Discovering Phase Field Models from Image Data with the Pseudo-Spectral Physics Informed Neural Networks

Jia Zhao

Communications on Applied Mathematics and Computation

ISSN 2096-6385
Volume 3
Number 2

Commun. Appl. Math. Comput. (2021)
3:357-369
DOI 10.1007/s42967-020-00105-2

Springer

Communications on Applied Mathematics and Computation (2021) 3:357–369
https://doi.org/10.1007/s42967-020-00105-2

ORIGINAL PAPER

Discovering Phase Field Models from Image Data with the Pseudo-Spectral Physics Informed Neural Networks

Jia Zhao[1]

Received: 5 July 2020 / Revised: 31 October 2020 / Accepted: 27 November 2020 / Published online: 2 April 2021

Abstract
In this paper, we introduce a new deep learning framework for discovering the phase-field models from existing image data. The new framework embraces the approximation power of physics informed neural networks (PINNs) and the computational efficiency of the pseudo-spectral methods, which we named pseudo-spectral PINN or SPINN. Unlike the baseline PINN, the pseudo-spectral PINN has several advantages. First of all, it requires less training data. A minimum of two temporal snapshots with uniform spatial resolution would be adequate. Secondly, it is computationally efficient, as the pseudo-spectral method is used for spatial discretization. Thirdly, it requires less trainable parameters compared with the baseline PINN, which significantly simplifies the training process and potentially assures fewer local minima or saddle points. We illustrate the effectiveness of pseudo-spectral PINN through several numerical examples. The newly proposed pseudo-spectral PINN is rather general, and it can be readily applied to discover other PDE-based models from image data.

Keywords Phase field · Linear scheme · Cahn-Hilliard equation · Physics informed neural network

Mathematics Subject Classification 65M32

1 Introduction

The phase-field models have been widely used for investigating multiphase problems. Among them, the well-known models are the Allen-Cahn (AC) equation and the Cahn-Hilliard (CH) equation [3]. In general, most acknowledged phase field models are thermodynamically consistent, i.e., they satisfy the first and second thermodynamic laws. In scenarios when the temperature is assumed constant, the Helmholtz free energy is non-increasing in time. In isothermal case, the thermodynamically consistent phase-field models can usually be written in a generic gradient flow form

✉ Jia Zhao
jia.zhao@usu.edu

[1] Department of Mathematics and Statistics, Utah State University, Logan, UT 84322, USA

$$\partial_t \phi = \mathcal{G}\frac{\delta E}{\delta \phi} \tag{1}$$

with proper initial values and boundary conditions. Here ϕ represents the state variables (or phase-field variables) and E is the Helmholtz free energy, and $\mathcal{G}$ is a negative semidefinite operator known as the mobility operator. In this paper, we consider the free energy

$$E = \int_\Omega \left[\frac{\varepsilon^2}{2}|\nabla\phi|^2 + F(\phi)\right]\mathrm{d}x \tag{2}$$

with ε a model parameter to control the interfacial thickness. Here the first part in (2) is the conformation entropy and F is the bulk free energy. The choice of the bulk potential $F(\phi)$ depends on the material properties. The double-well potential and the Flory-Huggins free energy are the common choices. We use $f(\phi) = F'(\phi)$ to denote the bulk chemical potential. Specifically, if we choose the mobility operator as $\mathcal{G} = -M$, with $M > 0$ a constant, it ends up with the AC equation

$$\partial_t \phi = M(\varepsilon^2 \Delta\phi - f(\phi)). \tag{3}$$

Similarly, if we pick the mobility operator as $\mathcal{G} = M\Delta$, with $M > 0$ a constant, it gives the CH equation

$$\partial_t \phi = M\Delta(-\varepsilon^2 \Delta\phi + f(\phi)). \tag{4}$$

Though there is a considerable amount of work on solving the phase-field models [6, 7, 17–19, 23], there is less achievement on solving the inverse problem, i.e., discovering the phase-field models from data. In the classical approach, the PDE models are usually derived based on empirical observation, i.e., the PDE models with free parameters are introduced first, and the free parameters are fitted with data afterward. In this paper, we introduce a novel method using the physics informed neural network (PINN) approach, where the model would be directly learned from data. In particular, we assume the bulk potential $F(\phi)$ is unknown, and we will discover it via learning from the existing data using the deep neural network (DNN).

The DNN has been widely used to investigate problems in various fields. Mathematically, the feed-forward neural network could be defined as compositions of nonlinear functions. Given an input $x \in \mathbb{R}^{n_1}$, with n_1 the dimension of the input, and denote the output of the l-th layer with a dimension n_l as $a^{[l]} \in \mathbb{R}^{n_l}$, which is the input for the $(l+1)$ layer. In general, we can define the neural network with L-layers as [9]

$$\begin{cases} a^{[1]} = x \in \mathbb{R}^{n_1}, \\ a^{[k]} = \sigma\left(W^{[k]}a^{[k-1]} + b^{[k]}\right) \in \mathbb{R}^{n_k}, \quad \text{for } k = 2, 3, \cdots, L, \\ a^{[L+1]} = W^{[L+1]}a^{[L]} + b^{[L+1]}, \end{cases} \tag{5}$$

where $W^{[k]} \in \mathbb{R}^{n_k \times n_{k-1}}$ and $b^{[k]} \in \mathbb{R}^{n_k}$ denote the weights and biases at layer k, respectively, σ denotes the activation function, and $a^{[L+1]}$ is the final output of the neural network. Many interesting works have been published in terms of solving or discovery of PDEs [1, 2, 5, 8, 10, 12–15, 20, 21, 24]. Here we briefly point out some relevant work of PDE discovery with DNN or machine learning method. In [15], the authors use the undetermined coefficient approach. They assume the PDE model is a linear combination of a few known terms with

the coefficients unknown. Then they fit the coefficients from data using the PINN. Based on the format of known data, they propose two approaches for calculating the loss function. In [16], the authors use the sparse regression approach. They assume the PDE model is a linear combination of many terms from a predefined dictionary, and use the sparse regularization technique to rule out most of the terms. Thus, after fitting with the data, only a few terms with non-zero coefficients will be left and the model is discovered. In [11], the authors introduce a feed-forward DNN to predict the PDE dynamics and uncover the underlying PDE model (in a black box form) simultaneously.

