other is the Flory-Huggins potential in which

$$F(u) = \frac{\theta}{2}[(1+u)\ln(1+u) + (1-u)\ln(1-u)] - \frac{\theta_c}{2}u^2, \quad f(u) = \frac{\theta}{2}\ln\frac{1-u}{1+u} + \theta_c u \tag{3}$$

with the constant parameters $0 < \theta < \theta_c$.

Mass-conserving is often necessary in many physical systems, however, the classic AC equation (1) doesn't maintain the total mass $M(t) = \int_\Omega u(\boldsymbol{x}, t)d\boldsymbol{x}$ as time evolves. The *conservative* Allen-Cahn equations are some variants of the classic Allen-Cahn equation that can additionally conserve the mass. A typical one was derived and analyzed in [21] that takes the following form:

$$\partial_t u = \epsilon^2 \Delta u + f(u) - \frac{1}{|\Omega|}\int_\Omega f(u(\boldsymbol{y}, t))d\boldsymbol{y}, \quad \boldsymbol{x} \in \Omega, t > 0, \tag{4}$$

where $\frac{1}{|\Omega|}\int_\Omega f(u(\boldsymbol{y}, t))d\boldsymbol{y}$ is a nonlocal Lagrange multiplier introduced to conserve the total mass over time, i.e., $M(t) = M(0)$ for any $t \geq 0$. Both the classic AC equation (1) and the conservative AC equation (4) are *autonomous* systems under the periodic or homogeneous boundary condition.

The classic AC equation (1) can be regarded as the L_2 gradient flow with respect to the energy functional:

$$E(u) = \int_\Omega \left(\frac{\epsilon^2}{2}|\nabla u(\boldsymbol{x}, t)|^2 + F(u(\boldsymbol{x}, t))\right) d\boldsymbol{x}. \tag{5}$$

The conservative AC equation (4) is also energy dissipative under the same energy functional (5). In addition, the maximum bound principle (MBP) [6,16] is another important property of the classic and conservative Allen-Cahn equations. For the double-well potential case (2), the solution of the classic AC equation (1) always stays inside the interval $[-1, 1]$ and that of the conservative AC equation (4) is within $[-\frac{2}{3}\sqrt{3}, \frac{2}{3}\sqrt{3}]$. For the Flory-Huggins potential case (3), the solution of the classic AC equation is bounded in the interval $[-\rho_1, \rho_1]$ for all the time, where $1 > \rho_1 > 0$ is the constant satisfying $f(\rho_1) = 0$, and that of the conservative AC equation is within $[-\rho_2, \rho_2]$, where $1 > \rho_2 > 0$ is the constant satisfying $f(\rho_2) = f(-\sqrt{1 - \theta/\theta_c})$.

Analytical solutions to the classic and conservative AC equations are very challenging or even impossible to find. Hence, effective and accurate numerical methods for computing their approximate solutions become highly necessary in order to simulate and understand their dynamics. Spatial discretization methods, such as the finite element method (FEM) [8,34,12] and the finite difference method (FDM) [13,14,39] and spectral methods [10], have been investigated extensively for discretizing these two equations. It is required that the computational mesh size should be less than the interfacial parameter ϵ in order to accurately describe the transition layers and capture the correct dynamics. Consequently, when the interfacial parameter ϵ gets smaller, the computational cost gets larger. Meanwhile, many different time-stepping algorithms, such as fully implicit schemes [7,36], stabilized semi-implicit schemes [26,27], convex splitting schemes [23,32,28], exponential integrator schemes [5,6,10], invariant energy quadratization (IEQ) [37,38], and scalar auxiliary variable (SAV) [24,25] have been designed for the AC equation, and many of them can ensure the energy stability and the MBP in the discrete sense. Generally speaking, explicit schemes require a restrictive time step size due to the CFL condition which can be very time-consuming when the interface parameter ϵ is small. Implicit or semi-implicit schemes can avoid the CFL condition restriction and thus allow for relatively larger time step sizes, but the computational costs still could be high due to the use of linear solvers and/or nonlinear iterations at each time step.

In recent years, deep learning-based methods, in addition to conventional numerical methods, have been introduced to solve the above classic and conservative AC equations, as well as some other phase field models, by utilizing both data and physical knowledge. One such method is the well-known Physics-Informed Neural Network (PINN) [20], which approximates the mapping between the solution and the spatiotemporal input while complying with any specific physical principle defined by general nonlinear partial differential equations. To accelerate the training of PINN, Wight and Zhao proposed two time-adaptive approaches in [40]: one involves training every small interval separately after splitting the time domain into small intervals, while the other involves starting training with small time intervals and gradually increasing the time span when each span is learned well. Furthermore, the backward compatible PINN (bc-PINN) [19] was introduced to split successive time into small time intervals, adding backward compatibility to the physical law during training of each time interval to ensure that the neural network approximation is effective in all previous time segments. In [29], a modified PINN was proposed for solving the one-dimensional conservative AC equation (4) by adding a term representing the difference of input and output masses into the loss function. This approach can be regarded as a soft constraint implementation for mass conservation in PINN, but it was only applicable to the one-dimensional case due to its special construction process. On the other hand, there also exist some limitations in the above PINN-based learning approaches. The first one is the high computational cost due to the need of retraining the network as the initial conditions change. This is because their loss functions involve initial condition data (Note that the periodic or homogeneous boundary conditions do not impose any

extra nontrivial data). Second, the PINN-based learning methods usually couldn't produce numerical solutions with high accuracy even when lots of neurons with enough representation ability are used in the network. This is mostly due to the difficulty in effective training. Third, they often lack good generalization ability, which becomes critical for predicting solutions of time-dependent PDEs. Finally, it is still a hard problem for the PINN-based methods to effectively handle the nonlocal Lagrange multiplier term in the conservative AC equation in two or three dimensions. There also exist some operator-based learning methods for PDEs and surrogate models. For example, DeepONet [18] and its variant [30] extend the universal approximation theorem to deep neural networks to learn the operators, and the physics-informed neural operator (PINO) [17] combines operator learning with function approximation to achieve better accuracy. These methods can deal with various initial/boundary conditions, but usually are very hard to train with good accuracy for large scale problems.

In this paper, we propose a novel deep learning method for computing the dynamics of the classic and conservative AC equations. Our method uses two specially designed convolutional neural networks (CNNs) in an end-to-end fashion (one for each AC equation) to learn the fully-discrete operators with second-order approximation accuracy between two adjacent time steps for the target equations, formulated by using the central finite difference discretization in space and the Crank–Nicolson approximation in time. Particularly, the loss function for training our CNN models is defined through the residual of the fully-discrete system instead of the original PDE-based physical law. Consequently, the learned fully-discrete operators then can be repeatedly used to predict the solutions of the AC equations along time subject to various initial conditions, which is a major difference from PINN-based learning methods. Since the initial data could be of very high dimension (the functional value at each mesh node gives one degree of freedom), a key question is how to design an effective training strategy for the proposed models with very few initial condition data so that they still can learn and predict accurately the solutions of the AC equations for any given unseen initial condition. For this purpose, we develop a novel training strategy that can automatically generate useful samples along the time evolution to facilitate training of the models. In order to preserve the MBP and avoid the potential overflow errors during the training in the Flory-Huggins potential case (due to the existence of logarithmic terms), we also integrate a bound limiter module into the neural networks. Additionally, unlike the soft constraint approach used in [35], the mass-conservation property is treated as a hard constraint in the proposed CNN model for the conservative AC equation (4). This, to the best of our knowledge, is a novel contribution, which guarantees the exact conservation of total mass and plays a key role in accurately learning the dynamics of the conservative AC equation.

The rest of the paper is organized as follows. In Section 2, we introduce the space and time discretization for the classic and conservative AC equations and review the concept of CNNs. Then we propose and discuss our deep learning method for simulating the dynamics of the two AC equations, including network architectures, loss functions, and training strategies. Extensive numerical experiments and comparisons are presented in Section 3 to demonstrate the outstanding performance of the proposed method, including some ablation studies and its prediction and generalization ability under different scenarios in two and three dimensions, respectively. Finally, some concluding remarks are drawn in Section 4 together with a brief discussion on future work.

2. The proposed deep learning method

In this section we first introduce the fully-discrete systems for the classic and conservative AC equations, (1) and (4), and then propose a deep learning method based on the fully-discrete operators and end-to-end CNNs for predicting the dynamics of these two equations. The network architectures, loss functions, and training strategies will be discussed. For ease of illustration, we take Ω to be a square domain $[0, L]^2$ in two dimensions (2D) and assume that u satisfies the periodic boundary condition, although the method can be easily extended to three dimensional (3D) problems and applied to different boundary conditions.

2.1. Fully-discrete systems of the classic and conservative Allen-Cahn equations

When solving the two AC equations using conventional numerical methods, it is well known that explicit schemes are limited by the CFL condition, which necessitates a restrictive time step size. Therefore, semi-implicit and fully implicit time stepping methods are commonly used in practice. Let us uniformly partition the domain Ω and obtain the set of grid points $\{(x_i, y_j)\}_{i,j=0}^{N}$ where $N > 0$ is an integer, $x_i = ih$ and $y_j = jh$ with $h = L/N$. Then we choose a uniform time step size $\Delta t > 0$ and set $t_n = n\Delta t$ for $n \geq 0$. Correspondingly, we set $U_0^{i,j} = u_0(x_i, y_j)$ and assume $U_n^{i,j} \approx u(x_i, y_j, t_n)$. The periodic boundary condition implies $U_n^{0,j} = U_n^{N,j}$ and $U_n^{j,0} = U_n^{j,N}$ for $j = 0, 1, \cdots N$, thus the set of unknown can be represented by $U_n = \{U_n^{i,j}\}_{i,j=1}^{N}$.

After applying the central finite difference discretization in space and the Crank–Nicolson approximation in time, we obtain a fully-discrete system for the classic AC equation (1) with the periodic boundary condition as follows: given U_0, to find U_{n+1} for $n \geq 0$ such that

$$\frac{U_{n+1}^{i,j} - U_n^{i,j}}{\Delta t} = \epsilon^2 \frac{\Delta_h U_{n+1}^{i,j} + \Delta_h U_n^{i,j}}{2} + \frac{f(U_{n+1}^{i,j}) + f(U_n^{i,j})}{2},$$

for $i, j = 1, 2, \cdots, N$, where Δ_h denotes the five-point stencil for the Laplacian as

$$\Delta_h U_{n+1}^{i,j} = \frac{U_{n+1}^{i+1,j} + U_{n+1}^{i-1,j} + U_{n+1}^{i,j+1} + U_{n+1}^{i,j-1} - 4U_{n+1}^{i,j}}{h^2}.$$

This yields

$$U_{n+1}^{i,j} - \frac{\Delta t}{2}\left(\epsilon^2 \Delta_h U_{n+1}^{i,j} + f(U_{n+1}^{i,j})\right) = U_n^{i,j} + \frac{\Delta t}{2}\left(\epsilon^2 \Delta_h U_n^{i,j} + f(U_n^{i,j})\right), \tag{6}$$

for $i, j = 1, 2, \cdots, N$. It is known that the system (6) is guaranteed to have a unique solution U_{n+1} for any given U_n under a quite relaxed restriction on the time step size Δt and the resulting numerical solution is of second order accuracy in both time and space. The system can be rewritten as

$$U_{n+1} = \mathrm{FDCN}_{AC}(U_n), \quad n = 0, 1, \cdots, \tag{7}$$

where FDCN_{AC} denotes the fully discrete operator that maps the input U_n to the output U_{n+1} through the relation (6).

Similarly, we can obtain a fully-discrete system with second-order accuracy for the conservative AC equation (4) with the periodic boundary condition as follows: given U_0, find U_{n+1} for $n \geq 0$ such that

$$\frac{U_{n+1}^{i,j} - U_n^{i,j}}{\Delta t} = \epsilon^2 \frac{\Delta_h U_{n+1}^{i,j} + \Delta_h U_n^{i,j}}{2} + \frac{f(U_{n+1}^{i,j}) + f(U_n^{i,j})}{2} - \frac{1}{N^2} \sum_{i',j'=1}^{N} \frac{f(U_{n+1}^{i',j'}) + f(U_n^{i',j'})}{2},$$

for $i, j = 1, 2, \cdots, N$. It leads to

$$U_{n+1}^{i,j} - \frac{\Delta t}{2}\left(\epsilon^2 \Delta_h U_{n+1}^{i,j} + f(U_{n+1}^{i,j}) - \frac{1}{N^2} \sum_{i',j'=1}^{N} f(U_{n+1}^{i',j'})\right) = U_n^{i,j} + \frac{\Delta t}{2}\left(\epsilon^2 \Delta_h U_n^{i,j} + f(U_n^{i,j}) - \frac{1}{N^2} \sum_{i',j'=1}^{N} f(U_n^{i',j'})\right), \tag{8}$$

for $i, j = 1, 2, \cdots, N$, which can be rewritten as

$$U_{n+1} = \mathrm{FDCN}_{mAC}(U_n), \quad n = 0, 1, \cdots, \tag{9}$$

where FDCN_{mAC} denotes the fully discrete operator that maps the input U_n to the output U_{n+1} through the relation (8).

For conventional numerical methods, the systems (6) and (8) need to be solved using some nonlinear iterative solvers, such as the Picard iteration or the Newton's method, and the computational cost can become high for large-scale problems. For example, the Picard iteration solves (6) iteratively as follows: let $U_{n+1,(0)} = U_n$ and for $m = 0, 1, \cdots$, compute $U_{n+1,(m+1)}$ by solving the linear system

$$\left(I - \frac{\epsilon^2 \Delta t}{2} \Delta_h\right) U_{n+1,(m+1)}^{i,j} = U_n^{i,j} + \frac{\Delta t}{2}\left(\epsilon^2 \Delta_h U_n^{i,j} + f(U_n^{i,j}) + f(U_{n+1,(m)}^{i,j})\right) \tag{10}$$

for $i, j = 1, 2, \cdots, N$, and finally set $U_{n+1} = U_{n+1,(m^*)}$ if $U_{n+1,(m^*)}$ satisfies certain convergence criteria. When solving (8), one replaces (10) by

$$\left(I - \frac{\epsilon^2 \Delta t}{2} \Delta_h\right) U_{n+1,(m+1)}^{i,j} = U_n^{i,j} + \frac{\Delta t}{2}\left(\epsilon^2 \Delta_h U_n^{i,j} + f(U_n^{i,j}) + f(U_{n+1,(m)}^{i,j}) - \frac{1}{N^2} \sum_{i',j'=1}^{N} (f(U_n^{i',j'}) + f(U_{n+1,(m)}^{i',j'}))\right) \tag{11}$$

for $i, j = 1, 2, \cdots, N$. Note that FFT-based fast implementations are available [6] for solving (10) and (11) on uniform rectangular grids, but such Picard iterations converge linearly. On the other hand, if Newton's method is applied, the iterations would converge quadratically, but FFT-based efficient implementations don't exist anymore.

