

# A Physics Informed Neural Network for Time-Dependent Nonlinear and Higher Order Partial Differential Equations

Revanth Mattey <sup>a</sup>, Susanta Ghosh  <sup>a,b,\*</sup>

<sup>a</sup>*Department of Mechanical Engineering—Engineering Mechanics, Michigan Technological University, MI, USA*

<sup>b</sup>*The Center for Data Sciences, Michigan Technological University, MI, USA*

## Abstract

A physics informed neural network (PINN) incorporates the physics of a system by satisfying its boundary value problem through a neural network's loss function. The PINN approach has shown great success in approximating the map between the solution of a partial differential equation (PDE) and its spatio-temporal input. However, for strongly non-linear and higher order partial differential equations PINN's accuracy reduces significantly. To resolve this problem, we propose a novel PINN scheme that solves the PDE sequentially over successive time segments using a single neural network. The key idea is to re-train the same neural network for solving the PDE over successive time segments while satisfying the already obtained solution for all previous time segments. Thus it is named as backward compatible PINN (bc-PINN). To illustrate the advantages of bc-PINN, we have used the Cahn Hilliard and Allen Cahn equations, which are widely used to describe phase separation and reaction diffusion systems. Our results show significant improvement in accuracy over the PINN method while using a smaller number of collocation points. Additionally, we have shown that using the phase space technique for a higher order PDE could further improve the accuracy and efficiency of the bc-PINN scheme.

**Keywords:** Physics informed neural networks,, Partial differential equation (PDEs), Allen Cahn equation, Cahn Hilliard equation

## 1. Introduction

Traditional physics-based numerical methods for solving partial differential equations (PDEs) have found remarkable success in solving various science and engineering problems. These methods are accurate but computationally expensive for complex problems such as nonlinear PDEs and requires problem-specific techniques. In the last decade data driven methods have gained a lot of attention in almost all areas of science and engineering. Data driven methods for PDEs can help in identifying highly non-linear mappings (between the inputs and outputs) which can substitute or augment expensive physics based simulations. Due to their versatility and fast evaluation capabilities, machine learning based models can be used as PDE solvers in situations when there is a requirement of large number of simulations such as the inverse-problem and homogenization [1, 2].

Several data-driven techniques have been attempted to solve PDEs. For instance, the Gaussian Process based approaches described in [3, 4, 5, 6, 7]. Despite the ease of training for Gaussian Process, this approach did not gain as much popularity as neural network for solving

\*Corresponding author; Email: susantag@mtu.edu

PDEs due to its difficulties in handling high dimensional problems.

Among different data–driven techniques for PDEs the Physics Informed Neural Networks (PINN) has shown remarkable promise and versatility. PINN is a new class of machine learning technique where a neural network’s loss function is designed to satisfy the Initial Boundary Value Problem (IBVP) [8]. A PINN “learns” the non linear map between the spatio–temporal input and the solution of the PDE in a given domain. PINN utilizes the automatic-differentiation capability [9] to compute the derivatives of the field variables.

Different variants of PINN are shown to work effectively in solving many forward and inverse problems [10, 11, 12]. Recently in [13], PINNs have been extended to satisfy various conservation laws while solving the PDEs. This approach is named as cPINNs. cPINNs solve the problem over several sub-domains and ensure flux continuity at the boundaries of the sub-domains. While most of the PINN approaches solves the strong form of a PDE, it can also be used to solve the weak (variational) form of a PDE. Since the weak form incorporates the natural boundary conditions, the neural network solution only needs to satisfy the essential boundary conditions *a priori*. This aspect is used in several numerical methods for PDEs such as finite element method. Due to this advantage of weak form over strong form the application of PINNs on the weak form have been investigated in [14]. In this study the authors have considered the variational form for stochastic PDEs and applied the idea of PINNs to obtain the solution of the PDE. Also the corresponding uncertainty propagation through their model is presented in [14, 15]. Uncertainty quantification provides the variation associated with the prediction of the model. It is particularly useful for systems where there is a high cost of data acquisition or lack of high resolution data [16]. The authors in [17] proposed a Bayesian approach for physics informed neural network to solve forward and inverse problems.

The promise and versatility of PINN have been demonstrated through its application for a wide range of problems. PINN has been used in modelling subsurface transport phenomena [18], approximating Euler equations for high speed flows [1], constitutive modeling of stress strain behavior in biological tissues [19], predicting arterial blood pressure from noisy MRI data of flow velocity [20], and cardiac activation mapping for diagnosing atrial fibrillation [21].

In the present work, we demonstrate that the accuracy of the state-of-the-art PINN [8] suffers in the presence of (i) Strong Non-linearity, and (ii) Higher order partial differential operators. In order to illustrate the above, we chose the Allen Cahn equation having *strong non-linearity* and Cahn Hilliard equation having *strong non-linearity and fourth order derivative*. These are the two most widely used PDEs to study diffusion separation and multi phase flows [22, 23, 24, 25]. To overcome the drawbacks of state of the art PINN, we have proposed an extension, which is named as backward compatible PINN (bc-PINN). The proposed bc-PINN solves the PDE over successive time segments by re-training the same neural network, where the key idea is:

*To ensure that the neural network can reproduce the solution for all the prior time segments while solving the PDE for a particular time segment.*

Henceforth, this idea is referred as *backward compatibility*. Some of the main advantages of the proposed bc-PINN method are as follows:

1. It works for higher order and strongly nonlinear PDEs by using less number of iterations and collocation points while achieving significantly higher accuracy when compared to standard PINN.

2. A single neural network is used for the entire domain and continuity across the time segments is ensured for the predicted solution and its derivatives.

The rest of the paper is organized as follows: in section (2) the PINN method is briefly reviewed; in section (3) the proposed bc-PINN method is presented; in section (4) and (5) the bc-PINN method is analyzed and compared against the PINN method for the Allen Cahn equation and Cahn Hilliard equation respectively. Finally, the conclusions are presented in section (6).