However, in most of the existing work in literature, there is a strong assumption on the data format and sampling strategies. For instance, most existing PDE discovery strategies request well-sampled data across temporal and spatial domains. Also, they usually assume the collected data at different time slots are from a single time sequence. In reality, especially from lab experiments, the data are usually sampled as snapshots (images) from various time slots, and the images might have limited spatial and temporal resolution. The major goal of this paper is to address the problem of PDE discovery with limited data or image snapshots. We introduce the pseudo-spectral PINN method that provides a handy approach to deal with data that are provided as image snapshots. Though we only focus on the discovery of phase-field models, the proposed approach is general that can be easily applied to some other types of PDE models.

The rest of this paper is organized as follows. In Sect. 2, we introduce some notations and set up the phase-field PDE discovery problem. In Sect. 3, we introduce the SPINNs by introducing the deep network structures and the various definitions of the loss function. Afterward, several examples are shown to demonstrate the approximation power of the pseudo-spectral PINN in Sect. 4. A brief conclusion is drawn in the last section.

2 Problem Setup

For simplicity of notations, we introduce our idea with a generic example. Consider a rectangular domain $\Omega = [0, L_x] \times [0, L_y]$ with L_x and L_y the lengths in each direction. Recall from (1), we use $\phi(\mathbf{x}, t)$ to denote the state variables, where $\mathbf{x} \in \Omega \subset \mathbb{R}^d$ are the spatial coordinates and $t \in (0, T]$ denotes the time. Given the PDE problem

$$\partial_t \phi = \mathcal{G}\Big[g(\phi, \nabla\phi, \Delta\phi) + f(\phi)\Big], \quad (\mathbf{x}, t) \in \Omega \times (0, T] \tag{6}$$

with periodic boundary condition, where operators $\mathcal{G}$ and g are known, but operator f is unknown. The major goal of this paper is to identify the operator/functional f, thus discover the PDE model in (6). The baseline PINN [15] would require a rather amount of data pairs $\{(\mathbf{x}_i, t_i, \phi(\mathbf{x}_i, t_i))\}_{i=1}^N$, with $(\mathbf{x}_i, t_i) \in \Omega \times (0, T]$ properly sampled from the spatial-temporal domain. Also, it requires all the data are from a single time sequence, given that a DNN $\mathcal{N}(\mathbf{x}, t)$ is needed to approximate $\phi(\mathbf{x}, t)$ as an aiding neural network for the PDE discovery. However, this is usually not the case in reality. In practice, the data are usually collected as image snapshots at various time slots from different time sequences. For instance, for certain experiments, the experiments will repeat multiple times from which the data will be collected. Also, in each experiment, a picture is taken at t_0, and another picture is taken at $t_1 = t_0 + \delta t$, where δt might be large due to constraints of resources.

To clearly describe the problem, we need to introduce some notations. Let N_x, N_y be two positive even integers. The spatial domain $\Omega = [0, L_x] \times [0, L_y]$ is uniformly partitioned with mesh size $h_x = L_x/N_x, h_y = L_y/N_y$ and

$$\Omega_h = \left\{(x_j, y_k) | x_j = jh_x, y_k = kh_y,\ 0 \leqslant j \leqslant N_x - 1, 0 \leqslant k \leqslant N_y - 1\right\}.$$

In order to derive the algorithm conveniently, we denote the discrete gradient operator and the discrete Laplace operator

$$\nabla_h, \quad \Delta_h = \nabla_h \cdot \nabla_h,$$

following our previous work [4]. Let $V_h = \{u | u = [u_{j,k}], (x_j, y_k) \in \Omega_h, 0 \leqslant j \leqslant N_x - 1, 0 \leqslant k \leqslant N_y - 1\}$ be the space of grid functions on Ω_h. By discretizing the PDE problem (6) using the pseudo-spectral method in space, it can be written as

$$\partial_t \Phi_{ij} = \mathcal{G}_h \left[g(\Phi_{ij}, \nabla_h \Phi_{ij}, \Delta_h \Phi_{ij}) + f(\Phi_{ij}) \right], \quad i, j = 1, 2, \cdots, M, \quad t \in (0, T], \tag{7}$$

where $\mathcal{G}_h$ is the spatially discretized mobility operator. Here we use $\Phi \in V_h$ to denote the discrete function values of ϕ on Ω_h. The periodic boundary condition is assumed in this paper. If other type of boundary conditions is considered, the finite-difference or finite-element method for spatial discretization might be more proper.