To efficiently solve the classic and conservative AC equations, we next develop a deep learning method that learns the two fully-discrete operators FDCN_{AC} and FDCN_{mAC} without utilizing any ground-truth data for training. The goal is to achieve accurate approximation of these operators, enabling the fast prediction of the dynamics of the AC equations under arbitrary initial conditions without the need of retraining.

2.2. Network architectures

Next we will construct two CNNs in an end-to-end fashion to model the nonlinear mapping from U_n to U_{n+1} with the fixed time step size Δt: one is called "ACNN" for the classic AC equation (1) and its full discretization (6), the other is referred to as "mACNN" for the conservative AC equation (4) and its full discretization (8).

Network architecture for the classic Allen-Cahn equation Fig. 1 illustrates the proposed "ACNN" for learning the dynamics of the classic AC equation (1), which consists of one single convolutional layer, one or more successive Residual blocks (ResBlocks) [9], and one final convolutional layer. Note that there is no pooling layer or fully-connected layer in the ACNN. We take U_n, an $N \times N \times 1$ tensor, as the input and predict the solution at next time step, U_{n+1}, by using a loss function formulated from the fully-discrete Crank–Nicolson scheme (6) (concrete expression given in (12)). The filter size in each convolutional layer is fixed as $K \times K$ and for each convolutional layer, we fix the number of different filters to be C for all intermediate layers. The first layer will broadcast the input tensor to the shape of $N \times N \times C$ while the last layer will bring it back to $N \times N \times 1$. Motivated by the periodic boundary condition of the target problem, the *"circular"* padding mode is applied across all convolutional layers. To maintain the spatial dimensions of the input feature map, the padding size is set in accordance with the filter size employed in each layer. To address the issue of vanishing gradients and promote better gradient flow during training, we incorporate ResBlocks into the ACNN architecture. Each ResBlock comprises two convolutional layers followed by an activation function σ, where the input to the ResBlock is added to the output of the second convolutional layer prior to applying the activation function, thereby mitigating the potential for vanishing gradients during back propagation. Note that the violation of the MBP may cause the overflow error in evaluating the loss function in the Flory-Huggins potential case (3) due to the presence of logarithmic terms, and consequently leads to the failure of the training.

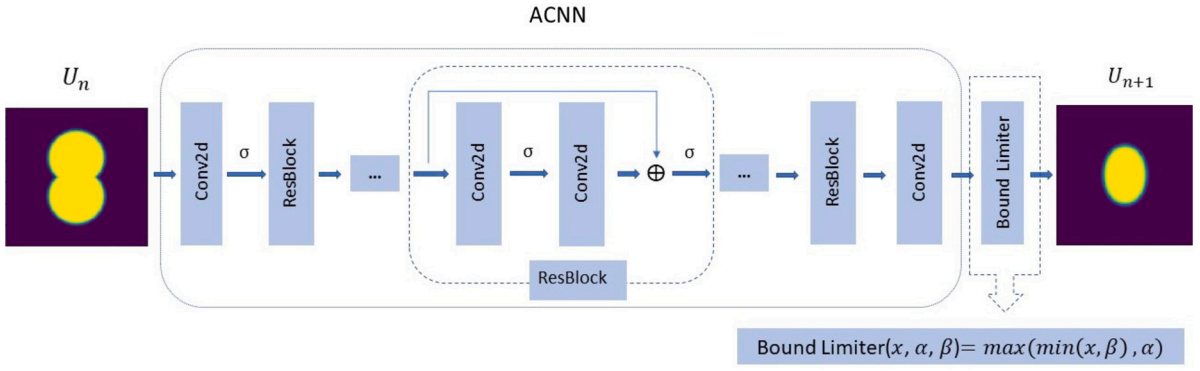


Fig. 1. The network architecture of ACNN, which learns the dynamics of the classic Allen-Cahn equations (1). In this work, it is used to approximate the operator FDCN_{AC} of the fully-discrete Crank–Nicolson scheme (6), mapping U_n to U_{n+1}, i.e., $\text{ACNN}(U_n;\Theta) \approx \text{FDCN}_{AC}(U_n)$, where Θ denotes the set of learnable parameters in ACNN.

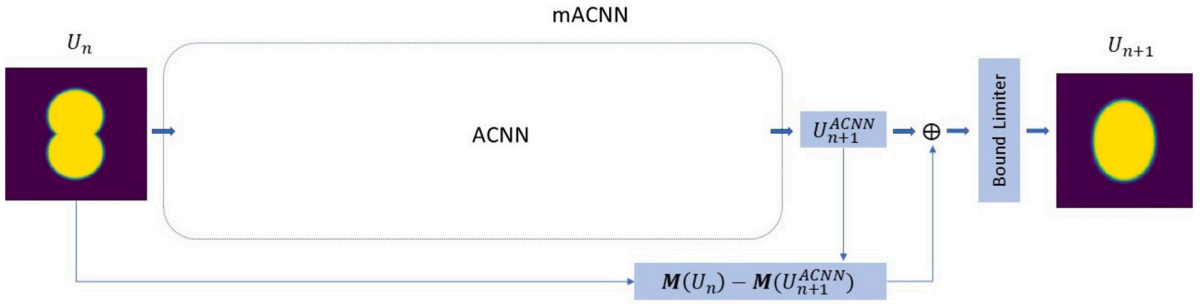


Fig. 2. The network architecture of mACNN, which learns the dynamics of the conservative Allen-Cahn equations (4). In this work, it is used to approximate the operator FDCN_{mAC} of the fully-discrete Crank–Nicolson scheme (8), mapping U_n to U_{n+1}, i.e., $\text{mACNN}(U_n;\Theta) \approx \text{FDCN}_{mAC}(U_n)$, where Θ denotes the set of learnable parameters in mACNN.

To uphold the MBP, we further add a bound limiter module into the network architecture, which ensures that the network output remains within the interval $[\alpha, \beta]$. We expect the bound limiter to play an important role at the early stage of training ACNN in preserving the MBP, but won't be triggered once ACNN is nearly optimized. Thus the parameters α and β will be set as some slightly relaxed values to the theoretical lower and upper bounds for the solution of the target AC equation.

Network architecture for the conservative Allen-Cahn equation For learning the dynamics of the mass-conserving AC equation, we leverage the ACNN as the foundational architecture and extend it to the "mACNN". The mACNN is different than the ACNN in the sense that the total mass is preserved as a hard constraint. This constraint is applied regardless of whether the model is being trained or being used for inference. This approach ensures that the solution obtained at each time step conserves mass, which is a crucial property of the mass-conservative AC equation. It would also enhance the numerical accuracy and robustness for solving the mass-conserving AC equation. The associated network architecture is displayed in Fig. 2. Given the input U_n with a total mass $M(U_n) = \frac{L^2}{N^2}\sum_{i,j=1}^{N} U_n^{i,j}$, the ACNN module generates an intermediate result, denoted by U_{n+1}^{ACNN}. This could result in a difference in mass:

$$\Delta M = M(U_n) - M(U_{n+1}^{ACNN}).$$

To enforce the mass preservation, a value of $\frac{\Delta M}{L^2}$ will be added to every entry of U_{n+1}^{ACNN}, which provides, after further applying the bound limiter, the final output of mACNN U_{n+1}. Note that although the bound limiter could affect the mass conservation at the early stage of training mACNN, but again it won't be triggered once mACNN is nearly optimized and thus the total mass is still well conserved in the end (as demonstrated by numerical experiments).

2.3. Loss functions

The goal of ACNN and mACNN is to learn the two fully-discrete operators, FDCN_{AC} and FDCN_{mAC}, respectively, that are applicable to various inputs U_n, including different initial conditions and any intermediate states. To this end, we consider for the training process an ensemble of classic AC or conservative AC equations consisting of S distinct problems that differ only in their initial data. Denote the network input by $\mathbf{U}_n = \{U_n^{(1)}, U_n^{(2)}, \ldots, U_n^{(S)}\}$, where $U_n^{(l)}$ represents the solution at the time t_n corresponding to the l^{th} problem. To learn the operator between $\mathbf{U}_n$ and $\mathbf{U}_{n+1}$, we define the loss function based on the fully-discrete scheme (6) for the

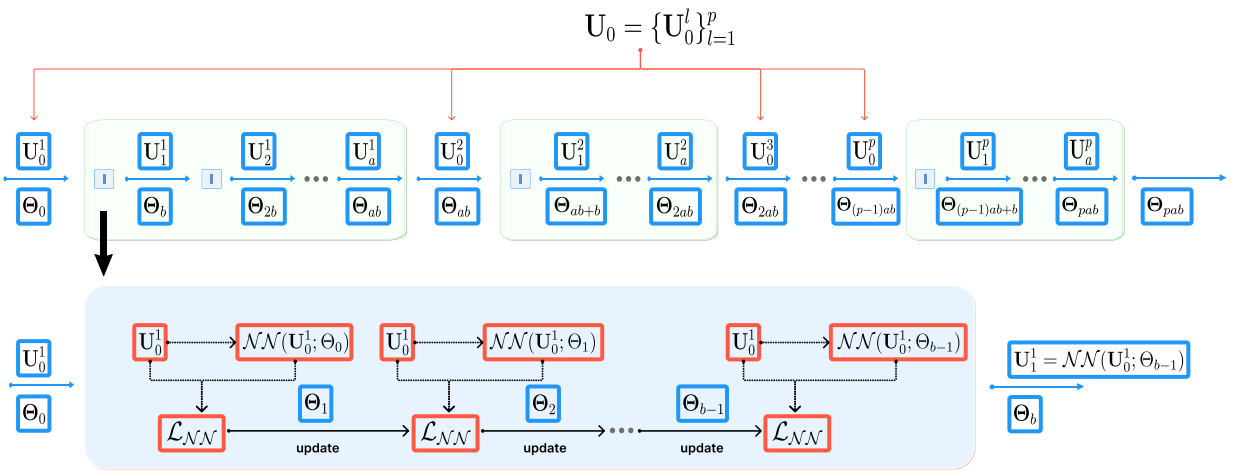


Fig. 3. Illustration of the training process for one epoch, where $\mathcal{NN}$ represents the proposed neural network, ACNN or mACNN.

classic AC equation and the fully-discrete scheme (8) for the conservative AC equation respectively. Specifically, the loss function is formulated to measure the violation of (6) or (8) in the L^2-norm:

$$\mathcal{L}_{ACNN} = \frac{1}{SN^2}\sum_{l=1}^{S}\sum_{i,j=1}^{N}\left[U_{n+1}^{i,j,(l)} - U_n^{i,j,(l)} - \frac{\Delta t}{2}\left(\epsilon^2\Delta_h(U_{n+1}^{i,j,(l)} + U_n^{i,j,(l)}) + f(U_{n+1}^{i,j,(l)}) + f(U_n^{i,j,(l)})\right)\right]^2 \tag{12}$$

for ACNN and

$$\begin{aligned}\mathcal{L}_{mACNN} = \frac{1}{SN^2}\sum_{l=1}^{S}\sum_{i,j=1}^{N}\Bigg[U_{n+1}^{i,j,(l)} - U_n^{i,j,(l)} - \frac{\Delta t}{2}\Bigg(&\epsilon^2\Delta_h(U_{n+1}^{i,j,(l)} + U_n^{i,j,(l)}) + f(U_{n+1}^{i,j,(l)}) + f(U_n^{i,j,(l)}) \\ &- \frac{1}{N^2}\sum_{i',j'=1}^{N}(f(U_{n+1}^{i',j',(l)}) + f(U_n^{i',j',(l)}))\Bigg)\Bigg]^2\end{aligned} \tag{13}$$

for mACNN, such that $U_{n+1}^{i,j,(l)} = \mathrm{ACNN}(U_n^{i,j,(l)};\Theta)$ in (12) and $U_{n+1}^{i,j,(l)} = \mathrm{mACNN}(U_n^{i,j,(l)};\Theta)$ in (13), where $\{U_{n+1}^{i,j,(l)}\}$ denotes the approximate solution of the l^{th} problem. It is worth noting that with the above choices of loss functions, both ACNN and mACNN do not need any ground truth data for training.

2.4. The training strategy

Next, we describe the strategy for training ACNN and mACNN. Suppose there are totally S problems with different initial conditions and $S = pq$. The initial data is assembled in $\mathbf{U}_0 = \{U_0^{(l)}\}_{l=1}^{S}$, each value of $\{U_0^{i,j,(l)}\}_{i,j=1}^{N}$ is bounded between $(-1,1)$. The training time interval is taken as $[0, T_{train}]$ with $T_{train} = a\Delta t$ and a a positive integer.

The training process involves updating learnable parameters of ACNN and mACNN over epochs. Fig. 3 presents the detailed training flow within one epoch. At the beginning of the epoch, the target neural network (ACNN or mACNN) is fed with the same initial data $\mathbf{U}_0$ (corresponding to the initial time $t = 0$). Assume the target neural network before the current epoch is parameterized by Θ. It will be repeatedly updated using self-generated data. The following optimization is performed within the epoch:

(i). Randomly split $\mathbf{U}_0 = \{U_0^{(1)}, U_0^{(2)}, \ldots, U_0^{(S)}\}$ into p subsets $\{\mathbf{U}_0^l\}_{l=1}^{p}$ and each subset contains q distinct initial conditions (since $S = pq$). Set $\Theta_0 = \Theta$.