## 2. A brief review of physics informed neural network (PINN) for partial differential equations

Physics informed neural network (PINN) is a class of machine learning model where the governing PDE is satisfied through the loss function of the neural network. The efficient optimization and prediction capabilities of neural network are exploited in the PINN approach. In PINN a neural network is trained to predict the solution at any point in the entire spatial-temporal domain. Let's consider the general form of a  $m^{\text{th}}$  order partial differential equation (PDE):

$$h_t = F(h(x, t), h_x^{(1)}(x, t), h_x^{(2)}(x, t), \dots, h_x^{(m)}(x, t)), \quad x \in \Omega \subset \mathbb{R}, \quad t \in (0, T] \quad (1)$$

Here,  $\Omega$  is an open set of  $\mathbb{R}$ .  $F$  is a non linear function of the solution  $h(x, t)$  and it's spatial derivatives  $(h_x^{(1)}(x, t), h_x^{(2)}(x, t), \dots, h_x^{(m)}(x, t))$  where  $x$  and  $t$  are the space and time coordinates respectively. The corresponding boundary conditions and initial conditions are

$$\begin{aligned} h(x, 0) &= \phi(x), \quad x \in \Omega \\ h(-x, t) &= h(x, t), \quad (x, t) \in \Gamma \times (0, T] \\ h_x^{(1)}(-x, t) &= h_x^{(1)}(x, t), \quad (x, t) \in \Gamma \times (0, T] \end{aligned} \quad (2)$$

Where,  $\Gamma$  is the boundary of  $\Omega$ . The PDE, the Initial and the Boundary Conditions (given by equation (1-2)) form a initial-boundary value problem (IBVP) considered in this study. The boundary conditions are taken as periodic and the initial condition is a real function.

PINN approximates the map between points in the spatio-temporal domain to the solution of the PDE. The parameters of the neural network are randomly initialized and iteratively updated by minimizing the loss function that enforces the PDE. The PINN's loss function consists of three error components, for the prediction of the neural network as in the following (i) Initial Condition, (ii) Boundary Condition, and (iii) PDE. Let  $\hat{h}(x, t)$  be the output of neural network. The three components of the PINN's loss function are given below:

- Mean squared error on the Initial Condition

$$\text{MSE}_I = \frac{1}{N_i} \sum_{k=1}^{N_i} \left( \hat{h}(x_k^i, 0) - h_k^i \right)^2, \quad x_k^i \in \Omega \quad (3)$$

where  $\hat{h}(x_k^i, 0)$  is the neural network output and  $h_k^i$  is the given initial condition at  $(x_k^i, 0)$ . Here, the superscript,  $(\bullet)^i$  stands for initial condition.

- Mean squared error on the Boundary Condition

$$\text{MSE}_B = \frac{1}{N_b} \sum_{k=1}^{N_b} \sum_{d=1}^{n_d} \left( \hat{h}^{(d-1)}(x_k^b, t_k^b) - \hat{h}^{(d-1)}(-x_k^b, t_k^b) \right)^2, \quad (x_k^b, t_k^b) \in \Gamma \times (0, T] \quad (4)$$

where  $n_d$  is the highest order of derivative to which the periodicity is enforced on the boundary,  $\Gamma$ . Here, the superscript,  $(\bullet)^b$  stands for boundary condition.

- The Mean squared error due to Residual of the partial differential equation

$$R := \hat{h}_t - F(\hat{h}, \hat{h}_x^{(1)}, \hat{h}_x^{(2)}, \dots \hat{h}_x^{(m)})$$

$$\text{MSE}_R = \frac{1}{N_r} \sum_{k=1}^{N_r} (R(x_k^r, t_k^r))^2, \quad (x_k^r, t_k^r) \in \Omega \times (0, T] \quad (5)$$

The superscript,  $(\bullet)^r$  stands for residual of the PDE.  $(x_k^i)$  and  $(x_k^b, t_k^b)$ , represent the set of points where the initial and boundary errors are computed. The residual/collocation error is computed at the collocation points  $(x_k^r, t_k^r)$ . These points on the domain and the boundary are obtained using a latin hypercube sampling approach. Therefore, the total loss function of the neural network is given by adding all the aforementioned mean squared errors

$$\text{MSE} = \text{MSE}_I + \text{MSE}_B + \text{MSE}_R \quad (6)$$

Once the PINN is trained, the accuracy of the predicted solution is computed with respect to the true/exact solution at unknown points (called testing points). Highly accurate solution of the initial boundary value problem obtained by the Chebyshev polynomial based numerical algorithm [26] and is considered as the exact solution. The relative total error ( $\varepsilon_{total}$ ) of the PINN's prediction over the entire domain is obtained by normalizing the error with respect to the true solution as

$$\varepsilon_{total} = \frac{\left[ \frac{1}{N} \sum_{k=1}^N (\hat{h}(x_k, t_k) - h(x_k, t_k))^2 \right]^{1/2}}{\left[ \frac{1}{N} \sum_{k=1}^N (h(x_k, t_k))^2 \right]^{1/2}} \quad (7)$$

The relative error ( $\varepsilon$ ) of the PINN's prediction at each point is obtained by normalizing the absolute error with respect to the true solution as

$$\varepsilon(x_k, t_k) = \frac{|\hat{h}(x_k, t_k) - h(x_k, t_k)|}{\left[ \sum_{k=1}^N (h(x_k, t_k))^2 \right]^{1/2}} \quad (8)$$

Where  $h(x_k, t_k)$  is the true solution and  $\hat{h}(x_k, t_k)$  is the neural network prediction for a set of testing points  $\{(x_k, t_k)\}_{k=1}^N$ ,  $(x_k, t_k) \in \Omega \times (0, T]$ . For all comparisons between true and predicted solutions the relative total error ' $\varepsilon_{total}$ ' and relative error ' $\varepsilon$ ' is used.