Under such notations, the collected data is in the form as

$$\{(\Phi_i^{(1)}, \Phi_i^{(2)}, \delta_i)\}_{i=1}^N \subset V_h \times V_h \times \mathbb{R}^+, \tag{8}$$

where $\Phi_i^{(1)} \in V_h$ and $\Phi_i^{(2)} \in V_h$ are two snapshots with a time lag $\delta_i > 0$ between the two states, and N is the total number of snapshot pairs. The goal in this paper is to discover the bulk function f in the phase field models, with the data collected in the form of (8), i.e.,

$$\text{find the operator } f \text{ in (6) from the data } \{(\Phi_i^{(1)}, \Phi_i^{(2)}, \delta_i)\}_{i=1}^N \subset V_h \times V_h \times \mathbb{R}^+. \tag{9}$$

We emphasis that δ_i is not required to be constant, removing the assumption in [22]. Also, different data pairs $(\Phi_i^{(1)}, \Phi_i^{(2)})$ and $(\Phi_j^{(1)}, \Phi_j^{(2)})$ $(i \neq j)$ are not necessarily from the same single time sequence, removing the requirement of the baseline PINN [15].

3 Pseudo-Spectral Physics Informed Neural Networks

There are mainly two components (or ingredients) in the DNN method. First of all, we shall define what the neural network is meant to approximate along with its structures. Secondly, we shall define the loss function, which is enforced with known physics.

3.1 Neural Network Structure

The generic way to solve the problem (9) is by introducing a DNN

$$\mathcal{N}_f : \phi \to \mathcal{N}_f(\phi; \Theta) \tag{10}$$

to approximate $f(\phi)$, where Θ represents the free parameters. In this paper, we assume f is only a function of ϕ for simplicity. Notice the idea in this paper applies easily to cases where f is a function of ϕ and its derivatives, saying $f := f(\phi, \nabla\phi, \Delta\phi)$.

Notice, in the baseline PINN, an aiding neural network $\mathcal{N} : (\mathbf{x}, t) \to \mathbb{R}$ is introduced to approximate $\phi(\mathbf{x}, t)$, which is computationally expensive and applies only to a single time sequence. It will be apparent that applying the PINNs on the semi-discrete problem (7) will have several advantages.

3.2 Loss Functions

Next, we define the loss function L. Our idea of defining the loss functions is inspired by the linear numerical methods for solving PDEs.

3.2.1 Loss Functions Inspired by Stabilized Linear Schemes

We split g in (7) as $g(\Phi, \nabla_h\Phi, \Delta_h\Phi) = L_g(\Phi) + N_g(\Phi)$, where L_g is the linear operator and N_g is the rest, i.e.,

$$\partial_t\Phi = \mathcal{G}_h\Big[L_g(\Phi) + N_g(\Phi) + f(\Phi)\Big]. \tag{11}$$

To solve it numerically in the time interval $[t_i, t_i + \delta_i]$, we can propose the stabilized scheme

$$\frac{1}{\delta_i}(\Phi_{t_i+\delta_i} - \Phi_{t_i}) = \mathcal{G}_h\Big[L_g(\Phi_{t_i+\delta_i}) + N_g(\Phi_{t_i}) + f(\Phi_{t_i}) + C(\Phi_{t_i+\delta_i} - \Phi_{t_i})\Big], \tag{12}$$

where C is a stabilizing operator, which could be chosen as

$$C = \sum_{i=0}^{2}(-1)^i C_i(\Delta_h)^{2i}, \tag{13}$$

with C_i's constants. Then, we have

$$\Phi_{t_i+\delta_i} = (1 - \delta_i\mathcal{G}_h(C + L_g))^{-1}\Big\{\Phi_{t_i} + \delta_i\mathcal{G}_h\Big[N_g(\Phi_{t_i}) + f(\Phi_{t_i}) - C\Phi_{t_i}\Big]\Big\}. \tag{14}$$

Notice the scheme (14) is first-order accurate in time. When δ_i is small enough, the expression for $\Phi_{t_i+\delta_i}$ is an accurate approximation to $\Phi(t_i + \delta_i)$. Inspired by this, we can introduce our linear SPINN loss function, to discover (6) from the data (8).

Definition 1 (Linear SPINN loss function) Given $(\Phi_i^{(1)}, \delta_i)$, we can approximate $\Phi_i^{(2)}$ via $\mathcal{N}_R(\Phi_i^{(1)}, \delta_i)$, which is defined by

$$\mathcal{N}_R : (\Phi_i^{(1)}, \delta_i) \to (1 - \delta_i(\mathcal{G}_h(C + L_g)))^{-1}\Big\{\Phi_i^{(1)} + \delta_i\mathcal{G}_h\Big[N_g(\Phi_i^{(1)}) + \mathcal{N}_f(\Phi_i^{(1)};\Theta) - C\Phi_i^{(1)}\Big]\Big\}, \tag{15}$$

where C_i's in C are hyper-parameters. Hence, the loss function is defined as

$$L(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \|\Phi_i^{(2)} - \mathcal{N}_R(\Phi_i^{(1)}, \delta_i; \Theta)\|_2^2. \tag{16}$$

When the time step δ_i is small, by minimizing $L(\Theta)$, we can identify $\mathcal{N}_f$, which in turn discover the PDE problem in (6). For certain cases, δ_i might be large. A single step marching scheme might be inaccurate. We thus introduce a loss function that is based on a more general recursive linear SPINN.