(ii). For the first time step (from $t = 0$ to Δt), take the initial condition subset $\mathbf{U}_0^1$ as the input of the neural network with parameter Θ_0 and generate $\mathbf{U}_1^1$. To make sure the output $\mathbf{U}_1^1$ is a relatively accurate approximation to the true solution, at this step we train the neural network with respect to the loss function (12) or (13) for b times ($b > 0$ is an integer), and consequently update the parameters to Θ_b.

(iii). The same training approach is then used for the second time step (from $t = \Delta t$ to $2\Delta t$): given the input $\mathbf{U}_1^1$, the neural network produces $\mathbf{U}_2^1$ and updates the learnable parameters to Θ_{2b}. Such a process will be repeated for a time steps until the time T_{train} is reached. The neural network parameters are updated to Θ_{ab}.

(iv). Feed the second initial condition subset $\mathbf{U}_0^2$ to the neural network with parameters $\Theta_0 = \Theta_{ab}$; apply Steps (ii) and (iii) and update the neural network parameters to Θ_{2ab}. Such a process will be repeated till it loops through all the p subsets. At the end, the neural network parameters Θ_{pab} are obtained.

(v). Set $\Theta = \Theta_{pab}$ and go to the next epoch.

During the training process, a dynamic learning rate schedule is employed. Specifically, the learning rate is decreased whenever a new subset of initial conditions is fed into the network. This adjustment fine-tunes the model to different initial conditions. The training process is terminated if the loss function falls below a certain tolerance level or the maximum number of epochs is reached. In practice, the value of the hyperparameter b is also gradually reduced over time steps. It starts from a predefined large value for training the model with the first input U_0^l at the first time step, and decays after each time step until it reaches a predefined minimum value. This helps mitigate the influence of accumulated errors during the forward time stepping.

3. Numerical experiments

In this section, we first perform a series of ablation studies on the network architectures and the training strategy of the proposed ACNN and mACNN. Then, we test the trained models with several benchmark problems for the classic AC equation (1) and the conservative AC equation (4) in 2D and 3D. In all the experiments, the spatial domain $\Omega = [-0.5, 0.5]^2$ and the interfacial parameter $\epsilon = 0.01$ for 2D examples, and $\Omega = [-0.5, 0.5]^3$ and $\epsilon = 0.02$ for 3D examples. We consider the double-well potential function (2) and the Flory-Huggins potential function (3) with $\theta = 0.8$ and $\theta_c = 1.6$. Note the bounds for the classic and conservative AC equations are then respectively $\rho_1 \approx 0.95750402$ and $\rho_2 \approx 0.98678360$ in the Flory-Huggins potential case. The low and upper bounding parameters α and β in the bound limiter module are then set accordingly by expanding 1% from the corresponding theoretical values.

To train our networks, ACNN and mACNN, we choose $S = 20$ random initial conditions $u_0(x, y) = 0.9\,\mathbf{rand}(\cdot)$, where $\mathbf{rand}(\cdot)$ is the pseudo random generator producing a scalar value between -1 and 1. Following the training strategy described in Section 2.4, we split the set of 20 initial conditions into $p = 5$ subsets, each containing $q = 4$ initial conditions. For each subset, the neural network parameters are updated for b times at each time step, which is set to be $b = 500$ at the first time step and its value decreases by 10 per time step until $b = 100$ is reached. Unless stated otherwise, we adopt the Adam optimizer [15] with an initial learning rate of 0.001, and the learning rate decays by a factor of 0.6 after each subset training. We also set the convolution kernel size $K = 3$ and use $\mathbf{tanh}(\cdot)$ as the activation function in our networks. In addition, we set the maximum number of epochs to be 2 and the loss tolerance to be 1e-8. We implement our models in PyTorch, and all experiments are run on a server with a V100 GPU card with 32 GB memory. The cost of each training of ACNN or mACNN for the experiments tested in this section varies from 1 hour to 8 hours for 2D problems and about 30 hours for 3D problems, depending on the specific problem size and network architecture setting.

To measure the accuracy of our trained models, we calculate the prediction errors in the L_2 norm, $\|\mathbf{U}_{\mathcal{NN}} - \mathbf{U}_{ref}\|_2$ where $\mathbf{U}_{\mathcal{NN}}$ is the predicted solution by using the trained neural network models and $\mathbf{U}_{ref}$ is the reference solution produced by the corresponding fully-discrete schemes with the same spatial mesh and the sufficiently small time step size $\Delta t = 0.001$ (the resulting nonlinear systems are solved by using the Picard iteration and FFT-based implementation at each time step mentioned in Section 2.1).

3.1. Ablation studies

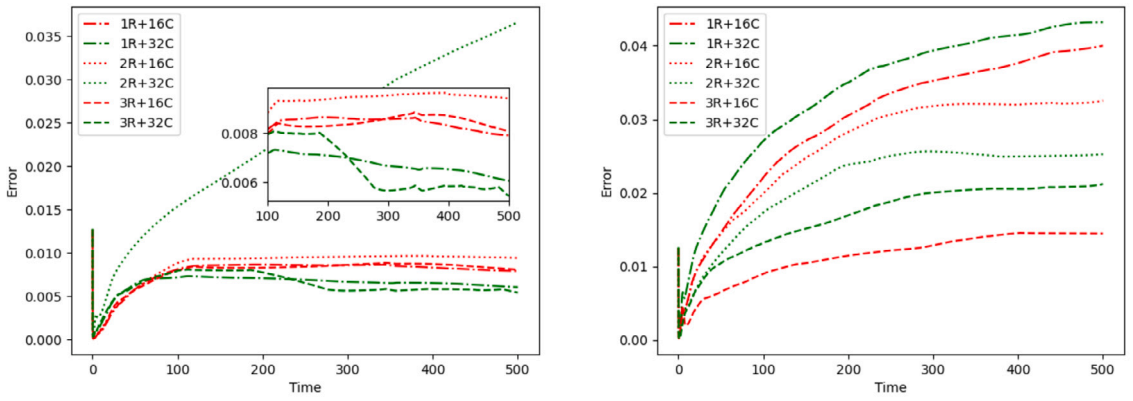
The goal of ablation studies in this section is to analyze the effect of network architecture (such as depth and width) and the choice of some hyper-parameters (such as the time step size Δt and the training time T_{train}) on the prediction ability of ACNN and mACNN, and investigate how mACNN improves the performance of solving conservative AC equations compared to ACNN. Only 2D problems with double-well are considered in the ablation studies, and we use the uniform 2D mesh of $h = 1/256$ (i.e., $N = 256$) for spatial discretization. The model prediction error is calculated by averaging results from 100 individual testing cases, each with a randomly selected initial condition that does not belong to the training set.

3.1.1. Effect of the network architecture in ACNN and mACNN

Our first objective is to investigate how the choice of depth and width of ACNN and mACNN affects the performance of the proposed methods for solving the AC equation. Specifically, we vary the number of ResBlocks from 1 to 3 to gradually increase the depth of the network and select the number of features in each convolutional layer as either 16 or 32. We also fix the training ending time $T_{train} = 10$ and the time step size $\Delta t = 0.1$, which means that our training process will cover 100 time steps. Fig. 4 and Fig. 5 present the evolution of the model prediction errors in the time interval $[0, 500]$ for ACNN and mACNN respectively under these structure settings. Note that almost all error curves have a very short period of jumping and oscillating at the beginning of the simulation, which is caused by the non-smoothness and randomness of the initial conditions (the same phenomenon also happens to conventional numerical solvers for the AC equations). It is noted that for both ACNN and mACNN, the networks with 3 ResBlocks and 16 channels appear to be robust and outperform all other configurations in both potential function cases. Even with a simulation ending time of 500, which is 50 times longer than the training duration, ACNN and mACNN with this particular structure still perform very well and the model prediction errors consistently remain under the level of 2e-2. Thus, we will use this particular structure for our ACNN and mACNN in all subsequent experiments.

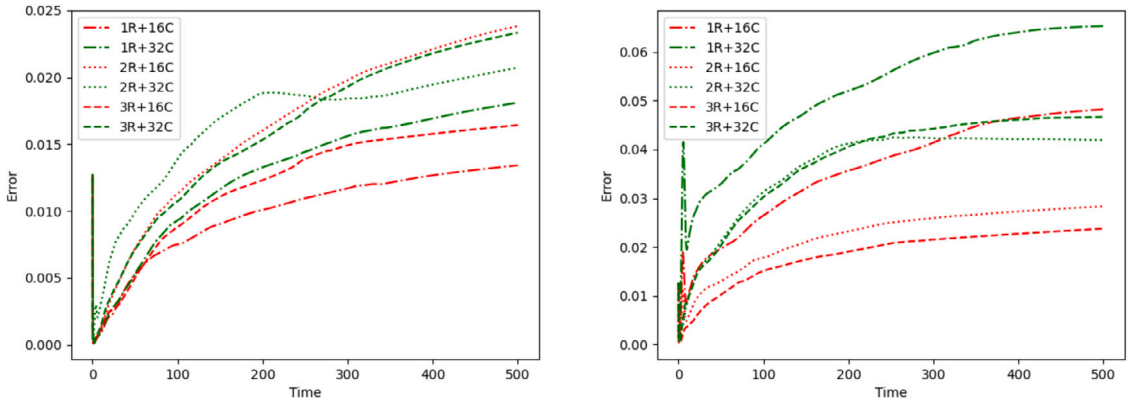
3.1.2. Effect of the time step size Δt and the training ending time T_{train}

To test the effect of the time step size Δt on the model prediction accuracy and check if our networks produce convergent results, we train and test ACNN and mACNN using different time step sizes $\Delta t = 0.05, 0.1$, and 0.2 respectively, by using the double-well potential and fixing $T_{train} = 10$ (correspondingly, $a = 200$, 100 and 50 time steps used for training). In general, a smaller step size leads to a smaller discretization error, but the model prediction accuracy is also affected by the network errors such as those caused by the representation ability of the networks, the effectiveness of the training, and the needed number of time steps (a smaller time step size requires more time steps to reach the same simulation time). The overall performance of our networks is determined by the



(a) Classic AC with double-well (b) Classic AC with Flory-Huggins

Fig. 4. Evolution of the ACNN model prediction errors for solving the classic AC equation (1) in 2D: (a) with the double-well potential and (b) with the Flory-Huggins potential. Note $h = 1/256$, $\Delta t = 0.1$ and $T_{train} = 10$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)



(a) Conservative AC with double-well (b) Conservative AC with Flory-Huggins

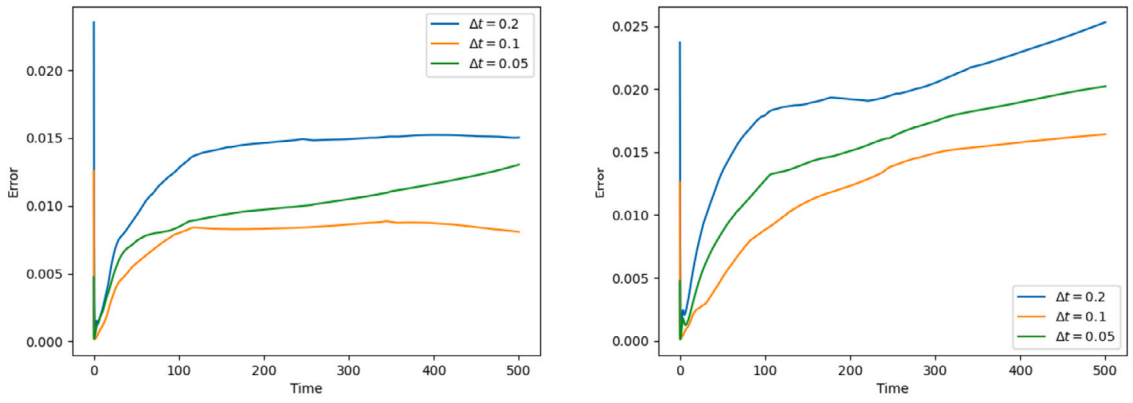
Fig. 5. Evolution of the mACNN model prediction errors for solving the conservative AC equation (4) in 2D: (a) with the double-well potential; (b) with the Flory-Huggins potential. Note $h = 1/256$, $\Delta t = 0.1$ and $T_{train} = 10$.

Table 1
The model prediction errors of ACNN and mACNN at $t = 5, 20, 100, 500$ when using different time step sizes for solving the classic AC equation (1) and the conservative AC equation (4) with the double-well potential in 2D. Note $h = 1/256$ and $T_{train} = 10$.

Error	ACNN			mACNN		
	$\Delta t = 0.2$	$\Delta t = 0.1$	$\Delta t = 0.05$	$\Delta t = 0.2$	$\Delta t = 0.1$	$\Delta t = 0.05$
$t = 5$	1.3942e-3	5.3620e-4	1.2754e-3	2.1655e-3	6.1348e-4	1.5406e-3
$t = 20$	5.3144e-3	2.8855e-3	4.1812e-3	7.1158e-3	2.5030e-3	4.1367e-3
$t = 100$	1.2778e-2	7.9914e-3	1.0008e-2	1.7908e-2	8.8159e-3	1.6237e-2
$t = 500$	1.5032e-2	8.0758e-3	1.3033e-2	2.5319e-2	1.6417e-2	2.0222e-2

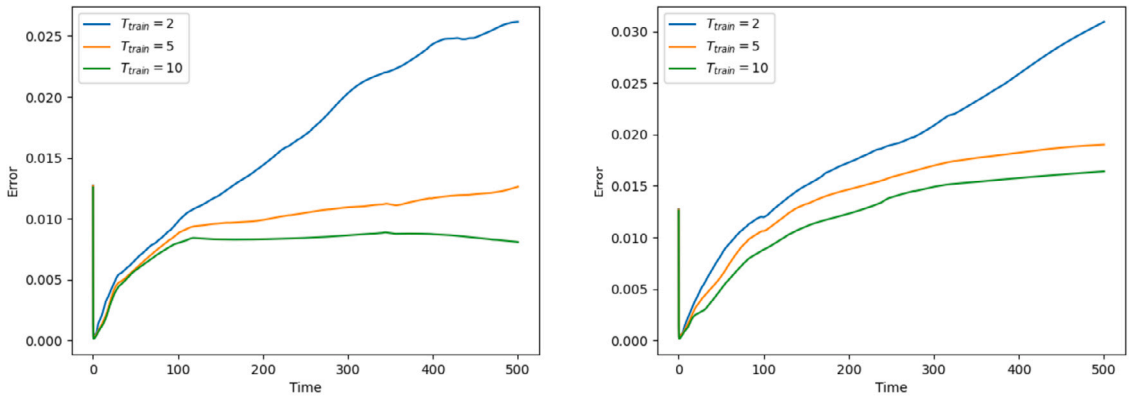
combination of all these issues. Table 1 reports the model prediction errors of ACNN and mACNN at $t = 5, 20, 100, 500$ when using different time step sizes respectively to solve the classic AC and mass conservative AC equations. The time evolutions of the model prediction errors during the time interval $[0, 500]$ are shown in Fig. 6. It is seen that ACNN and mACNN using $\Delta t = 0.1$ perform better than the ones with $\Delta t = 0.2$ and $\Delta t = 0.05$. Thus we take $\Delta t = 0.1$ in the remaining experiments.