### 3. The proposed backward compatible sequential PINN method (BC-PINN)

In this section, we introduce an extension of the standard PINN technique that solves an initial-boundary value problem sequentially in time.

#### 3.1. bc-PINN

In the proposed method the PDE is solved progressively in time by re-training a single neural network over successive time segments. The limitation of such retraining is that the network can predict only for the latest time segment and cannot predict for previous time segments for

those it has been trained earlier. To overcome this limitation, the proposed model is designed to satisfy the solution of all the previous time segments while solving the PDE over a particular time segment. This scheme ensures backward compatibility of the solution by a single network. The proposed method is henceforth referred as backward compatible PINN (bc-PINN). The schematics of bc-PINN for a particular time segment is shown in figure (1) and the sequential scheme of proposed bc-PINN approach is shown in figure (2). In bc-PINN the time domain  $[0, T]$  is discretized into  $n_{max}$  segments as

$$[T_0 = 0, T_1], [T_1, T_2], \dots, [T_{n-1}, T_n], \dots, [T_{n_{max}-1}, T_{n_{max}} = T] \quad (9)$$

where the  $n^{\text{th}}$  segment is denoted as  $\Delta T_n = [T_{n-1}, T_n]$ ,  $n = 1, \dots, n_{max}$ .

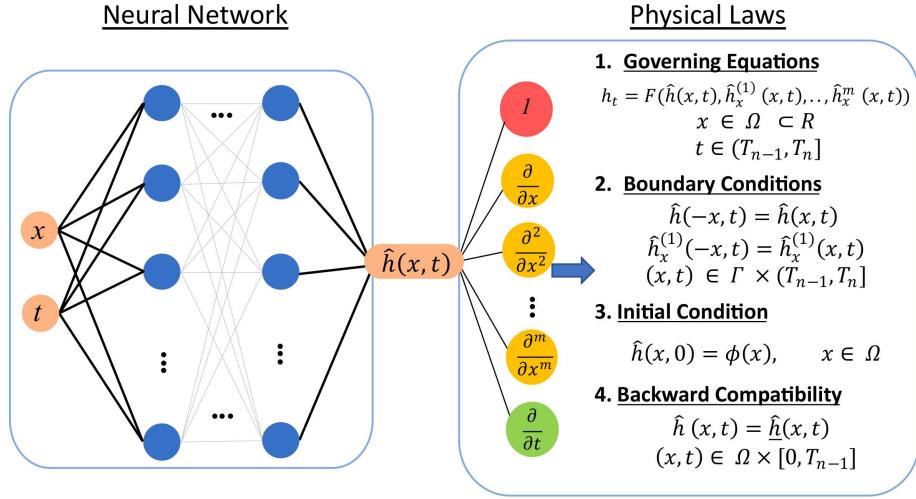


Figure 1: The schematics of the proposed backward compatible PINN (bc-PINN) approach for a time segment  $((T_{n-1}, T_n])$ . The neural network re-trains the PDE over  $(T_{n-1}, T_n]$  while satisfying the solution for all previous time segments.

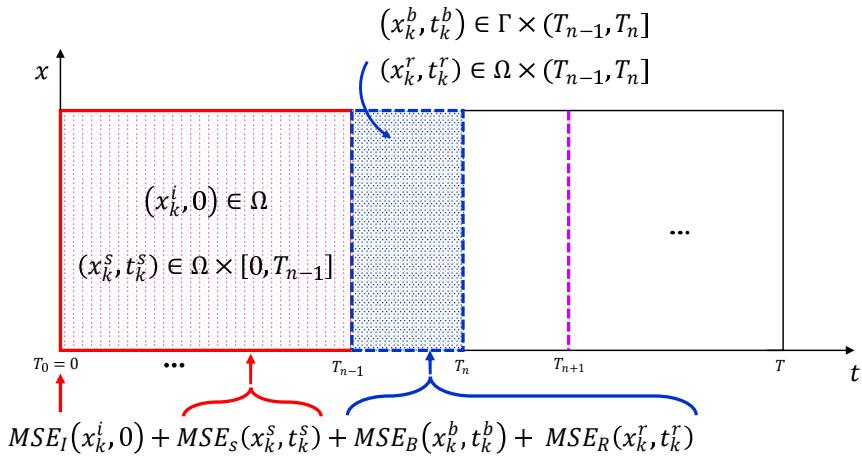


Figure 2: Illustration of the proposed backward compatibility scheme that satisfies the solutions obtained on all previous time segments  $([0, T_{n-1}])$  while satisfying the PDE on the current time segment  $((T_{n-1}, T_n])$ .

For the first time segment  $\Delta T_1$  the solution of the PDE is sought through the standard PINN by minimizing the following loss function

$$\begin{aligned} \text{MSE}_{\Delta T_1} &= \text{MSE}_I(x_k^i, 0) + \text{MSE}_B(x_k^b, t_k^b) + \text{MSE}_R(x_k^r, t_k^r) \\ &\quad (10) \\ x_k^i &\in \Omega, \quad (x_k^b, t_k^b) \in \Gamma \times (0, T_1] \quad (x_k^r, t_k^r) \in \Omega \times (0, T_1] \end{aligned}$$