Definition 2 (Recursive linear SPINN loss function) Given $(\Phi_i^{(1)}, \delta_i)$, we can approximate $\Phi_i^{(2)}$ via a mapping $\mathcal{N}_{R_K} : (\Phi_i^{(1)}, \delta_i) \to R_K$, where R_K is defined recursively as

$$\begin{cases} R_0 = \Phi_i^{(1)}, \\ R_j = \left(1 - \dfrac{\delta_i}{K}\mathcal{G}_h(C + L_g)\right)^{-1}\left(R_{j-1} + \dfrac{\delta_i}{K}\mathcal{G}_h(N_g(R_{j-1}) + \mathcal{N}_f(R_{j-1}) - CR_{j-1})\right), \\ j = 1, 2, \cdots, K, \end{cases} \tag{17}$$

where K and C_i's are hyper-parameters. And the loss function is defined as

$$L(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \|\Phi_i^{(2)} - \mathcal{N}_{R_K}(\Phi_i^{(1)}, \delta_i; \Theta)\|_2^2. \tag{18}$$

Remark 1 Similarly, we can design the neural network inspired by second-order or higher-order numerical schemes. For instance, consider the following predictor-corrector second-order scheme. To solve it numerically in the time interval $[t_i, t_i + \delta_i]$, we introduce the stabilized second-order scheme in two steps.

- First of all, we can obtain $\hat{\Phi}_{t_i + \frac{\delta_i}{2}}$ via

$$\frac{\hat{\Phi}_{t_i+\frac{\delta_i}{2}} - \Phi_{t_i}}{\delta_i/2} = \mathcal{G}_h\left[L_g(\hat{\Phi}_{t_i+\frac{\delta_i}{2}}) + N_g(\Phi_{t_i}) + f(\Phi_{t_i}) + C(\hat{\Phi}_{t_i+\frac{\delta_i}{2}} - \Phi_{t_i})\right]. \tag{19}$$

- Next, we can obtain $\Phi_{t_i+\delta_i}$ via solving

$$\begin{aligned} \frac{\Phi_{t_i+\delta_i} - \Phi_{t_i}}{\delta_i} =& \mathcal{G}_h\left[L_g\left(\frac{\Phi_{t_i+\delta_i} + \Phi_{t_i}}{2}\right) + N_g(\hat{\Phi}_{t_i+\frac{\delta_i}{2}}) + f(\hat{\Phi}_{t_i+\frac{\delta_i}{2}})\right. \\ &\left. + C\left(\frac{\Phi_{t_i+\delta_i} + \Phi_{t_i}}{2} - \hat{\Phi}_{t_i+\frac{\delta_i}{2}}\right)\right], \end{aligned} \tag{20}$$

Therefore, if we definite the neural network $\mathcal{N}_f : \phi \to \mathcal{N}_f(\phi; \Theta)$, once given $(\Phi_i^{(1)}, \delta_i)$, we can approximate $\Phi_i^{(2)}$ via the mapping defined as

$$\begin{aligned} \mathcal{N}_R : (\Phi_i^{(1)}, \delta_i) \to& \left(1 - \tfrac{\delta_i}{2}\mathcal{G}_h(C + L_g)\right)^{-1}\left(\Phi_i^{(1)}\right. \\ &\left. + \delta_i\mathcal{G}_h\left(L_g\left(\frac{\Phi_i^{(1)}}{2}\right) + N_g(\hat{\Phi}_{i+\frac{\delta_i}{2}}) + \mathcal{N}_f(\hat{\Phi}_{i+\frac{\delta_i}{2}}; \Theta) + C\left(\frac{\Phi_i^{(1)}}{2} - \hat{\Phi}_{i+\frac{\delta_i}{2}}\right)\right)\right), \end{aligned} \tag{21}$$

where $\hat{\Phi}_{i+\frac{\delta_i}{2}}$ is defined by

$$\hat{\Phi}_{i+\frac{\delta_i}{2}} = \left(1 - \frac{\delta_i}{2}\mathcal{G}_h(C + L_g)\right)^{-1}\left(\Phi_i^{(1)} + \frac{\delta_i}{2}\mathcal{G}_h\left[N_g(\Phi_i^{(1)}) + \mathcal{N}_f(\Phi_i^{(1)};\Theta) - C\Phi_i^{(1)}\right]\right). \tag{22}$$

Then the loss function could be defined similarly. Some other ideas, such as exponential time integration can also be utilized for designing neural network loss functions. For brevity, these ideas will not be pursued in this paper. Interested readers are encouraged to explore these interesting topics.

3.2.2 Loss Functions Inspired by Runge-Kutta Method

As an analogy, we can mimic the Runge-Kutta method for solving (6) to design neural networks. Inspired by the idea of a four-stage explicit Runge-Kutta method for the time discretization, we can introduce the following loss function.

Definition 3 (Four-stage explicit Runge-Kutta SPINN loss function) Given $(\Phi_i^{(1)}, \delta_i)$, we can approximate $\Phi_i^{(2)}$ by the mapping $\mathcal{N}_R(\Phi_i^{(1)}, \delta_i)$ defined by the four-stage method

$$\mathcal{N}_R : (\Phi_i^{(1)}, \delta_i) \to \Phi_i^{(1)} + \frac{\delta_i}{6}(K_1 + 2K_2 + 2K_3 + K_4), \tag{23}$$

where

$$\begin{cases} K_1 = \mathcal{G}_h\left[g(\Phi_i^{(1)}) + \mathcal{N}_f(\Phi_i^{(1)};\Theta)\right], \\ K_2 = \mathcal{G}_h\left[g\left(\Phi_i^{(1)} + \frac{\delta_i}{2}K_1\right) + \mathcal{N}_f\left(\Phi_i^{(1)} + \frac{\delta_i}{2}K_1;\Theta\right)\right], \\ K_3 = \mathcal{G}_h\left[g\left(\Phi_i^{(1)} + \frac{\delta_i}{2}K_2\right) + \mathcal{N}_f\left(\Phi_i^{(1)} + \frac{\delta_i}{2}K_2;\Theta\right)\right], \\ K_4 = \mathcal{G}_h\left[g(\Phi_i^{(1)} + \delta_i K_3) + N_f(\Phi_i^{(1)} + \delta_i K_3;\Theta)\right]. \end{cases} \tag{24}$$