The value of T_{train} also has impact on ACNN and mACNN in terms of prediction accuracy and training efficiency. To quantify its effect, we consider three values $T_{train} = 2, 5$, and 10 (correspondingly $a = 20, 50, 100$ time steps in training). The associated model prediction errors of ACNN and mACNN are shown in Fig. 7. It is observed that, for both ACNN and mACNN, the choice of $T_{train} = 2$ results in the worst prediction accuracy. Longer training intervals do lead to improved model prediction, but the extent of improvement gradually gets smaller along the increasing of the training time.



(a) by ACNN (b) by mACNN

Fig. 6. Evolution of the model prediction errors in the double-well potential case when different time step sizes are used: (a) ACNN for solving the classic AC equation (1); (b) mACNN for solving the conservative AC equation (4). Note $h = 1/256$ and $T_{train} = 10$.



(a) by ACNN (b) by mACNN

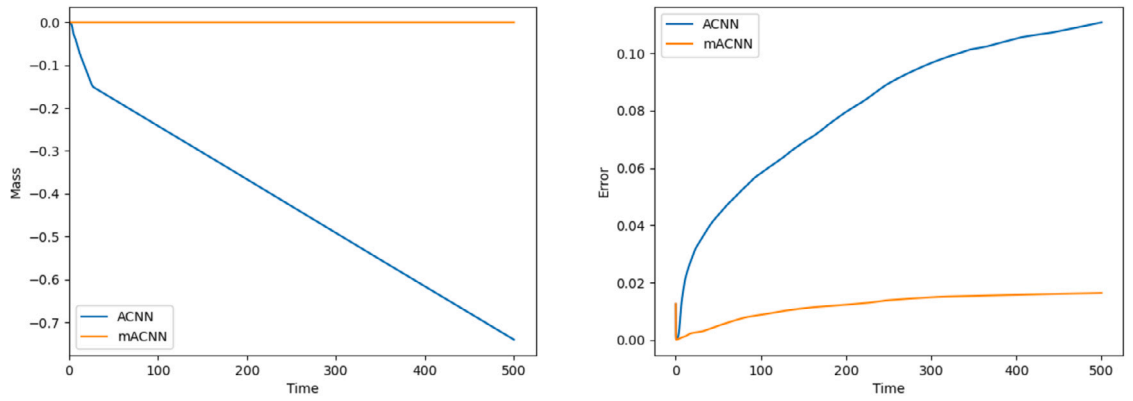
Fig. 7. Evolution of the model prediction errors in the double-well potential case when the final training time T_{train} varies: (a) ACNN for solving the classic AC equation (1); (b) mACNN for solving the conservative AC equation (4). Note $h = 1/256$ and $\Delta t = 0.1$.

3.1.3. Effect of the hard constraint for mass conservation in mACNN

To illustrate the advantage of employing the hard constraint on mass conservation in mACNN, we compare the performance of ACNN to mACNN based on the same loss function (13) in learning the dynamics of the conservative AC equation. We remark that the ACNN can also be applied to the conservative AC equation since the loss function (13) does implicitly enforce the mass conservation from t_n to t_{n+1} (i.e., treating the mass conservation as a soft constraint). We use $\Delta t = 0.1$ and $T_{train} = 10$ for training. Fig. 8 presents the evolution of total mass and model prediction errors where the trained ACNN and mACNN are applied to simulate the conservative AC equation over $[0, 500]$. It is observed that ACNN fails to preserve the total mass while mACNN does so perfectly during the simulation. Meanwhile, mACNN achieves much smaller prediction errors (around 2e-2 during nearly the entire time interval) than ACNN. In other words, adding the hard constraint of the mass in mACNN not only ensures the predicted solution complies with the mass conservation, but also greatly enhances the accuracy of the prediction.

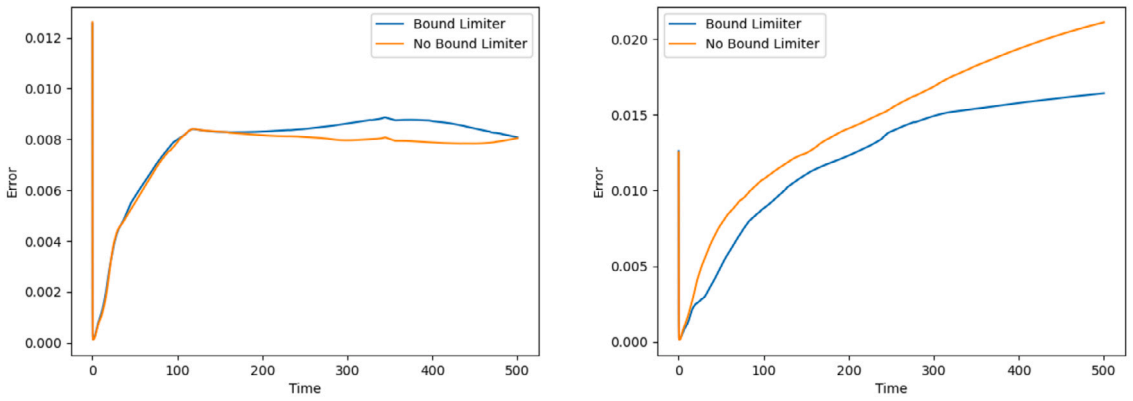
3.1.4. Effect of the bound limiter for ACNN and mACNN

Finally, we conduct experiments to evaluate the effect of the bound limiter module on the performance of ACNN and mACNN. We set $\Delta t = 0.1$ and $T_{train} = 10$ in the training process for the classic and conservative AC equations with the double-well potential. Fig. 9 shows the errors associated with ACNN and mACNN, both when the bound limiter is incorporated into the neural networks and when it is not. Our observations indicate that when solving the classic AC equation, the presence of the bound limiter does not significantly impact the training accuracy. However, in the context of the conservative AC equation, the inclusion of the bound limiter enhances the accuracy during the entire time evolution. In addition, we would like to remark that it would become crucial to integrate the bound limiter in learning the two AC equations with the Flory-Huggins potential. Due to the presence of the logarithmic term (i.e., $\ln \frac{1-u}{1+u}$) in the corresponding loss functions (12) and (13), $|u| \leq 1$ is strictly required in the whole training process. Without imposing appropriate bounds for u, the training loss often becomes infinity at early stages of the training as we have observed from experiments.



(a) Total mass (b) Prediction error

Fig. 8. Comparison between the predicted solutions by ACNN and mACNN for solving the conservative AC equation (4) with the double-well potential in 2D: (a) total mass; (b) prediction error. Note $h = 1/256$, $\Delta t = 0.1$ and $T_{train} = 10$.



(a) ACNN (b) mACNN

Fig. 9. Comparison between the predicted solutions in the double-well potential case produced by ACNN and mACNN with or without the bound limiter module: (a) the classic AC equation (1); (b) the conservative AC equation (4). Note $h = 1/256$, $\Delta t = 0.1$ and $T_{train} = 10$.

3.2. Dynamics prediction for 2D examples

Next, we demonstrate through two well-known 2D benchmark examples the excellent performance of our proposed methods, in terms of both accuracy and generalization ability, for learning the dynamics of the classic and conservative AC equations. One is the *bubble merging* problem, the other is the *grain coarsening* problem. They are governed by either the classic AC equation (1) or the conservative AC equation (4). Based on the ablation studies done in Section 3.1, we choose $h = 1/256$ for the spatial mesh size, and set $\Delta t = 0.1$ and $T_{train} = 10$ in both ACNN and mACNN. The trained neural networks are applied to predict solutions of the benchmark problems.

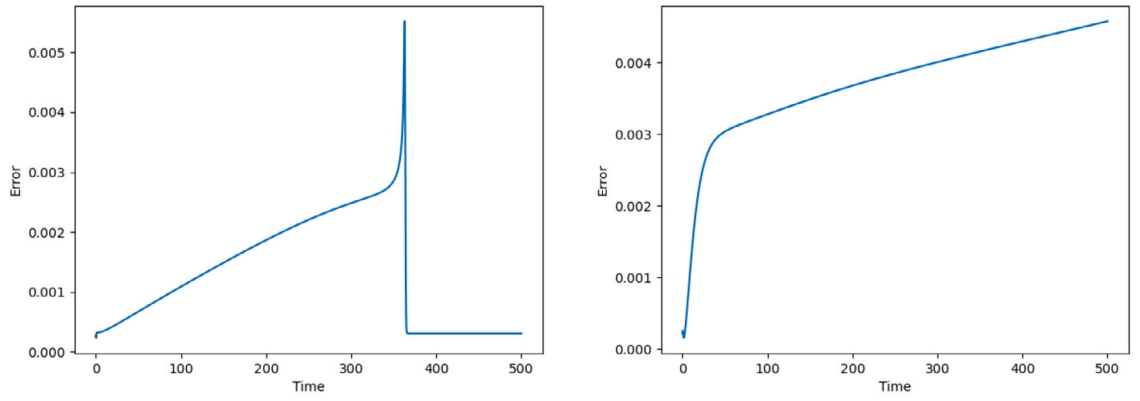
3.2.1. 2D bubble merging

To showcase the generalization ability of ACNN and mACNN, we test them using the 2D bubble merging example, which takes the following initial condition:

$$u_0(x, y) = \max\left(\tanh\left(\frac{0.2 - \sqrt{(x-0.14)^2 + y^2}}{\epsilon}\right), \tanh\left(\frac{0.2 + \sqrt{(x+0.14)^2 + y^2}}{\epsilon}\right)\right).$$

Only the double-well potential function is considered for this example. Fig. 10 presents the evolution of the prediction errors during the time interval $[0, 500]$ produced by ACNN for the classic AC equation and by mACNN for the conservative AC equation. It is found that both networks can accurately predict the corresponding dynamics and the prediction errors always remain under about the level of 5e-3 for both ACNN and mACNN.

The predicted solution by ACNN and associated errors at the times $t = 0, 10, 50, 200, 300$ are plotted in Fig. 11 for the classic AC equation. The time evolution of mass, energy and maximum norm of the predicted solution are shown in Fig. 12, where the red line



(a) by ACNN for the classic AC (b) by mACNN for the conservative AC

Fig. 10. Evolution of the model prediction errors in the 2D bubble merging example.

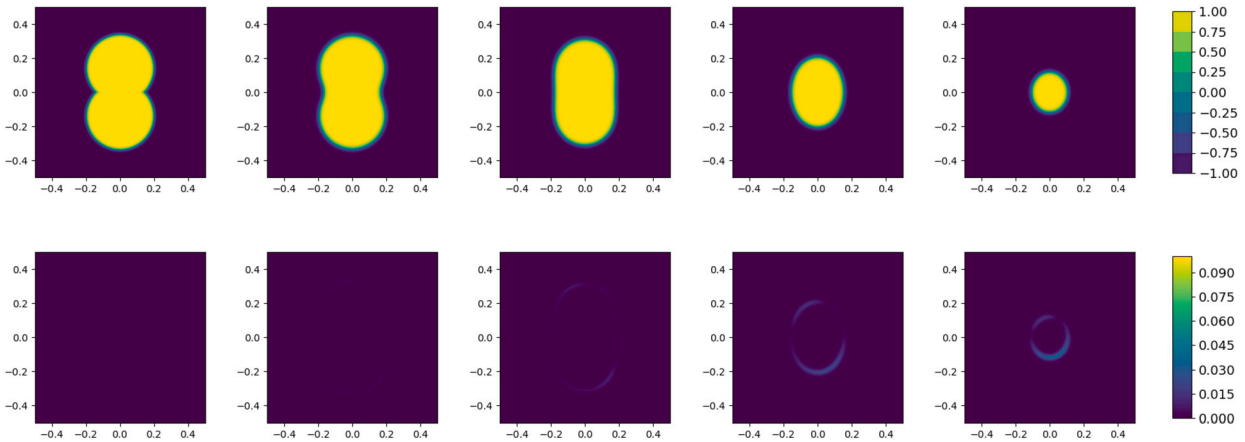


Fig. 11. Plots of the ACNN prediction results (top row) and associated numerical errors (bottom row) at the times $t = 0, 10, 50, 200, 300$ for solving the classic AC equation (1) with the double-well potential in the 2D bubble merging example.