Here,  $(x_k^i, t_k^i)$  represent the set of points where the error on initial condition is computed and  $(x_k^b, t_k^b)$  represent the set of points where the error on boundary condition is computed within the time segment  $\Delta T_1 = (0, T_1]$ . For all of the subsequent time segments (i.e.  $\Delta T_n$ ,  $n = 2, \dots, n_{max}$ ) we propose a novel loss function, which satisfies the solution of all previous time segments. The solution of all previous time segments is enforced by penalizing the departure from the already obtained solutions from the previous training, as given by

$$\begin{aligned} \text{MSE}_{\Delta T_n} &= \text{MSE}_I(x_k^i, 0) + \text{MSE}_B(x_k^b, t_k^b) + \text{MSE}_R(x_k^r, t_k^r) \\ &\quad + \text{MSE}_S(x_k^s, t_k^s), \quad n = 2, \dots, n_{max} \\ x_k^i &\in \Omega, \quad (x_k^b, t_k^b) \in \Gamma \times (T_{n-1}, T_n] \\ (x_k^r, t_k^r) &\in \Omega \times (T_{n-1}, T_n], \quad (x_k^s, t_k^s) \in \Omega \times [0, T_{n-1}] \end{aligned} \quad (11)$$

Here,  $(x_k^i, 0)$  represent the set of points where the error on initial condition is computed and  $(x_k^b, t_k^b)$  represent the set of points where the error on boundary conditions is computed within the time segment  $(T_{n-1}, T_n]$ . The residual/collocation error as given in equation (5) is computed at the collocation points  $(x_k^r, t_k^r)$ . We also minimize the departure from the already obtained solution that were stored at the grid points  $(x_k^s, t_k^s)$ . The solution obtained (on  $(0, T_n]$ ) at the  $n^{th}$  segment is stored for using it in the  $(n+1)^{th}$  segment.

### 3.2. Details of the neural network of bc-PINN

We have used a standard (deep) neural network with two input neurons consisting of the spatial variable ( $x$ ) and temporal variable ( $t$ ). The output of the neural network ( $\hat{h}(x, t)$ ) approximates the solution of the PDE ( $h(x, t)$ ). To avoid model bias due to input features of different scales we have performed “min-max” normalization to scale the data uniformly. The neural network has 4 hidden layers consisting 200 neurons in each layer with a  $tanh$  activation function. The neural network has more than 100,000 learning parameters which have been initialized using the “xavier initialization” technique. The optimization of the loss function and updating the learning parameters (weights and biases of the neural network) is performed using the ADAM and LBFGS optimizers. The learning rate for ADAM optimizer is considered as 0.001 with all other parameters as suggested in [27]. Following standard PINN, after training the neural network using the ADAM optimizer we again train it using the L-BFGS optimizer until one of the following stopping criteria is met: (i) Maximum iterations are equal to 50,000 (ii) Maximum number of function evaluations are equal to 50,000 (iii) Maximum number of line search steps (per iteration) equal to 50 (iv) The maximum number of variable metric corrections used to define the limited memory matrix are equal to 50 (v) The iteration stops when  $\frac{|f^k - f^{k+1}|}{\max(|f^k|, |f^{k+1}|, 1)} \leq 2.22044604925e-16$ , where  $f$  is the neural network objective function and  $k$  is the iteration number.

### 3.3. Details of the Computational Platform

All the neural networks are trained on Nvidia Tesla P100 (3584 CUDA cores and 16GB of HBM2 vRAM) and Nvidia Volta V100 GPU (5120 CUDA cores, 640 Tensor cores and 16GB of

HBM2 vRAM). For inferencing and generating the true solutions via chebfun, we have used Dell precision 3630 workstation with Intel core i7-9700k 8 core (4.9 GHz Turbo) and 32 GB RAM. The software packages used for all the computations are Tensorflow 1.15 and MATLAB R2020a.

### 3.4. The reference solution

Accurate numerical solutions for the Allen Cahn and Cahn Hilliard equations are obtained using the chebfun package [26]. The chebfun approach provides a polynomial interpolant for smooth functions in Chebyshev points. To solve time varying PDEs an exponential time differencing with Runge–Kutta time stepping scheme [28] has been implemented in chebfun, which is used in the present work. Henceforth, these solutions are considered as the true/exact solutions. We have taken 512 points for spatial discretization and 201 points for discretization in time scale. A fourth order Runge–Kutta time integrator with time step  $\Delta t = 10^{-5}$  is used.

The bc-PINN approach is applied to solve Allen Cahn equation and Cahn Hilliard equation in the next section to demonstrate its advantages for nonlinear and higher order PDEs in comparison to standard PINN method [8].

## 4. Allen Cahn Equation

### 4.1. Allen Cahn equation and parameters

The Allen Cahn equation is a semilinear partial differential equation which is well known for certain phase separation problems [29, 30, 31]. For every  $x \in \Omega$ , ( $\Omega$  is an open set of  $\mathbb{R}^n$ ) the Allen Cahn equation is the  $L^2$  gradient flow of the functional

$$I_{c_1}(h) = \frac{1}{2} \int_{\Omega} |\nabla h|^2 dx + \frac{1}{c_1^2} \int_{\Omega} F(h) dx \quad (12)$$

For a phase separation problem, the parameter  $h$  represents the concentration of the individual component and the parameter  $c_1$  represents the interfacial thickness. The solution  $h$  tends to minimize the energy functional  $I_{c_1}(h)$  such that it tends to the minimums of  $F(h)$ . The solution progressively develops interfaces separating different phases. For a given initial condition,  $h_0 \in L^2(\Omega)$  and  $T > 0$  we seek a function  $h : \Omega \times (0, T] \rightarrow \mathbb{R}$  which satisfies the equation below.

$$\begin{aligned} h_t - c_1^2 \nabla^2 h + f(h) &= 0, \quad t \in (0, T], \quad x \in \Omega \subset \mathbb{R} \\ f(h) &= c_2(h^3 - h) \\ h(x, 0) &= x^2 \cos(\pi x) \\ h(x, t) &= h(-x, t) \\ h_x^{(1)}(x, t) &= h_x^{(1)}(-x, t) \end{aligned} \quad (13)$$

The function  $f$  is the derivative with respect to  $h$  of a double well potential function  $F$  where,  $F \in C^1(\mathbb{R})$  is a non-negative function satisfying  $F(\pm 1) = 0$ . We have considered  $F = 5(h^2 - 1)^2/4$  and  $f = 5(h^3 - h)$ . Here the values of parameters for equation (13) are  $c_1^2 = 0.0001$  and  $c_2 = 5$ .