Then the loss function can be defined as

$$L(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \|\Phi_i^{(2)} - \mathcal{N}_R(\Phi_i^{(1)}, \delta_i)\|_2^2. \tag{25}$$

Similarly, when the time step δ_i is large, we can define the loss function via the explicit Runge-Kutta SPINN recursively as below.

Definition 4 (Recursive four-stage explicit Runge-Kutta SPINN loss function) Given the data $(\Phi_i^{(1)}, \delta_i)$, we can approximate $\Phi_i^{(2)}$ via the K-step recursive mapping denoted as $\mathcal{N}_{R_K} : (\Phi_i^{(1)}, \delta_i) \to R_K$, where

$$\begin{cases} R_0 = \Phi_i^{(1)}, \\ R_j = R_{j-1} + \frac{\delta_i}{6K}(K_1^{j-1} + 2K_2^{j-1} + 2K_3^{j-1} + K_4^{j-1}), j = 1, 2, \cdots, K, \\ K_1^{j-1} = \mathcal{G}_h\Big[g(R_{j-1}) + \mathcal{N}_f(R_{j-1};\Theta)\Big], \\ K_2^{j-1} = \mathcal{G}_h\Big[g\Big(R_{j-1} + \frac{\delta_i}{2K}K_1^{j-1}\Big) + \mathcal{N}_f\Big(R_{j-1} + \frac{\delta_i}{2K}K_1^{j-1};\Theta\Big)\Big], \\ K_3^{j-1} = \mathcal{G}_h\Big[g\Big(R_{j-1} + \frac{\delta_i}{2K}K_2^{j-1}\Big) + \mathcal{N}_f\Big(R_{j-1} + \frac{\delta_i}{2K}K_2^{j-1};\Theta\Big)\Big], \\ K_4^{j-1} = \mathcal{G}_h\Big[g\Big(R_{j-1} + \frac{\delta_i}{K}K_3^{j-1}\Big) + N_f\Big(R_{j-1} + \frac{\delta_i}{K}K_3^{j-1};\Theta\Big)\Big]. \end{cases} \tag{26}$$

And the loss function is defined by

$$L(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \|\Phi_i^{(2)} - \mathcal{N}_{R_K}(\Phi_i^{(1)}, \delta_i)\|_2^2. \tag{27}$$

Remark 2 When the time step is even larger, one can use the implicit Runge-Kutta method for time discretization. However, given the intermediate stages are unknown, we need to introduce an extra neural network to approximate it:

$$\mathcal{N}_q : (\Phi_i^{(1)}, \delta_i) \to (\mathcal{N}_{c_1}(\Phi_i^{(1)}, \delta_i), \mathcal{N}_{c_2}(\Phi_i^{(1)}, \delta_i), \cdots, \mathcal{N}_{c_q}(\Phi_i^{(1)}, \delta_i)), \tag{28}$$

where $\mathcal{N}_{c_j}(\Phi_i^{(1)}, \delta_i)$ is to approximate $\Phi(t_i + c_j\delta_i)$ for a q-stage Runge-Kutta method. By following the idea in [15], we can define the loss function

$$\begin{aligned} L(\Theta) =& \frac{1}{qN}\sum_{j=1}^{q}\sum_{i=1}^{N}\Big\|\Phi_i^{(1)} - \Big[\mathcal{N}_{c_j}(\Phi_i^{(1)}, \delta_i) + \delta_i\sum_{k=1}^{q} a_{jk}\mathcal{N}_{c_k}(\Phi_i^{(1)}, \delta_i)\Big]\Big\|_2^2 \\ &+ \frac{1}{qN}\sum_{j=1}^{q}\sum_{i=1}^{N}\Big\|\Phi_i^{(2)} - \Big[\mathcal{N}_{c_j}(\Phi_i^{(1)}, \delta_i) + \delta_i\sum_{k=1}^{q}(a_{jk} - b_k)\mathcal{N}_{c_k}(\Phi_i^{(1)}, \delta_i)\Big]\Big\|_2^2. \end{aligned} \tag{29}$$

For such a case, it requires expensive computational costs, and we will not investigate it in the current paper.

4 Numerical Examples

Next, we investigate the proposed SPINN approach with several examples, i.e., to identify the phase-field models from data (image snapshots).