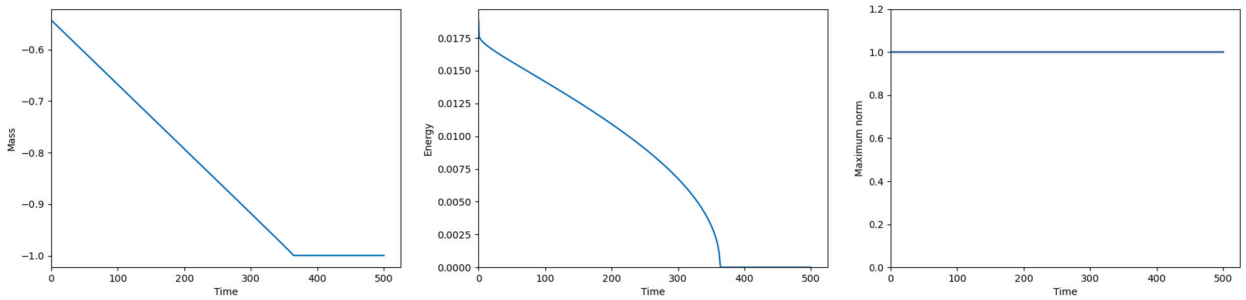


Fig. 12. Evolution of the mass (left), the energy (middle), and the maximum norm (right) of the ACNN predicted solution for the classic AC equation (1) with the double-well potential in the 2D bubble merging example.

indicates the value of theoretical bound $\alpha = 1$ in the rightmost figure. We observe that the two disks gradually shrink and merge into one smaller disk, which keeps shrinking and finally disappears around $t = 360$. The prediction errors seem to only occur in the transition regions of two phases, which implies that the phase shapes are accurately captured. The energy monotonically decays and stays at 0 after the disk disappears. The predicted solution also surprisingly preserves well the MBP, i.e., its maximum norm is always bounded by 1.

Fig. 13 plots the predicted solution by mACNN and corresponding prediction errors for the conservative AC equation at the times $t = 0, 10, 50, 200, 500$. We can see that the two disks gradually merge together and finally form a perfect disk with the same area as the initial state due to the property of mass conservation. Fig. 14 shows the evolution of mass, energy and maximum norm of the

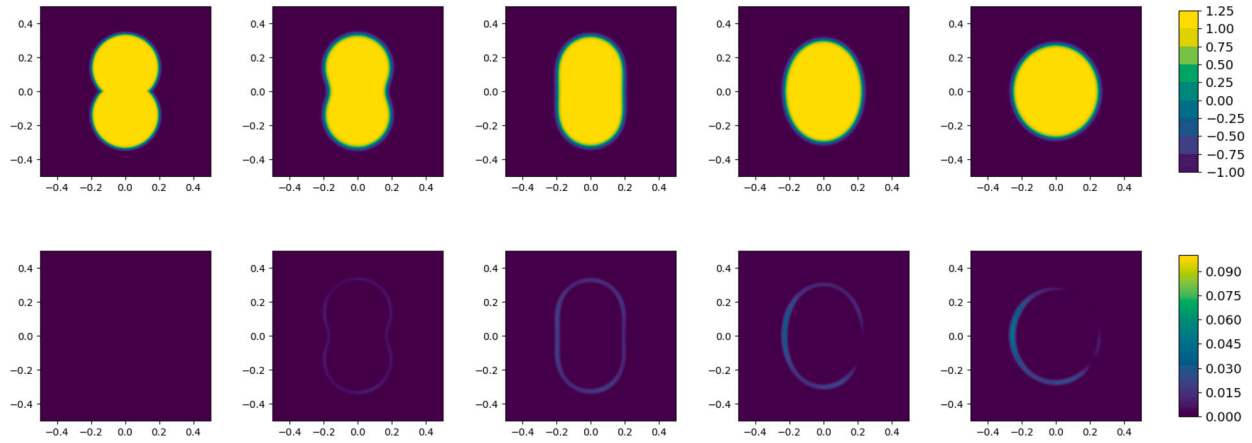


Fig. 13. Plots of the mACNN prediction results (top row) and corresponding numerical errors (bottom row) for solving the conservative AC equation (4) with the double-well potential at the times $t = 0, 10, 50, 200, 500$ in the 2D bubble merging example.

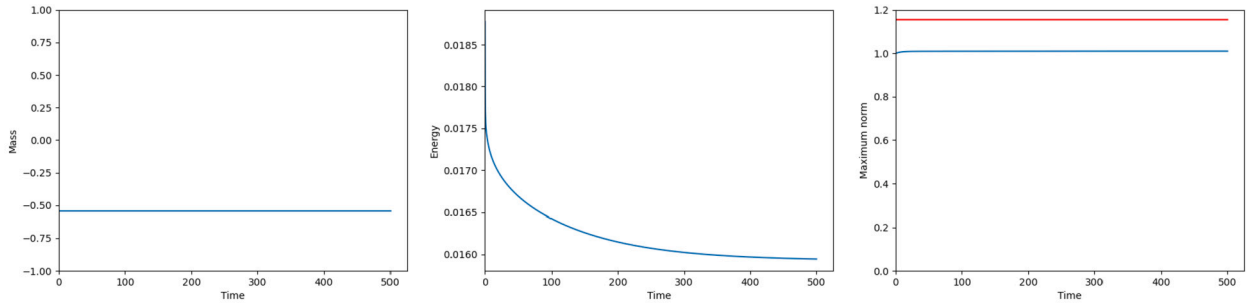


Fig. 14. Evolution of the mass (left), the energy (middle), and the maximum norm (right) of the mACNN predicted solution for the conservative AC equation (4) with the double-well potential in the 2D bubble merging example.

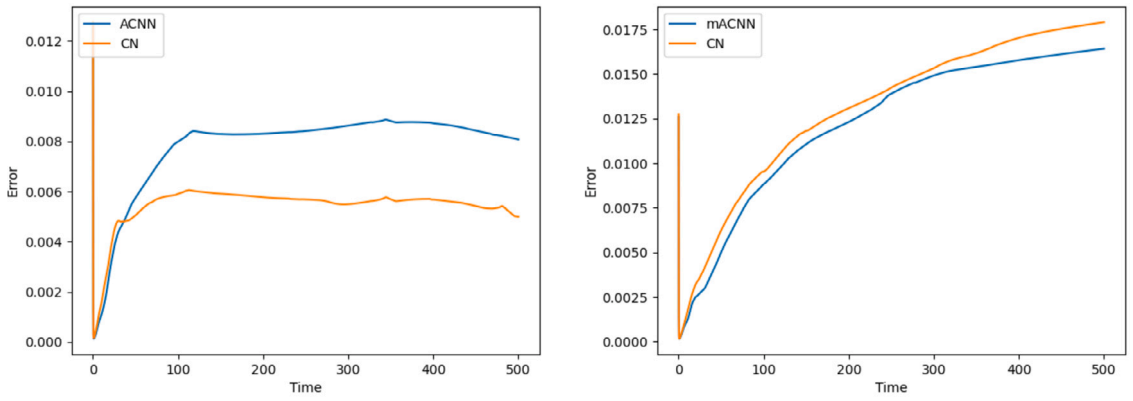
predicted solution. The prediction errors again only occur in the transition regions of the two phases. It is seen that the mass is exactly conserved along the time and the energy monotonically decays until the steady state is reached. We also note that the maximum norm of the predicted solution is a little bit larger than 1, but much less than the theoretical bound $\alpha = \frac{2\sqrt{3}}{3}$ (since this theoretical bound is not sharp), indicated by the red line in the plot. This phenomenon is also observed when conventional numerical methods are considered, for example, in [16].

3.2.2. 2D grain coarsening

Next, we use the trained ACNN and mACNN to simulate the 2D grain coarsening problem, where an initial condition is chosen randomly of the form $u_0(x, y) = 0.9\,\mathbf{rand}(\cdot)$ and then applied to all tests. Both the double-well potential and the Flory-Huggins potential are considered. Fig. 15 presents the evolution of the prediction errors during the time interval $[0, 500]$ produced by ACNN for the classic AC equation and by mACNN for the conservative AC equation with the double-well potential, together with the errors produced by the convectional Picard iteration solver with the Crank–Nicolson approximations under the same time step and spatial mesh sizes. We observe that our networks are able to accurately predict the dynamics. Indeed, the prediction errors are less than 1e-2 for ACNN and 1.8e-2 for mACNN during the entire simulation, which are close to those generated by the Crank–Nicolson scheme. This implies that our networks are able to successfully learn the fully-discrete operators FDCN_{AC} and FDCN_{mAC}. The same behavior can be observed in Fig. 16 when the two AC equations with the Flory-Huggins potential are considered.

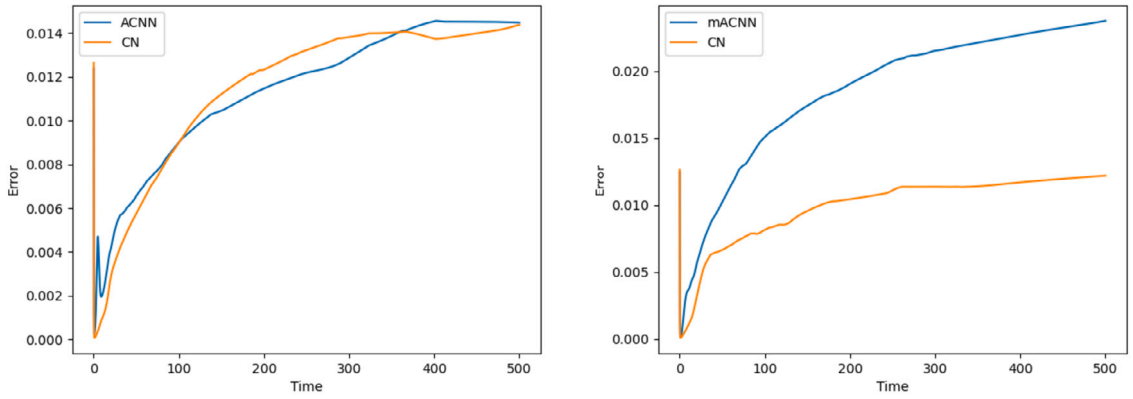
In addition, the average running time per step for prediction using ACNN or mACNN is only around 0.002 seconds, while, under the same computing environment using GPUs for parallel computing in PyTorch, the running time spent by the conventional Picard iteration solver (the termination criterion is that the maximum absolute change is less than 1e-10 or the iteration number reaches 40) is around 0.02 seconds per step, which is 10 times larger. This makes the proposed deep learning method more favorable in practice while still maintaining similar numerical accuracy.

We plot in Fig. 17 the predicted solution by ACNN and corresponding prediction errors for the classic AC equation with the double-well potential at the times $t = 0, 10, 50, 200, 500$, and in Fig. 18 the evolution of mass, energy and maximum norm of the predicted solution. It is seen that in the process of grain coarsening, one phase (the yellow-colored one) gradually gets smaller and smaller, which will finally vanish after $t = 500$. Fig. 19 shows the predicted solution by mACNN for the conservative AC equation with the double-well potential at the same time instances together with corresponding prediction errors, and Fig. 20 plots the evolution of mass, energy and maximum norm of the predicted solution. In this situation, we still clearly observe the grain coarsening



(a) by ACNN for the classic AC equation (b) by mACNN for the conservative AC equation

Fig. 15. Evolution of the prediction errors for solving AC equation with the double-well potential by ACNN and mACNN, compared with the numerical solutions of the Crank–Nicolson scheme in the 2D grain coarsening example. Note $h = 1/256$ and $\Delta t = 0.1$.



(a) by ACNN for the classic AC equation (b) by mACNN for the conservative AC equation

Fig. 16. Evolution of the prediction errors for solving AC equations with the Flory-Huggins potential by ACNN and mACNN, compared with the numerical solutions obtained by the Crank–Nicolson scheme in the 2D grain coarsening example. Note $h = 1/256$ and $\Delta t = 0.1$.

process, but the coarsening finally reaches a steady state without any phase disappeared due to the property of mass conservation. The prediction results by ACNN and mACNN for the classic and conservative AC equations with the Flory-Huggins potential are respectively presented in Figs. 21, 22, 23 and 24. The overall performance is similar to those in the double-well potential case in terms of the simulated solutions and their errors.

3.3. Dynamics prediction for 3D examples

Finally, we investigate the proposed method on 3D benchmark problems with the interfacial parameter $\epsilon = 0.02$. Represent the set of grid points by $\{(x_i, y_j, z_k)\}_{i,j,k=0}^{N}$ with $x_i = ih$, $y_j = jh$ and $z_k = kh$. Based on the central finite difference in space and the Crank-Nicolson in time, the fully-discrete system for the classic AC equation (1) reads:

$$\frac{U_{n+1}^{i,j,k} - U_n^{i,j,k}}{\Delta t} = \epsilon^2 \frac{\Delta_h U_{n+1}^{i,j,k} + \Delta_h U_n^{i,j,k}}{2} + \frac{f(U_{n+1}^{i,j,k}) + f(U_n^{i,j,k})}{2}, \tag{14}$$

for $i, j, k = 1, 2, \cdots, N$, where Δ_h denotes the seven-point stencil for the 3D Laplacian as

$$\Delta_h U_{n+1}^{i,j,k} = \frac{U_{n+1}^{i+1,j,k} + U_{n+1}^{i-1,j,k} + U_{n+1}^{i,j+1,k} + U_{n+1}^{i,j-1,k} + U_{n+1}^{i,j,k-1} + U_{n+1}^{i,j,k+1} - 6U_{n+1}^{i,j,k}}{h^2},$$

with $U_n^{i,j,k} \approx u(x_i, y_j, z_k, t_n)$. The fully-discrete system for the mass-conservative AC equation (4) in 3D becomes:

$$\frac{U_{n+1}^{i,j,k} - U_n^{i,j,k}}{\Delta t} = \epsilon^2 \frac{\Delta_h U_{n+1}^{i,j,k} + \Delta_h U_n^{i,j,k}}{2} + \frac{f(U_{n+1}^{i,j,k}) + f(U_n^{i,j,k})}{2} - \frac{1}{N^3} \sum_{i,j,k=1}^{N} \frac{f(U_{n+1}^{i,j,k}) + f(U_n^{i,j,k})}{2}, \tag{15}$$

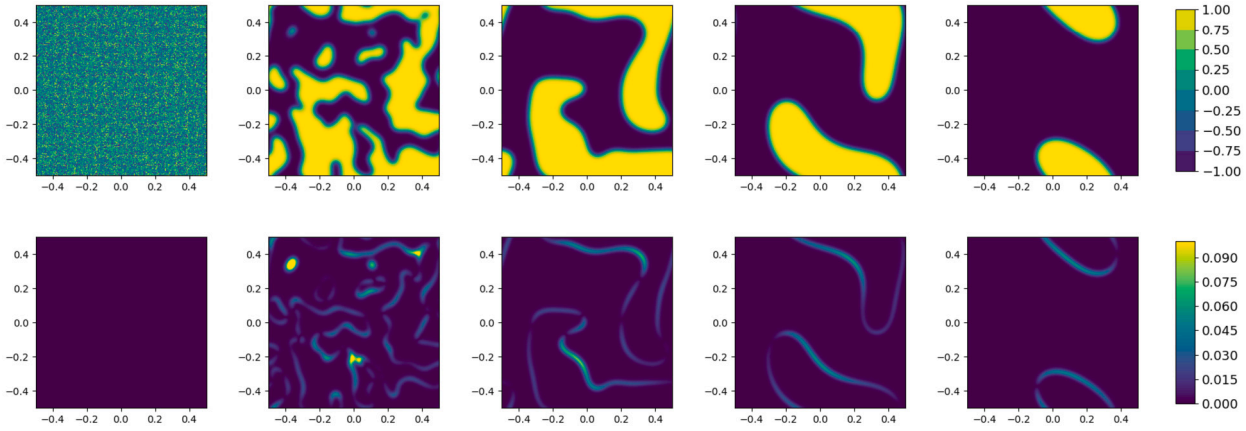


Fig. 17. Plots of the ACNN prediction results (top row) and corresponding prediction errors (bottom row) at the times $t = 0, 10, 50, 200, 500$ for solving the classic AC equation (1) with the double-well potential in the 2D grain coarsening example.