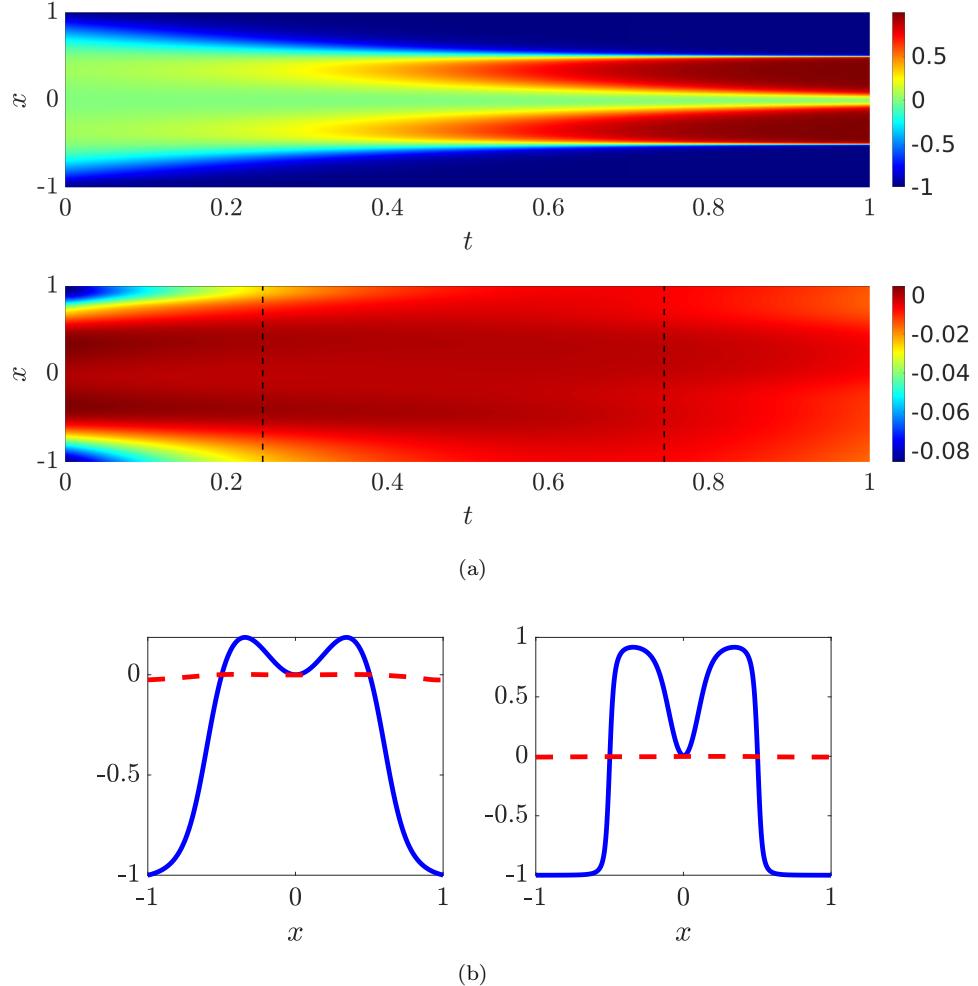


Figure 3: (a): The exact solution (Top) and the PINN solution (Bottom) of the Allen Cahn equation for the entire spatio-temporal domain. (b): Time snapshots for the exact solution (—) and the PINN solution (---) at  $t = 0.25$  and  $t = 0.75$ .

#### 4.2. PINN for Allen Cahn equation

At first we solve the Allen Cahn equation on  $\Omega = [-1, 1]$  and  $(0, 1]$  using standard PINN to demonstrate the challenge associated with non-linearity. For training the PINN, we have used 20,000 collocation points and trained for 100,000 ADAM iterations. The loss function for PINN is described in equation (3),(4) and (5). The solution of standard PINN is quite erroneous as shown in figure (3). In order to understand the reason for failure of the standard PINN, we analyze its prediction for the individual terms of the Allen Cahn equation. Figure (4) shows the individual terms of the Allen Cahn equation obtained through the Chebfun method and the PINN. We observe that the PINN fail to predict the non-linear term ( $5(h^3 - h)$ ) of the Allen Cahn equation. Therefore we have shown that the standard PINN [8] does not work for the Allen Cahn equation that consist of a strongly non-linear term.

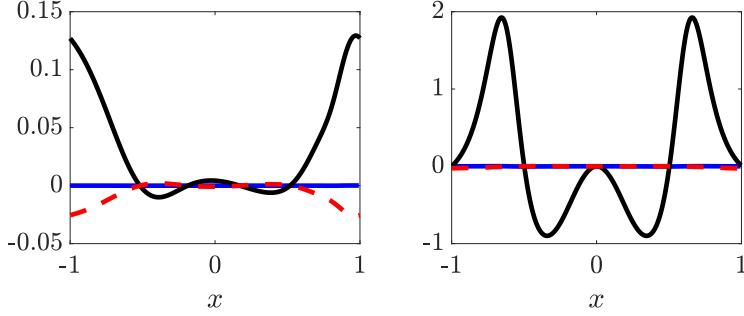


Figure 4: Individual terms of the Allen Cahn Equation obtained through the (Left): standard PINN method (Right): Chebfun method.  $h(x, t)$ , (---),  $0.0001\nabla^2 h$  (—) and  $5(h^3 - h)$  (—) at  $t = 0.25$ .

#### 4.3. bc-PINN for Allen Cahn equation

To overcome this limitation of PINN, we use the backward compatible PINN approach. In the bc-PINN we propose the following loss function for any given time segment  $\Delta T_n$  as given below.