Recall that we assume $f(\phi)$ in (6) is unknown, and the goal is to identify it via the existing data in the form of (8). As explained, we define a neural network $\mathcal{N}_f : \phi \to \mathcal{N}_f(\phi;\Theta)$ to approximate f and use the loss functions as defined in the previous section. In the rest of this paper, we assume $\mathcal{N}_f$ a feed-forward neural network with 2-hidden layers and each hidden layer having 20 neurons. The tanh activation function is applied in both hidden layers. During the training process, the Adam method with default learning rate is used for

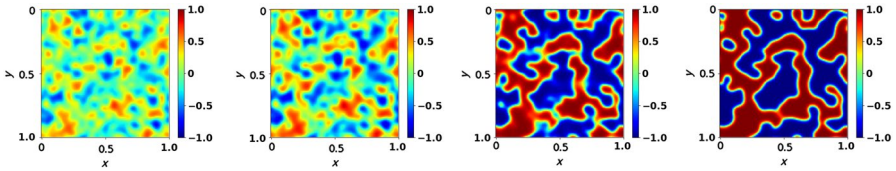


Fig. 1 Snapshots of the solution Φ for the AC equation at various time slots $t = 0.325, 0.375, 0.575, 0.825$

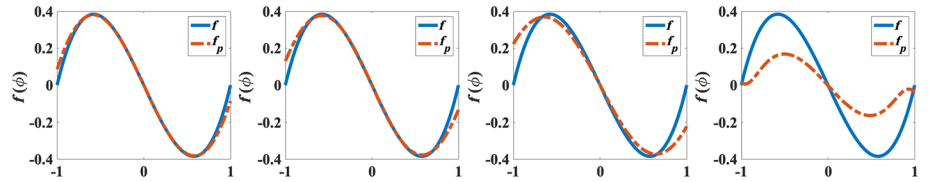


Fig. 2 Predicted bulk function f_p using the Runge-Kutta SPINN loss function vs. the accurate bulk function f with $\delta = 0.05, 0.1, 0.25, 0.5$

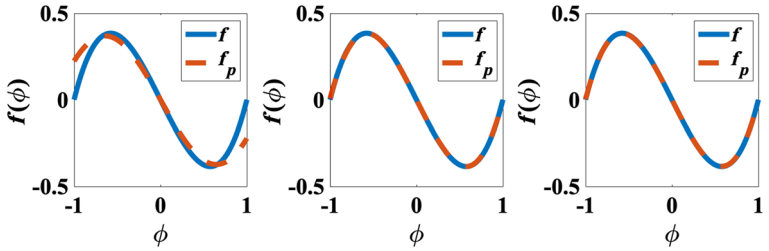


Fig. 3 Predicted bulk function f_p using the recursive Runge-Kutta SPINN loss function vs. the accurate bulk function f, where $K = 1, 50, 100$. This figure indicates that it gives an accurate prediction of f with sufficient recursive stages

10 000 training iterations, followed by an L-BFGS-B optimization training process. The algorithms are implemented with Tensorflow.

For simplicity of discussion, we chosen the domain $\Omega = [-1, 1]^2$, and choose $N_x = N_y = 128$ in Ω_h, i.e., the collected data $\Phi_i^{(1)}, \Phi_i^{(2)}$ are matrices in $\mathbb{R}^{128,128}$ for all the examples in this section. And we solve the PDE first with the high-order-accurate scheme with uniform time steps and uniform spatial discretization. The numerical solutions are randomly sampled at different time slots as training data to inversely discover the bulk function f. Given the free energy in (2), we get $g(\phi) = -\epsilon^2\Delta\phi$, and $L_g(\Phi) = -\epsilon^2\Delta_h\Phi$. And we will chose $C = 2$ for the AC equation, and $C = -2\Delta_h$ for the CH equation.

Example 1 In the first example, we generate data by solving the AC equation in (3) with $F(\phi) = \frac{1}{4}(1-\phi^2)^2$, which means $f(\phi) = \phi^3 - \phi$. The parameters used are $\epsilon = 0.02$ and $M = 10$, with the initial condition $\phi(\mathbf{x}, t = 0) = 0.25\text{rand}(\mathbf{x})$. Here rand() generates random numbers between $[-1, 1]$. Some snapshots of ϕ at various time slots are shown in Fig. 1.

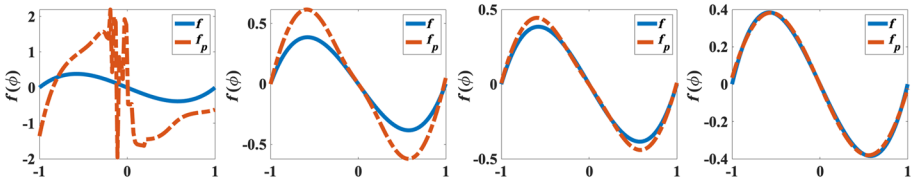
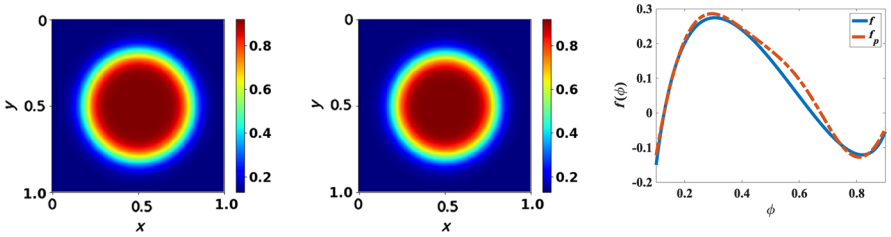


Fig. 4 Discovered bulk function f_p vs. the accurate bulk function f with $K = 1{,}10,\ 50{,}100$, using the recursive linear SPINN approach. It shows that the accuracy of the recursive linear SPINN improves with K increasing



(a) Snapshots of ϕ at $t = 0.325, 0.375$ **(b)** Predicted f

Fig. 5 Predicted bulk function f_p vs. the accurate bulk function f for the AC equation with Flory-Huggins bulk free energy using the recursive linear SPINN with $K = 10$. **a** snapshots of the solution ϕ for the Allen-Cahn equation at time $t = 0.325, 0.375$. **b** the learned bulk function from the data in (**a**)

First of all, we test out the Runge-Kutta SPINN approach. We choose $N = 1$, i.e., use only a single data pair $(\Phi^{(1)}, \Phi^{(2)}, \delta)$ to train the neural network. We randomly choose $\Phi^{(1)}$, and study how the size of δ would affect the learned result. One experiment results with $\Phi^{(1)}$ chosen at $t = 0.325$ and $\delta = 0.05, 0.1, 0.25, 0.5$ are summarized in Fig. 2. We observe that when the two snapshots are close, i.e., δ is small, the Runge-Kutta SPINN approach can accurately learn the bulk function f. However, when the time step δ is large, its accuracy drops.