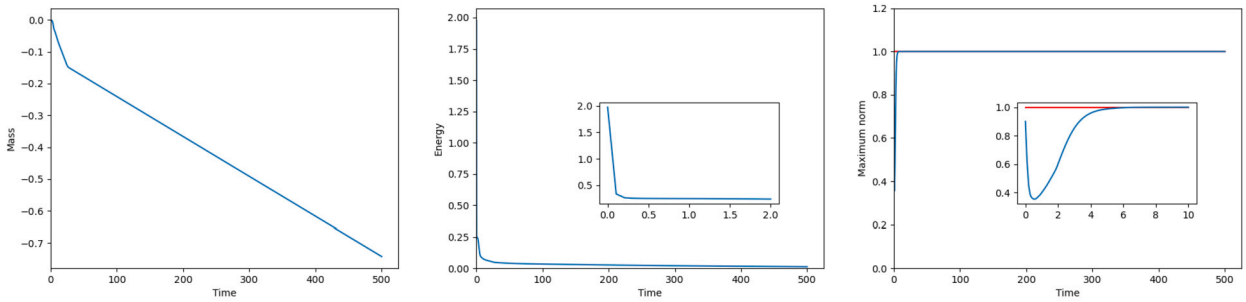


Fig. 18. Evolution of the mass (left), the energy (middle), and the maximum norm (right) of the ACNN predicted solution for the classic AC equation (1) with the double-well potential in the 2D grain coarsening example.

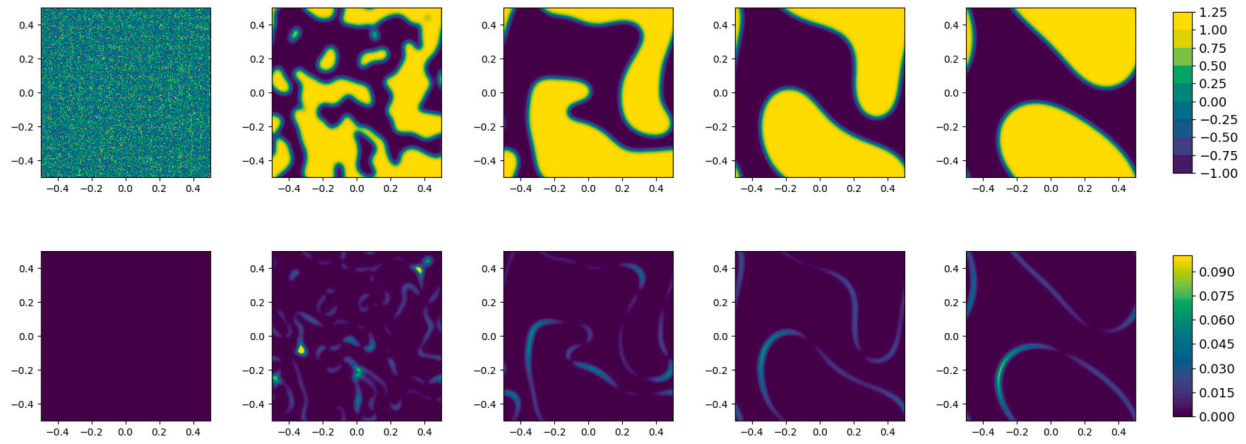


Fig. 19. Plots of the mACNN prediction results (top row) and corresponding prediction errors (bottom row) at the times $t = 0, 10, 50, 200, 500$ for solving the conservative AC equation (4) with the double-well potential in the 2D grain coarsening example.

for $i, j, k = 1, 2, \cdots, N$.

The same network architectures and training strategy as used for 2D problems are used to construct the ACNN and mACNN for 3D problems. Correspondingly, the input U_n now becomes a $N \times N \times N \times 1$ tensor, and the filter size in each convolutional layer changes to $K \times K \times K$ with $K = 3$. We set $\Delta t = 0.1$, $T_{train} = 5$, and the spatial mesh size $h = 1/64$. Since the interfacial parameter $\epsilon = 0.02$ is used for 3D problems, instead of $\epsilon = 0.01$ for 2D examples, we expect the coarsening process takes much short time to reach the steady state, therefore shorter simulation intervals will be considered.

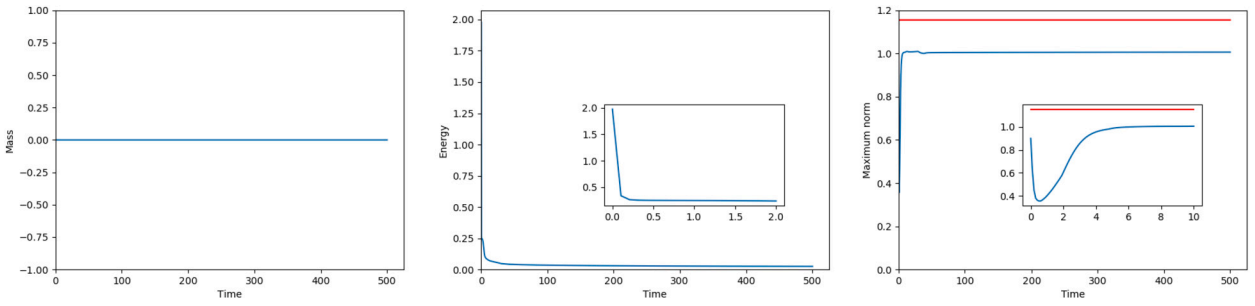


Fig. 20. Evolution of the mass (left), the energy (middle), and the maximum norm (right) of the mACNN predicted solution for the conservative AC equation (4) with the double-well potential in the 2D grain coarsening example.

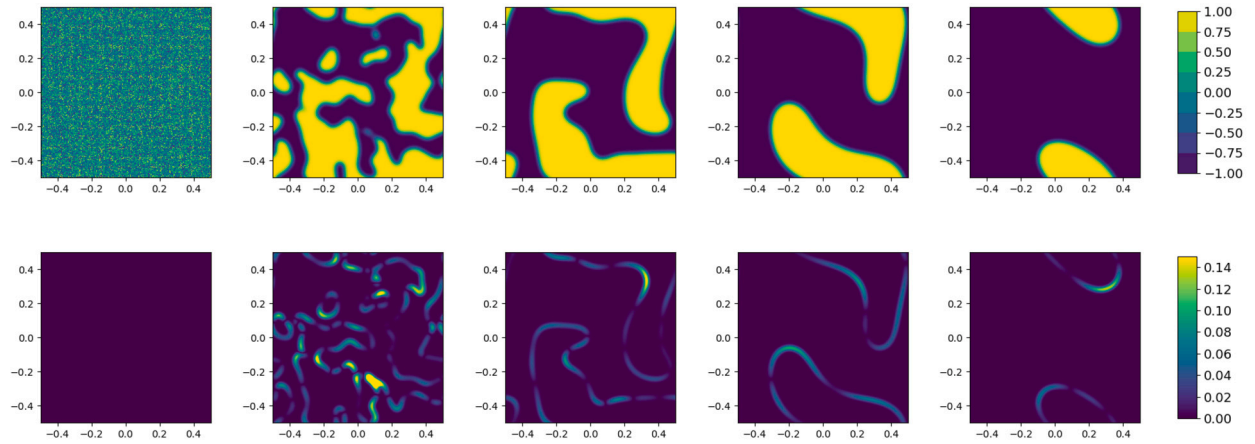


Fig. 21. Plots of the ACNN prediction results (top row) and corresponding prediction errors (bottom row) at the times $t = 0, 10, 50, 200, 500$ for solving the classic AC equation (1) with the Flory–Huggins potential in the 2D grain coarsening example.

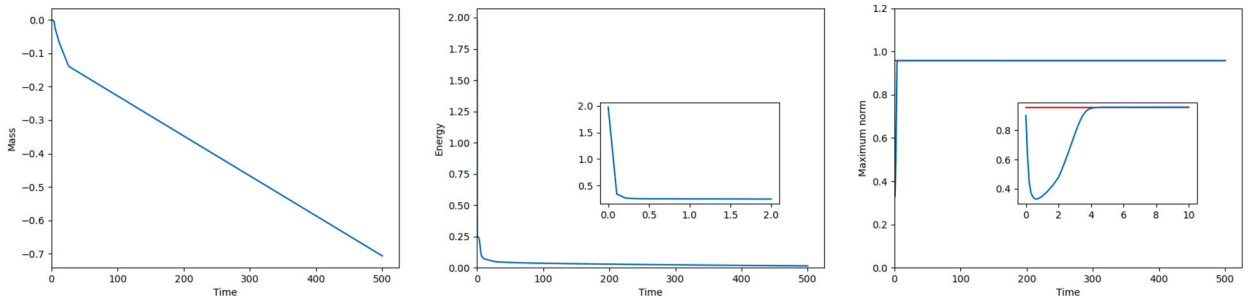


Fig. 22. Evolution of the mass (left), the energy (middle), and the maximum norm (right) of the ACNN predicted solution for the classic AC equation (1) with the Flory–Huggins potential in the 2D grain coarsening example.

3.3.1. 3D bubble merging

We first test our ACNN and mACNN with the 3D bubble merging example for solving AC equations with double-well potential function, which takes the following initial condition:

$$u_0(x, y, z) = \max\left(\tanh\left(\frac{0.2 - \sqrt{(x-0.14)^2 + y^2 + z^2}}{\epsilon}\right), \tanh\left(\frac{0.2 + \sqrt{(x+0.14)^2 + y^2 + z^2}}{\epsilon}\right)\right).$$

Again only the double-well potential function is considered for this example. The simulation time interval is set to be $[0, 50]$ for the classic AC equation and $[0, 100]$ for the conservative AC equation. Fig. 25 shows the iso-surfaces (value 0) of the predicted solution by ACNN at the time instances $t = 0, 10, 20, 30$ for the classic AC equation and the evolution of mass, energy and maximum norm of the predicted solution. We observe that the two balls gradually shrink and merge into one smaller ball, that finally disappears as expected (roughly at t = 36). The energy decays monotonically and stays at 0 after the ball disappears, and the maximum bound 1 is well preserved by the predicted solution. Fig. 26 presents the iso-surface (value 0) of the predicted solution by mACNN at the time instances $t = 0, 10, 30, 100$ for the conservative AC equation, together with the evolution of mass, energy and maximum norm of the

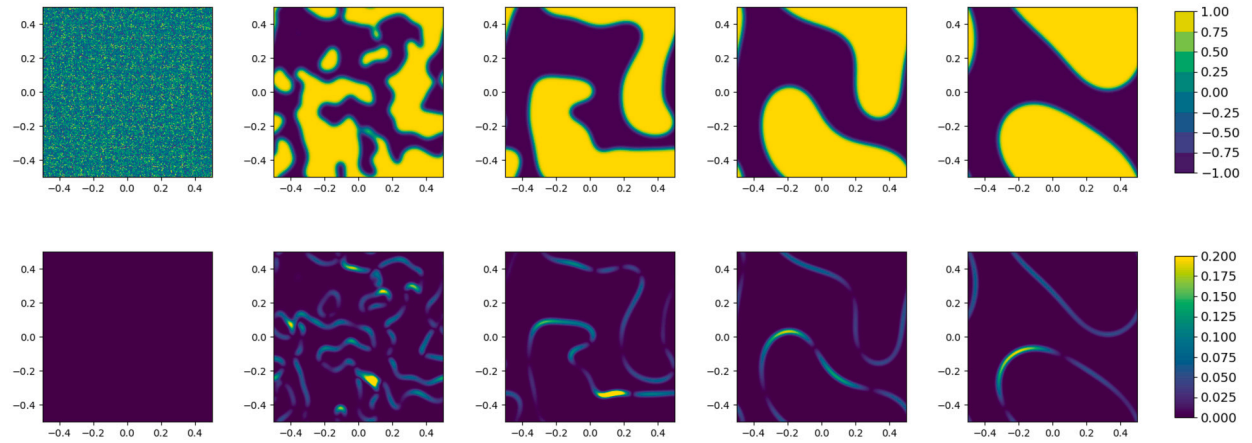


Fig. 23. Plots of the mACNN prediction results (top row) and corresponding prediction errors (bottom row) at the times $t = 0, 10, 50, 200, 500$ for solving the conservative AC equation (4) with the Flory-Huggins potential in the 2D grain coarsening example.