- Mean squared error on the initial Condition is the same as equation (3).
- Mean squared error on the boundary Condition

$$\text{MSE}_B = \frac{1}{N_b} \sum_{k=1}^{N_b} \sum_{d=1}^{n_d} \left( \hat{h}^{(d-1)}(x_k^b, t_k^b) - \hat{h}^{(d-1)}(-x_k^b, t_k^b) \right)^2, \quad (x_k^b, t_k^b) \in \Gamma \times (T_{n-1}, T_n] \quad (14)$$

where  $n_d$  is the order to which periodicity is enforced on the boundary  $\Gamma$ . Here, the superscript,  $(\bullet)^b$  stands for boundary condition.

- The Mean squared error due to residual of the partial differential equation

$$R := \hat{h}_t - c_1^2 \nabla^2 \hat{h} + f(\hat{h})$$

$$\text{MSE}_R = \frac{1}{N_r} \sum_{k=1}^{N_r} (R(x_k^r, t_k^r))^2, \quad (x_k^r, t_k^r) \in \Omega \times (T_{n-1}, T_n] \quad (15)$$

The superscript,  $(\bullet)^r$  stands for residual of the PDE.

- Mean squared error for backward compatibility

$$\text{MSE}_S = \frac{1}{N_s} \sum_{k=1}^{N_s} \left( \hat{h}(x_k^s, t_k^s) - \underline{h}(x_k^s, t_k^s) \right)^2, \quad (x_k^s, t_k^s) \in \Omega \times [0, T_{n-1}] \quad (16)$$

where,  $\hat{h}(x, t)$  is the neural network prediction and  $\underline{h}(x, t)$  is the known solution through the neural network from the previous time steps  $\Omega \times [0, T_{n-1}]$ . The superscript,  $(\bullet)^s$  stands for the backward compatible solution.

- The total mean squared error or loss is given as

$$\text{MSE}_{\Delta T_n} = \text{MSE}_I + \text{MSE}_B + \text{MSE}_R + \text{MSE}_S \quad (17)$$

The hyper-parameters associated with training the bc-PINN are number of ADAM iterations ( $N_{iter}$ ), time steps per segment and number of collocation points ( $N_r$ ) per segment. Table 1 shows the values of all the aforementioned hyper-parameters for training the bc-PINN.

Variable	Description	Number
$N_i$	Initial Datapoints	512
$N_b$	Boundary Datapoints	40/segment
$N_r$	Collocation points	20000/segment
$N_{iter}$	Number of ADAM iterations	10000/segment

Table 1: Description of Training Data for Allen Cahn Equation. The segment considered here consists of 40 time steps.

The exact and predicted solution at time  $t = 0.25$  obtained by the standard PINN and bc-PINN are shown in figure (5). While the standard PINN fails, the proposed bc-PINN predicts the solution quite accurately. The relative total errors ( $\varepsilon_{total}$ ) for both the approaches are shown in table 2.

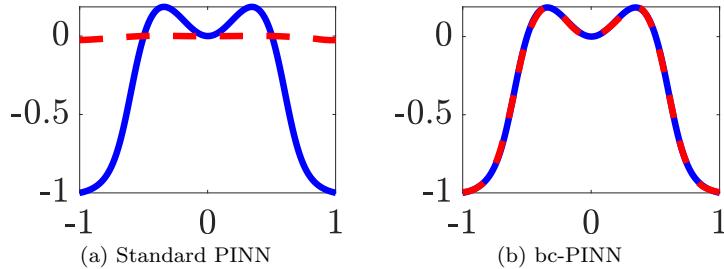


Figure 5: Exact (—) and Predicted (---) solution at time  $t = 0.25$

Method	Error( $\varepsilon_{total}$ )
Standard PINN	0.9919
bc-PINN	0.0701

Table 2: Relative total errors (equation (7)) over the entire domain with respect to Chebfun solution for different methods.

The comparison between the predicted solution using bc-PINN and the chebfun solution is shown in figure (6). This shows that the bc-PINN can accurately predict the solution for the entire domain. The solutions and errors by the PINN and bc-PINN are compared in figure (7), showing much higher accuracy by the bc-PINN. The error plots confirms high accuracy of bc-PINN. The error increases with time very slowly. This is due to two reasons: (i) the solution becomes progressively phase-separated (between zero and one) yielding greater curvatures and sharp phase-boundaries that are difficult to capture, and (ii) due to the sequential nature of the bc-PINN approach the error accumulates with time progression, which is similar to the time-integrators. To illustrate the high accuracy of the bc-PINN approach, solutions and errors for different values of the interfacial thickness ( $c_1$ ) is plotted in figure (8). As we decrease the

parameter  $c_1$ , it can be seen that the error in the prediction decreases. The parameter  $c_1$  controls the effect of the double derivative of the solution ( $\nabla^2 h$ ). Therefore, as we decrease  $c_1$  the error due to the approximation in derivative reduces and thus the accuracy of the bc-PINN solution increases. In Appendix B a new loss function including a logarithmic residual for the Allen Cahn equation is discussed. This new logarithmic residual bc-PINN approach and its results are presented in comparison with the simple bc-PINN approach without a logarithmic residual.