To overcome the inaccuracy when δ is large, we utilize the recursive Runge-Kutta SPINN approach. Here we fix the time step $\delta = 0.25$, and test the accuracy by using different recursive stage $K = 1, 20, 50, 100$. The results are summarized in Fig. 3. We observe that the accuracy improves as the recursive stage K increases. The bulk function f can be learned accurately when sufficient stages are used.

Example 2 However, when the time step δ is large enough, the recursive Runge-Kutta SPINN approach will not provide an accurate approximation to f. For instance, with the time step $\delta = 0.5$, the recursive Runge-Kutta SPINN approach fails. Meanwhile, the linear SPINN approach shows superior accuracy. This can be intuitively explained by the stability of numerical schemes. Even though the Runge-Kutta method is high-order accurate in time, it is conditionally stable concerning the time step. Therefore, when the time step δ is relatively large, the Runge-Kutta method loses its stability. In the meanwhile, the stabilized linear semi-implicit scheme is energy stable under mild conditions. Therefore, a larger time step can be used while preserving the stability.

As an example, we use a single data pair $(\Phi^{(1)}, \Phi^{(2)}, \delta)$ with a fixed time step $\delta = 0.5$. We vary the recursive stage K for the recursive linear SPINN, and the results are summarized

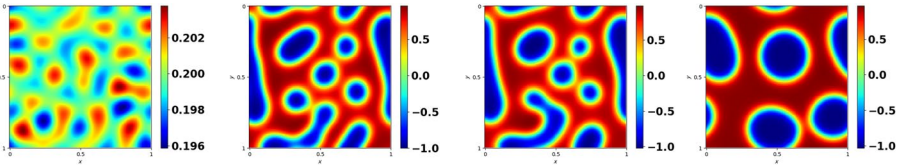


Fig. 6 Snapshots of the solution Φ for the CH equation at various time slots $t = 0.05, 0.325, 0.375, 1$

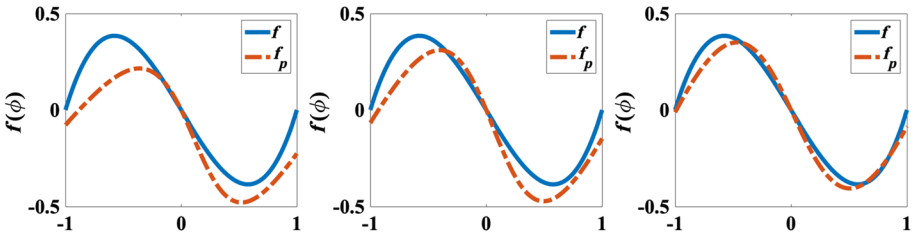


Fig. 7 Predicted bulk function f_p vs. the accurate bulk function f for the CH equation with various data pairs $N = 1, 5, 10$

in Fig. 4. We observe that, even when the time step δ is large, the recursive linear SPINN approach still provides accurate approximation to f, so long as the recursive stage K is large enough.

We remark that the training strategy and the quality of training data might also be factors for the approximation accuracy, which we will not pursue in detail. Interested readers are strongly encouraged to explore.

Example 3 In the next example, we increase the problem complexity to identify a highly nonlinear bulk function. In details, we get the data by solving an AC equation with the Flory-Huggins free energy $F(\phi) = \phi \ln(\phi) + 0.5(1-\phi)\ln(1-\phi) + 2\phi(1-\phi)$, which means the bulk function $f = \ln\phi - 0.5\ln(1-\phi) + 2.5 - 4\phi$. The parameters used are $M = 10$ and $\varepsilon = 0.1$, along with the initial condition $\phi(\mathbf{x}, t=0) = \frac{1}{2}(1 + \tanh\frac{0.8-\sqrt{x^2+y^2}}{\sqrt{2}\varepsilon})$. We randomly sample two snapshots with $\delta t = 0.05$, and train the neural network. An example of using two snapshots at $t = 0.325, 0.375$ is shown in Fig. 5, where the two snapshots are shown in Fig. 5a, and the predicted function is shown in Fig. 5b. We observe the linear SPINN approach can learn the bulk function f from only two images accurately.

Example 4 In the last example, we use the linear SPINN approach to discover the bulk function f from the solution snapshots of the CH equation in (4). Here the data is obtained by solving the CH equation with $F(\phi) = \frac{1}{4}(\phi^2 - 1)^2$, i.e., $f(\phi) = \phi^3 - \phi$. We use $M = 1$, $\varepsilon = 0.05$, and initial condition $\phi(t=0) = 0.2 + 0.001\text{rand}(\mathbf{x})$. Here rand() generates random numbers between $[-1, 1]$.