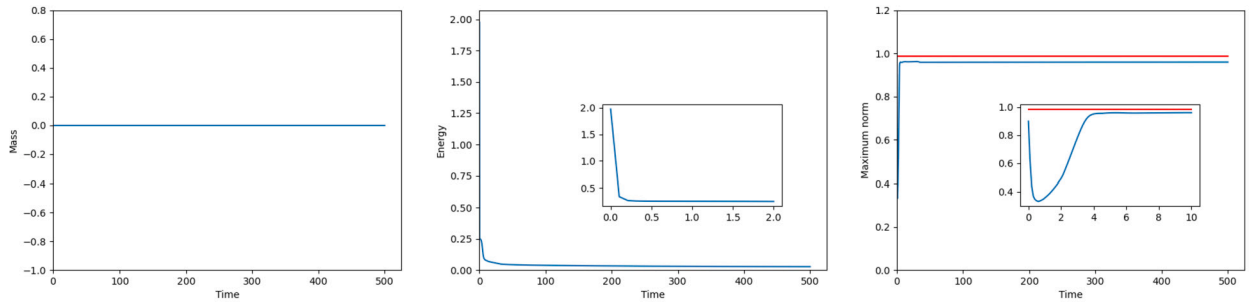


Fig. 24. Evolution of the mass (left), the energy (middle), and the maximum norm (right) of the mACNN predicted solution for the conservative AC equation (4) with the Flory-Huggins potential in the 2D grain coarsening example.

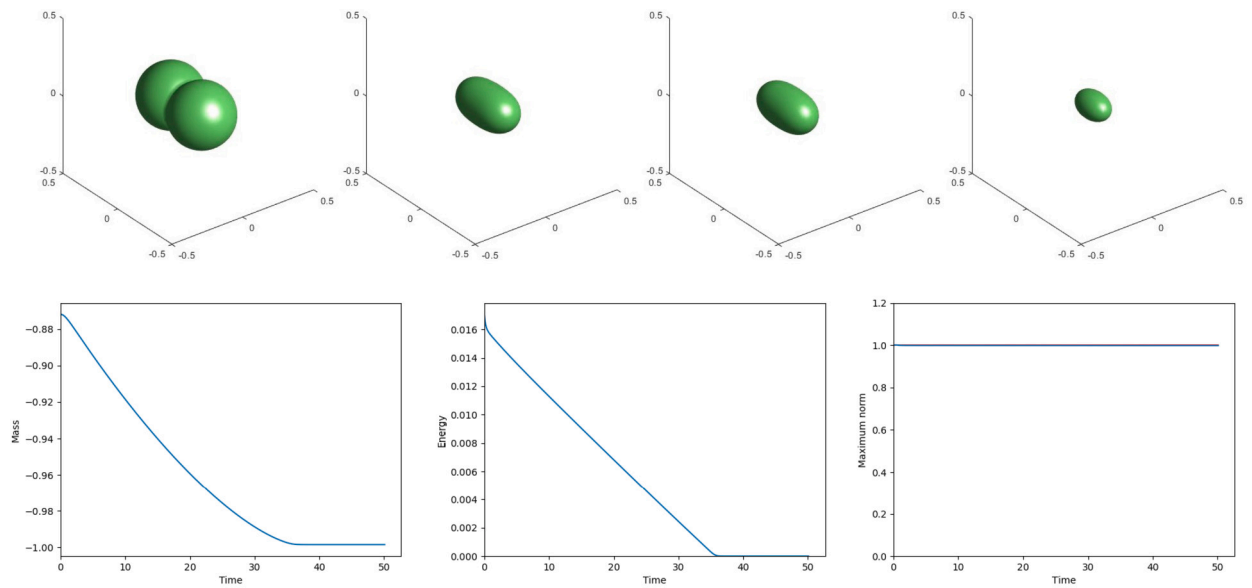


Fig. 25. The ACNN prediction results for the classic AC equation (1) with the double-well potential in the 3D bubble merging example. Top row: plots of the iso-surfaces (value 0) of the predicted solution at the time instances t = 0, 10, 20, 30; Bottom row: evolution of the mass (left), the energy (middle), and the maximum norm (right) of the predicted solution.

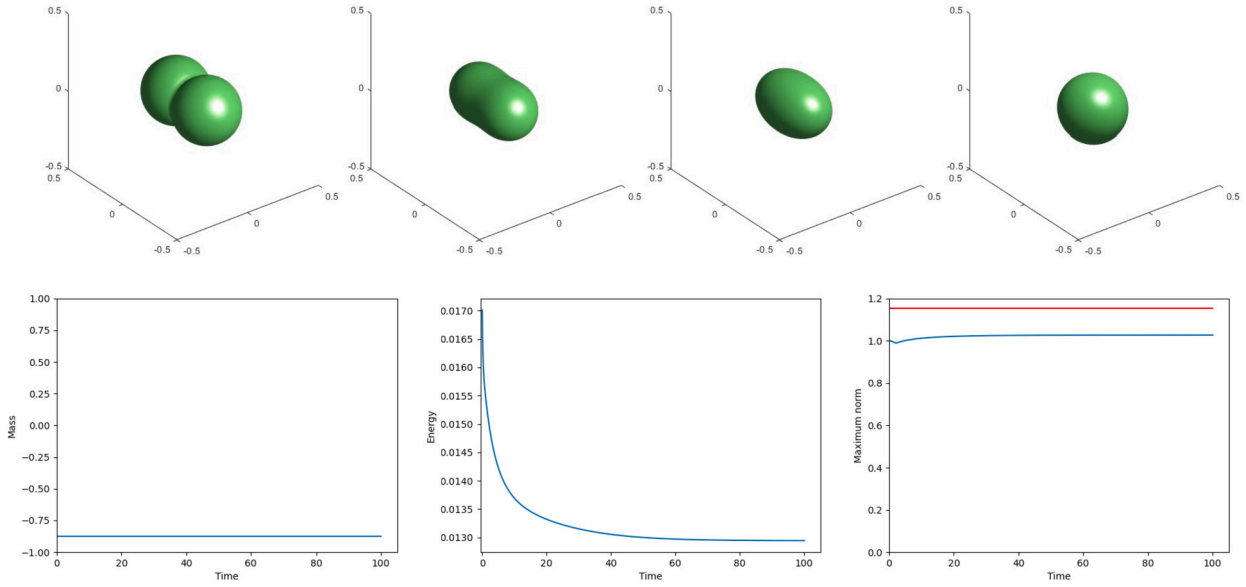


Fig. 26. The mACNN prediction results for the conservative AC equation (4) with the double-well potential in the 3D bubble merging example. Top row: plots of the iso-surfaces (value 0) of the predicted solution at the times t = 0, 10, 30, 100; Bottom row: evolution of the mass (left), the energy (middle), and the maximum norm (right) of the predicted solution.

predicted solution. We observe that the two balls gradually merge together and finally form a perfect ball with the same volume as the initial state due to the mass conservation in this case. In addition, the mass is exactly conserved, and the maximum norm of the predicted solution is a little bit larger than 1.

3.3.2. 3D grain coarsening

Finally, we use the trained ACNN and mACNN to predict the 3D grain coarsening dynamics with an initial condition generated randomly by $u_0(x, y, z) = 0.9\,\mathbf{rand}(\cdot)$. The simulation time interval is set to be $[0, 100]$ for both the classic and conservative AC equations. For the double-well potential case, the iso-surfaces (value 0) of the predicted solution at the times $t = 0, 10, 50, 100$ obtained by ACNN for the classic AC equation and by mACNN for the conservative AC equation are displayed in Fig. 27 and Fig. 28, respectively, together with the time evolution of the associated mass, energy and maximum norm of the predicted solution. Fig. 29 and Fig. 30 show the prediction results of the AC equations with the Flory-Huggins potential and the performance is again similar. They clearly show that the grain coarsening process in both cases. Indeed, one phase gets smaller and smaller in the classic AC setting, while in the conservative AC setting the coarsening finally reaches a steady state without any phase disappeared due to the mass conservation. The energy dissipation and the maximum bound of the predicted solutions again show similar behaviors as observed in the previous example. The average running time per step using ACNN or mACNN is only around 0.005 seconds for these 3D problems.

4. Conclusion

In this paper, we have introduced two novel neural network architectures in the end-to-end fashion, ACNN and mACNN, for learning the dynamics of classic AC equation and the conservative AC equation respectively. The loss functions are constructed to minimize the residuals of the fully-discrete system of the target equations based on the central finite difference discretization in space and the Crank–Nicolson approximation in time. We also design an effective training strategy for our neural networks without using any ground truth data. We demonstrate the outstanding performance of our proposed models, in terms of prediction accuracy and computational cost, through extensive numerical tests and comparisons in two and three dimensions.

Our current networks for ACNN and mACNN are designed with a fixed time step size and for problems defined on rectangular domains. To extend the applicability of our method, we first aim to develop new network structures that can handle both variable time step sizes and more general domains. The former one is especially useful for the adaptive time stepping method for long term simulations. It also remains very interesting to introduce some attention mechanisms to our ACNN and mACNN to further enhance their prediction and generalization ability. In addition, the framework proposed in the paper also could be used to solve other second-order and even high-order nonlinear PDEs. One such equation is the so-called Cahn-Hilliard (CH) equation, which is a fourth-order nonlinear parabolic PDE that satisfies the mass conservation law, and has many important applications in phase field modeling. The CH equation is notoriously stiff, requiring extremely small time steps even with fully implicit numerical methods, making it highly time-consuming to solve. Our framework offers a promising direction for developing efficient deep learning methods to solve such stiff equations in a fraction of the time. It would open up exciting possibilities for future research in this area.

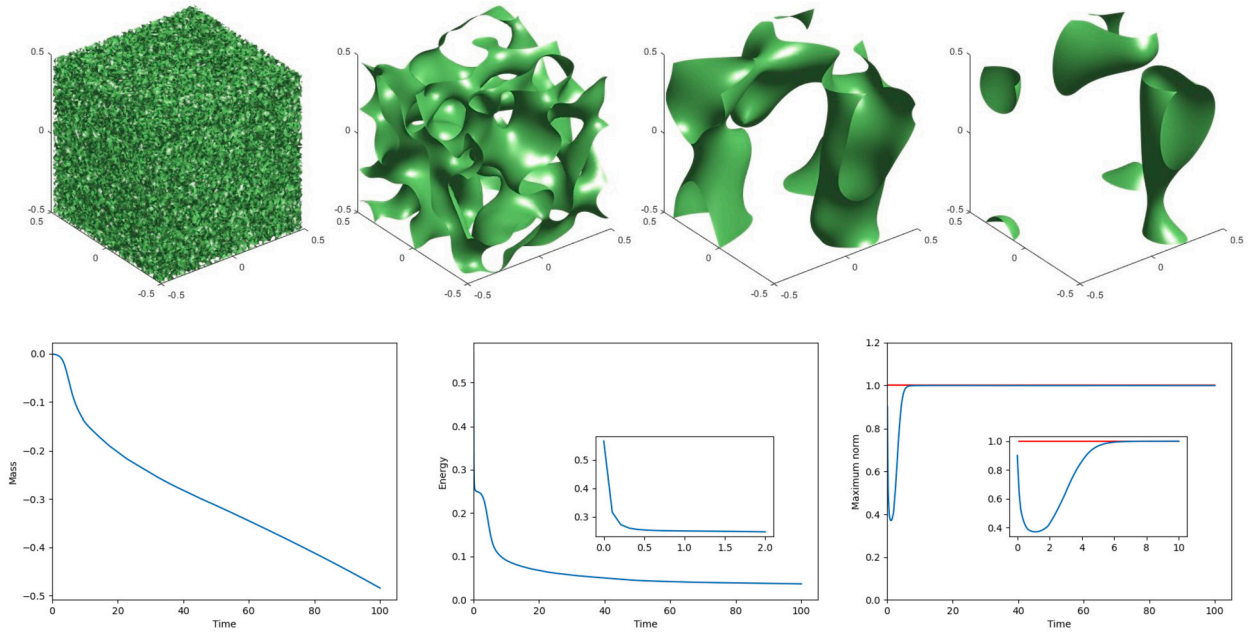


Fig. 27. The ACNN prediction results for the classic AC equation (1) with the double-well potential in the 3D grain coarsening example. Top row: plots of the iso-surfaces (value 0) of the predicted solution at the times t = 0, 10, 50, 100; Bottom row: evolution of the mass (left), the energy (middle), and the maximum norm (right) of the predicted solution.

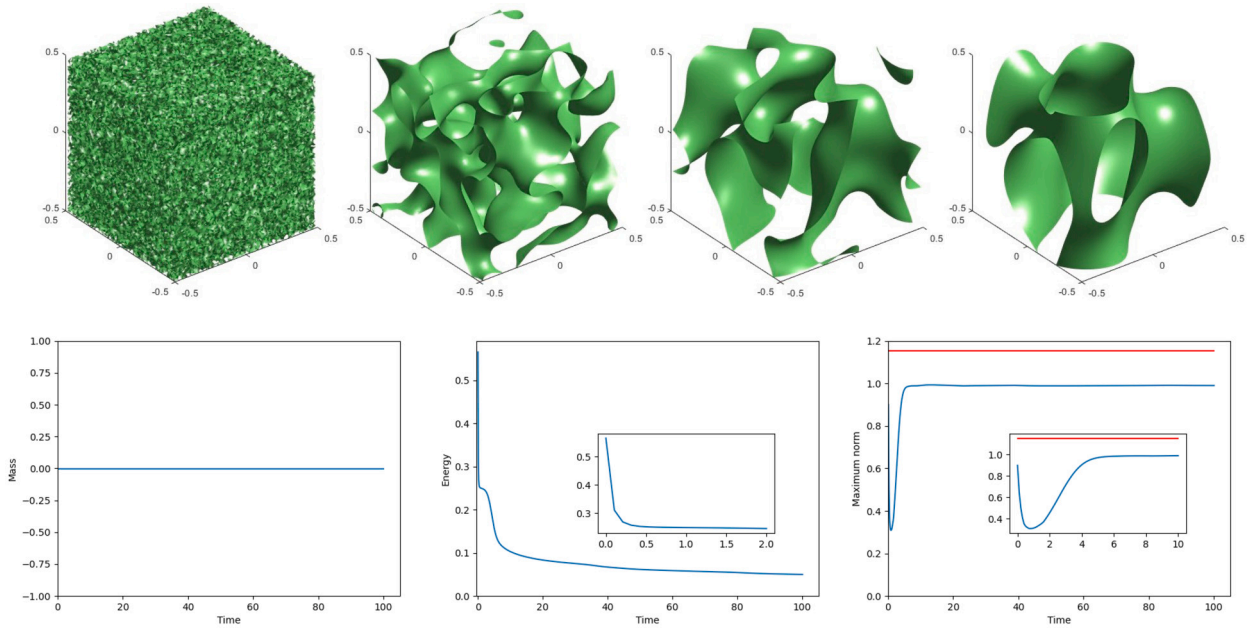


Fig. 28. The mACNN prediction solution for the conservative AC equation (4) with the double-well potential in the 3D grain coarsening example. Top row: plots of the iso-surfaces (value 0) of the predicted solution at the times t = 0, 10, 50, 100; Bottom row: evolution of the mass (left), the energy (middle), and the maximum norm (right) of the predicted solution.