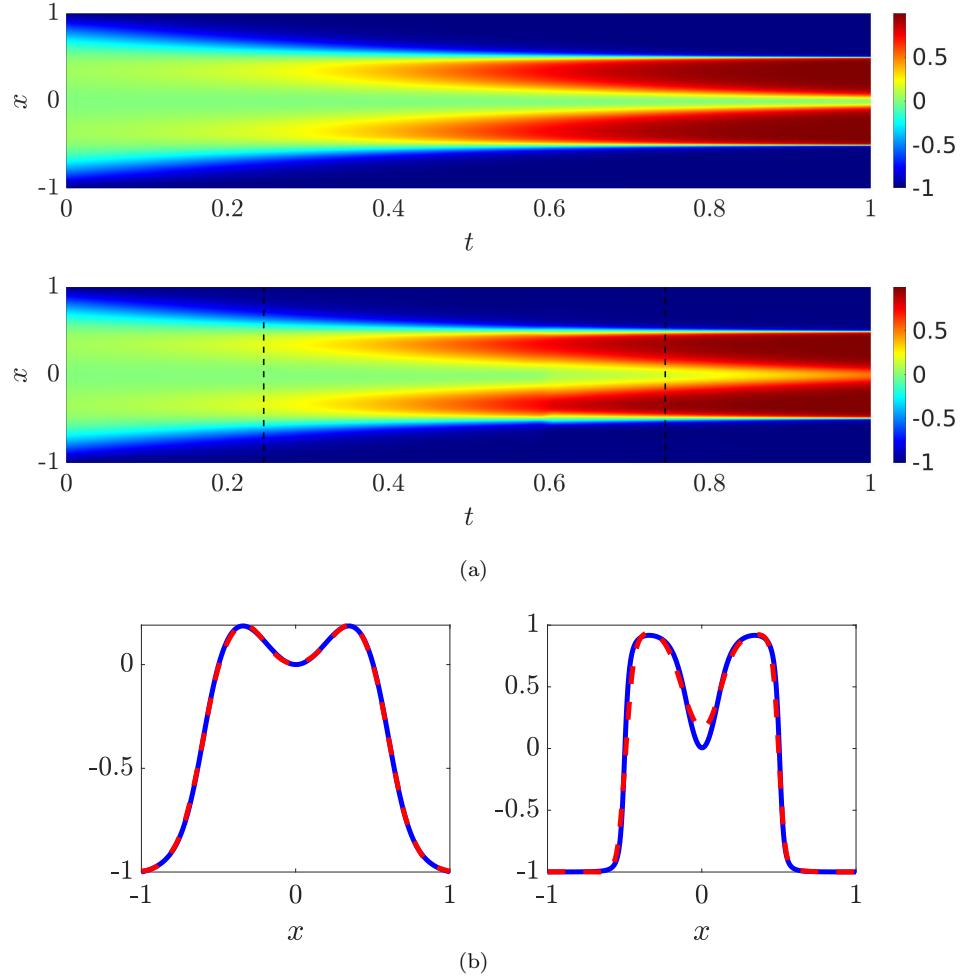
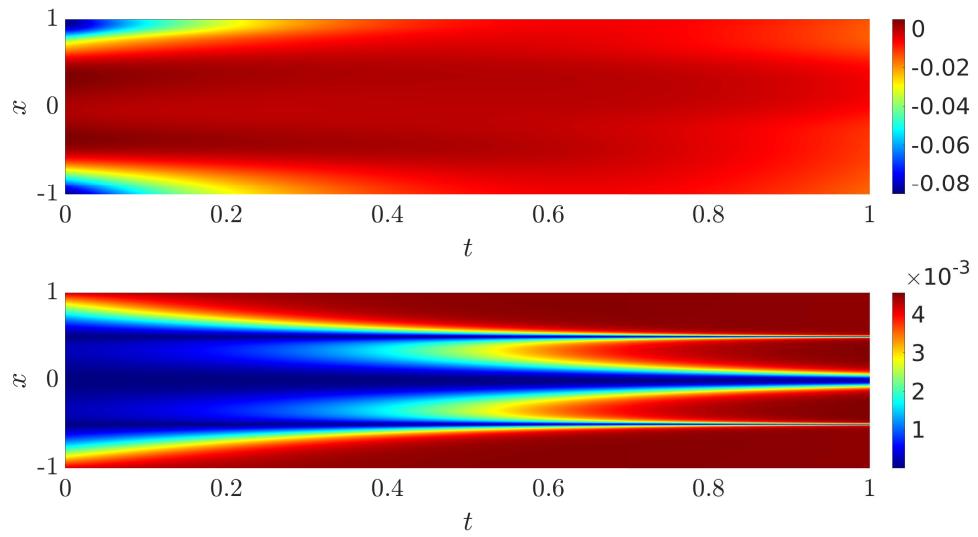
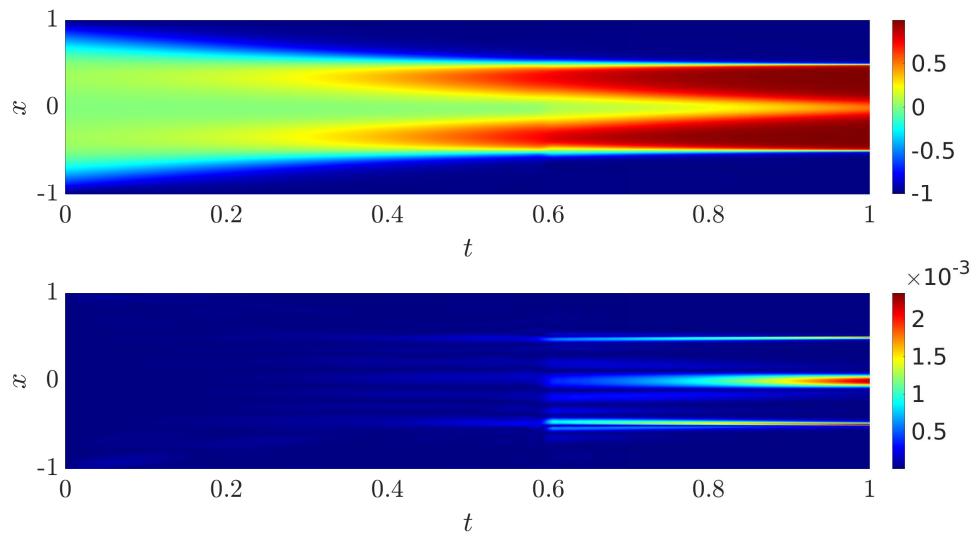


Figure 6: (a): Exact (Top) and bc-PINN (Bottom) solutions of the Allen Cahn equation for the entire spatio-temporal domain. (b): The exact (—) and the bc-PINN (---) solutions at time  $t = 0.25$  and  $t = 0.75$ .