Notice that the bulk free energy F for a given CH equation is not unique (that is off by a constant), i.e., if F is the bulk free energy, $F + C_0$ is also the bulk free energy, with C_0

a constant. In this example, we enforce $F(0) = 0$. Hence, for the loss function, we add an extra term $\lambda_0 \|\mathcal{N}_f(0)\|^2$, to enforce the uniqueness of $\mathcal{N}_f$, with λ_0 a hyper-parameter, acting as a weight for this term. Here we choose $\lambda_0 = 10^3$.

Some temporal snapshots for Φ are summarized in Fig. 6. We randomly choose data $\{(\Phi_i^{(1)}, \Phi_i^{(2)}, \delta_i)\}_{i=1}^N$, with $N = 1, 5, 10$ data points and a fixed time step $\delta_i = 0.05$. The learned result f_p is summarized in Fig. 7. We observe that the predicted bulk function has improved accuracy with more data used to train the neural network.

5 Conclusion

In this paper, we introduce pseudo-spectral PINNs to discover the bulk function in the phase-field models. This newly proposed method well fits the data collection strategy in practice, i.e., taking snapshots/images at various time slots with large time lags. The definition of loss functions is inspired by classical numerical algorithms for PDEs. The effectiveness of the proposed pseudo-spectral PINN, or SPINN, has been verified by identifying the bulk function f of several phase-field models. The idea of pseudo-spectral PINN introduced in this paper is rather general, and it can be applied to discover other PDE models from image data, which will be investigated in our later research projects.

Acknowledgements Jia Zhao would like to acknowledge the support from NSF DMS-1816783 and NVIDIA Corporation for their donation of a Quadro P6000 GPU for conducting some of the numerical simulations in this paper.

Compliance with Ethical Standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Berg, J., Nystrom, K.: A unified deep artificial neural network approach to partial differential equations in complex geometries. Neurocomputing **317**, 28–41 (2018)
2. Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. **113**(15), 3932–3937 (2016)
3. Cahn, J.W., Hilliard, J.E.: Free energy of a nonuniform system. I. Interfacial free energy. J. Chem. Phys. **28**, 258–267 (1958)
4. Chen, L., Zhao, J., Gong, Y.: A novel second-order scheme for the molecular beam epitaxy model with slope selection. Commun. Comput. Phys. **4**(25), 1024–1044 (2019)
5. E, Weinan., Yu, B.: The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**, 1–12 (2018)
6. Guillen-Gonzalez, F., Tierra, G.: Second order schemes and time-step adaptivity for Allen-Cahn and Cahn-Hilliard models. Comput. Math. Appl. **68**(8), 821–846 (2014)
7. Han, D., Wang, X.: A second order in time uniquely solvable unconditionally stable numerical schemes for Cahn-Hilliard-Navier-Stokes equation. J. Comput. Phys. **290**(1), 139–156 (2015)
8. Han, J., Jentzen, A., E, Weinan.: Solving high-dimensional partial differential equations using deep learning. Proc. Natl. Acad. Sci. **115**(34), 8505–8510 (2018)
9. Higham, C., Higham, D.: Deep learning: an introduction for applied mathematicians. SIAM Rev. **61**(4), 860–891 (2019)
10. Li, B., Tang, S., Yu, H.: Better approximations of high dimensional smooth functions by deep neural networks with rectified power units. Commun. Comput. Phys. **27**, 379–411 (2020)

11. Long, Z., Lu, Y., Ma, X., Dong, B.: PDE-net: learning PDEs from data. Proceedings of the 35th International Conference on Machine Learning **80**, 3208–3216 (2018)
12. Lu, L., Meng, X., Mao, Z., Karniadakis, G.E.: DeepXDE: a deep learning library for solving differential equations. arXiv: 1907.04502 (2019)
13. Qin, T., Wu, K., Xiu, D.: Data driven governing equations approximation using deep neural networks. J. Comput. Phys. **395**, 620–635 (2019)
14. Raissi, M.: Deep hidden physics models: deep learning of nonlinear partial differential equations. J. Mach. Learn. Res. **19**, 1–24 (2018)
15. Raissi, M., Perdikaris, P., Karniadakis, G.E.: 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)
16. Rudy, S., Brunton, S.L., Proctor, J.L., Kutz, J.N.: Data-driven discovery of partial differential equations. Sci. Adv. **3**(4), e1602614 (2017)
17. Shen, J., Yang, X.: Numerical approximations of Allen-Cahn and Cahn-Hilliard equations. Discrete and Continuous Dynamical Systems **28**(4), 1669–1691 (2010)
18. Wang, C., Wang, X., Wise, S.: Unconditionally stable schemes for equations of thin film epitaxy. Discrete and Continuous Dynamical Systems **28**(1), 405–423 (2010)
19. Wang, C., Wise, S.: An energy stable and convergent finite-difference scheme for the modified phase field crystal equation. SIAM J. Numer. Anal. **49**(3), 945–969 (2011)
20. Wang, Y., Lin, C.: Runge-Kutta neural network for identification of dynamical systems in high accuracy. IEEE Trans. Neural Netw. **9**(2), 294–307 (1998)
21. Wight, C.L., Zhao, J.: Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks. Commun. Comput. Phys. (2021)
22. Xu, K., Xiu, D.: Data-driven deep learning of partial differential equations in modal space. J. Comput. Phys. **408**, 109307 (2020)
23. Yang, X., Zhao, J., Wang, Q.: Numerical approximations for the molecular beam epitaxial growth model based on the invariant energy quadratization method. J. Comput. Phys. **333**, 102–127 (2017)
24. Zhao, J., Mau, J.: Discovery of governing equations with recursive deep neural networks. arXiv: 2009.11500 (2020)