CRediT authorship contribution statement

Yuwei Geng: Methodology, Implementation, Writing-Original draft preparation, Reviewing and Editing. **Yuankai Teng:** Implementation, Reviewing and Editing. **Zhu Wang:** Methodology, Reviewing and Editing. **Lili Ju:** Conceptualization, Methodology, Writing-Original draft preparation, Reviewing and Editing.

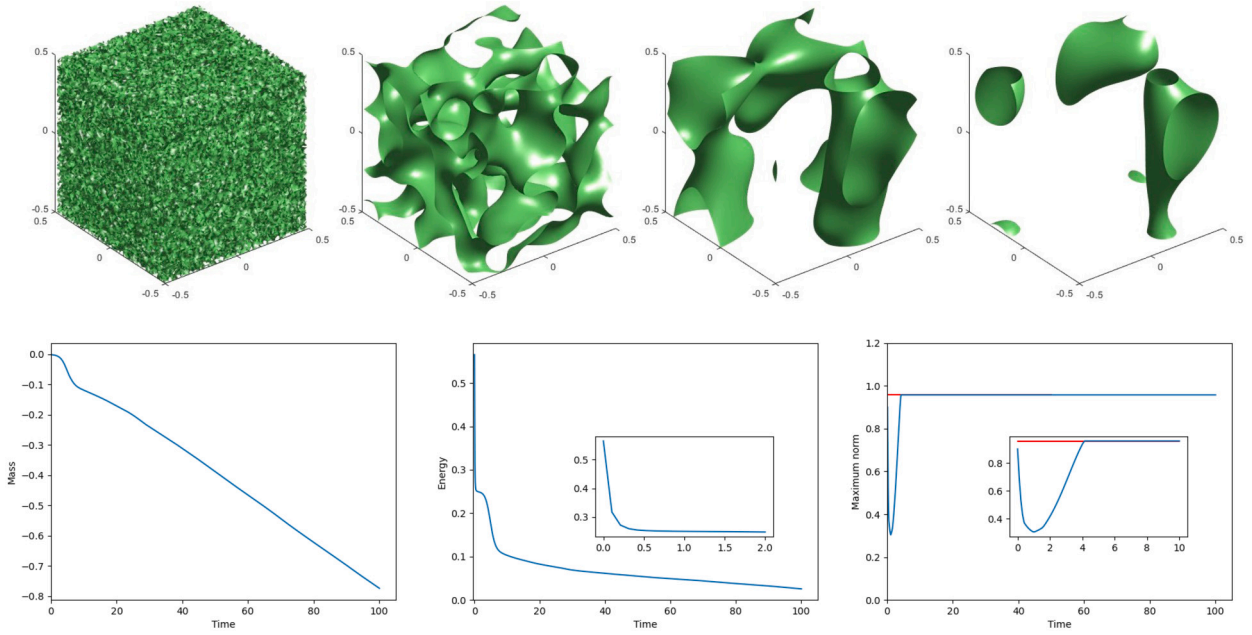


Fig. 29. The ACNN prediction results for the classic AC equation (1) with the Flory-Huggins potential in the 3D grain coarsening example. Top row: plots of the iso-surfaces (value 0) of the predicted solution at the times t = 0, 10, 50, 100; Bottom row: evolution of the mass (left), the energy (middle), and the maximum norm (right) of the predicted solution.

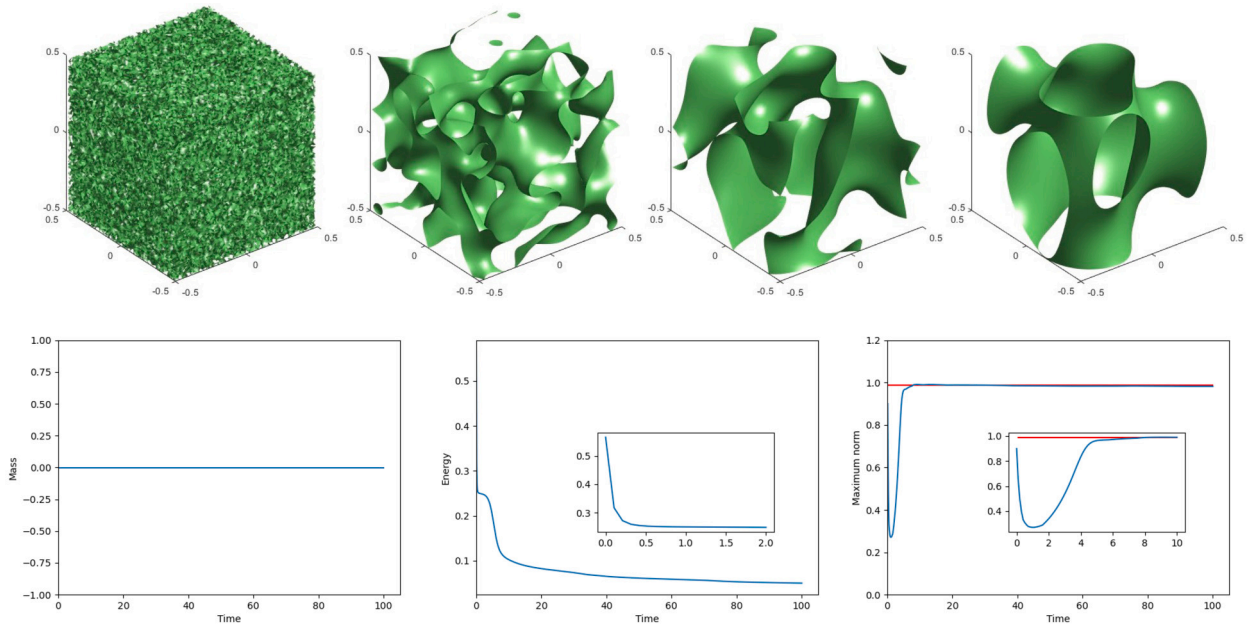


Fig. 30. The mACNN prediction solution for the conservative AC equation (4) with the Flory-Huggins potential in the 3D grain coarsening example. Top row: plots of the iso-surfaces (value 0) of the predicted solution at the times t = 0, 10, 50, 100; Bottom row: evolution of the mass (left), the energy (middle), and the maximum norm (right) of the predicted solution.

Declaration of competing interest

None

Data availability

Data will be made available on request.

Acknowledgements

This work is partially supported by U.S. Department of Energy under grant number DE-SC0022254 and U.S. National Science Foundation under grant numbers DMS-2109633, DMS-2012469, and DMS-2038080.

References

[1] S.M. Allen, J.W. Cahn, A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, Acta Metall. 27 (6) (1979) 1085–1095.
[2] M. Beneš, V. Chalupeckỳ, K. Mikula, Geometrical image segmentation by the Allen–Cahn equation, Appl. Numer. Math. 51 (2–3) (2004) 187–205.
[3] M. Cheng, J.A. Warren, An efficient algorithm for solving the phase field crystal model, J. Comput. Phys. 227 (12) (2008) 6241–6248.
[4] J.A. Dobrosotskaya, A.L. Bertozzi, A wavelet-Laplace variational technique for image deconvolution and inpainting, IEEE Trans. Image Process. 17 (5) (2008) 657–663.
[5] Q. Du, L. Ju, X. Li, Z. Qiao, Maximum principle preserving exponential time differencing schemes for the nonlocal Allen–Cahn equation, SIAM J. Numer. Anal. 57 (2) (2019) 875–898.
[6] Q. Du, L. Ju, X. Li, Z. Qiao, Maximum bound principles for a class of semilinear parabolic equations and exponential time-differencing schemes, SIAM Rev. 63 (2) (2021) 317–359.
[7] X. Feng, A. Prohl, Numerical analysis of the Allen-Cahn equation and approximation for mean curvature flows, Numer. Math. 94 (2003) 33–65.
[8] X. Feng, H.-j. Wu, A posteriori error estimates and an adaptive finite element method for the Allen–Cahn equation and the mean curvature flow, J. Sci. Comput. 24 (2005) 121–146.
[9] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[10] L. Ju, X. Li, Z. Qiao, H. Zhang, Energy stability and error estimates of exponential time differencing schemes for the epitaxial growth model without slope selection, Math. Comput. 87 (312) (2018) 1859–1885.
[11] D.A. Kay, A. Tomasi, Color image segmentation by the vector-valued Allen–Cahn phase-field model: a multigrid solution, IEEE Trans. Image Process. 18 (10) (2009) 2330–2339.
[12] D. Kessler, R.H. Nochetto, A. Schmidt, A posteriori error control for the Allen-Cahn problem: circumventing Gronwall's inequality, ESAIM: Math. Model. Numer. Anal. (Modél. Math. Anal. Numér.) 38 (1) (2004) 129–142.
[13] J. Kim, D. Jeong, S.-D. Yang, Y. Choi, A finite difference method for a conservative Allen–Cahn equation on non-flat surfaces, J. Comput. Phys. 334 (2017) 170–181.
[14] Y. Kim, G. Ryu, Y. Choi, Fast and accurate numerical solution of Allen–Cahn equation, Math. Probl. Eng. 2021 (2021) 1–12.
[15] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
[16] J. Li, L. Ju, Y. Cai, X. Feng, Unconditionally maximum bound principle preserving linear schemes for the conservative Allen–Cahn equation with nonlocal constraint, J. Sci. Comput. 87 (2021) 1–32.
[17] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, arXiv preprint, arXiv:2111.03794, 2021.
[18] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[19] R. Mattey, S. Ghosh, A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations, Comput. Methods Appl. Mech. Eng. 390 (2022) 114474.
[20] M. Raissi, P. Perdikaris, 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 (2019) 686–707.
[21] J. Rubinstein, P. Sternberg, Nonlocal reaction—diffusion equations and nucleation, IMA J. Appl. Math. 48 (3) (1992) 249–264.
[22] D. Shao, W.-J. Rappel, H. Levine, Computational model for cell morphodynamics, Phys. Rev. Lett. 105 (10) (2010) 108104.
[23] J. Shen, C. Wang, X. Wang, S.M. Wise, Second-order convex splitting schemes for gradient flows with Ehrlich–Schwoebel type energy: application to thin film epitaxy, SIAM J. Numer. Anal. 50 (1) (2012) 105–125.
[24] J. Shen, J. Xu, Convergence and error analysis for the scalar auxiliary variable (sav) schemes to gradient flows, SIAM J. Numer. Anal. 56 (5) (2018) 2895–2912.
[25] J. Shen, J. Xu, J. Yang, A new class of efficient and robust energy stable schemes for gradient flows, SIAM Rev. 61 (3) (2019) 474–506.
[26] J. Shen, X. Yang, Numerical approximations of Allen-Cahn and Cahn-Hilliard equations, Discrete Contin. Dyn. Syst. 28 (4) (2010) 1669–1691.
[27] T. Tang, J. Yang, Implicit-explicit scheme for the Allen-Cahn equation preserves the maximum principle, J. Comput. Math. (2016) 451–461.
[28] C. Wang, S.M. Wise, An energy stable and convergent finite-difference scheme for the modified phase field crystal equation, SIAM J. Numer. Anal. 49 (3) (2011) 945–969.
[29] H. Wang, X. Qian, Y. Sun, S. Song, A modified physics informed neural networks for solving the partial differential equation with conservation laws, Available at SSRN 4274376, 2022.
[30] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed deeponets, Sci. Adv. 7 (40) (2021) eabi8605.
[31] A.A. Wheeler, W.J. Boettinger, G.B. McFadden, Phase-field model for isothermal phase transitions in binary alloys, Phys. Rev. A 45 (10) (1992) 7424.
[32] S.M. Wise, C. Wang, J.S. Lowengrub, An energy-stable and convergent finite-difference scheme for the phase field crystal equation, SIAM J. Numer. Anal. 47 (3) (2009) 2269–2288.
[33] X. Wu, G. Van Zwieten, K. Van der Zee, Stabilized second-order convex splitting schemes for Cahn–Hilliard models with application to diffuse-interface tumor-growth models, Int. J. Numer. Methods Biomed. Eng. 30 (2) (2014) 180–203.
[34] Y. Xia, Y. Xu, C.-W. Shu, Application of the local discontinuous Galerkin method for the Allen-Cahn/Cahn-Hilliard system, Commun. Comput. Phys. 5 (2009) 821–835.
[35] H. Xu, J. Chen, F. Ma, Adaptive deep learning approximation for Allen-Cahn equation, in: Computational Science–ICCS 2022: 22nd International Conference, London, UK, June 21–23, 2022, Proceedings, Part IV, Springer, 2022, pp. 271–283.
[36] J. Xu, Y. Li, S. Wu, A. Bousquet, On the stability and accuracy of partially and fully implicit schemes for phase field modeling, Comput. Methods Appl. Mech. Eng. 345 (2019) 826–853.
[37] X. Yang, Linear, first and second-order, unconditionally energy stable numerical schemes for the phase field model of homopolymer blends, J. Comput. Phys. 327 (2016) 294–316.
[38] X. Yang, G.-D. Zhang, Convergence analysis for the invariant energy quadratization (ieq) schemes for solving the Cahn–Hilliard and Allen–Cahn equations with general nonlinear potential, J. Sci. Comput. 82 (2020) 1–28.
[39] J. Zhang, Q. Du, Numerical studies of discrete approximations to the Allen–Cahn equation in the sharp interface limit, SIAM J. Sci. Comput. 31 (4) (2009) 3042–3063.
[40] C.L. Zhao, Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks, Commun. Comput. Phys. 29 (3) (2020).