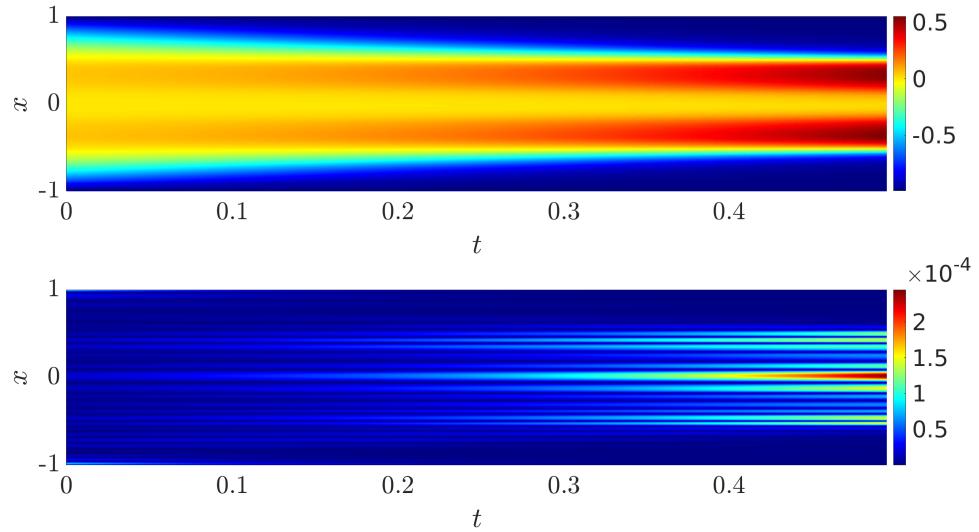


(a) Predicted solution (top) and relative error (bottom) obtained using PINN

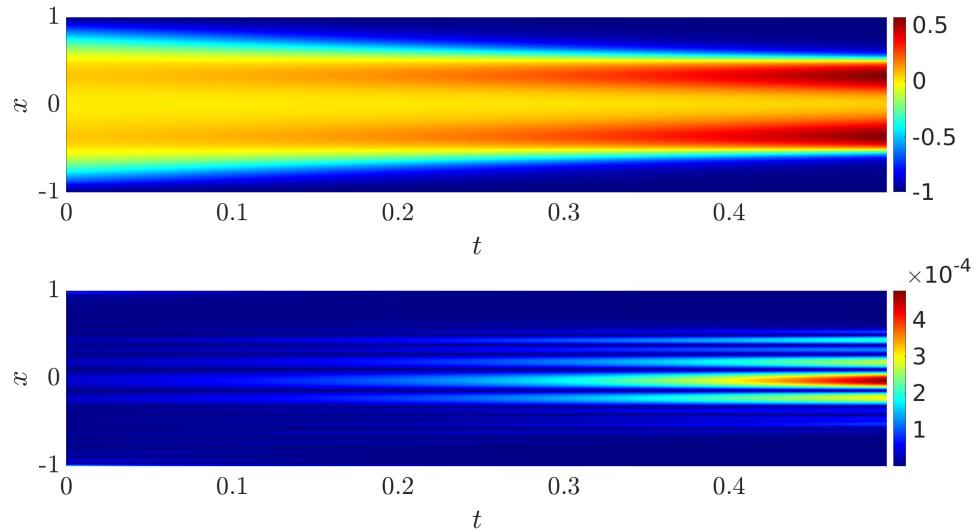


(b) Predicted solution (top) and relative error (bottom) obtained using bc-PINN

Figure 7: Solution and relative error for Allen Cahn equation through bc-PINN.



(a) Predicted solution (top) and relative error (bottom) obtained using bc-PINN for  $c_1^2 = 0.00001$



(b) Predicted solution (top) and relative error (bottom) obtained using bc-PINN for  $c_1^2 = 0.00005$

Figure 8: Solutions and relative errors of the Allen Cahn equation for different  $c_1^2$  (of equation (13)) obtained by the bc-PINN method.

## 5. Cahn Hilliard Equation

### 5.1. Cahn Hilliard equation and parameters

The Cahn Hilliard equation is considered here to demonstrate the advantages of the proposed bc-PINN method for nonlinear and higher order PDEs. The Cahn Hilliard equation [32, 33, 34] plays an essential role in the field of material science for describing the qualitative features in a phase separation process for two phase systems (assuming isotropy and constant temperature)<sup>1</sup>. Since the entire process is governed by the Cahn Hilliard equation it is essential to understand the physical significance of each individual variable.

$$h_t(x, t) - \nabla^2(-\alpha\kappa\nabla^2 h(x, t) + \kappa f(h(x, t))) = 0, \quad x \in \Omega, \quad t \in (0, T] \quad (18)$$

Here  $\Omega$  is an open set in  $\mathbb{R}$ . The order parameter  $h$  in equation (18), refers to the rescaled density or concentration of one of the material components in the system and it takes values between (-1 and 1 which corresponds to their pure states). The density of second component is  $1 - h$ , and this ensures that the total density over the simulation domain is a conserved quantity. In addition, the function  $f(h)$ , is the derivative of a double well potential function. The two phases of the system correspond to the two wells of the double well potential function. The parameter  $\kappa$  is the mobility parameter and the parameter  $\alpha$  is related to the surface tension at the interface. Here the values of parameters are taken as  $\alpha = 0.02$  and  $\kappa = 1$ .

### 5.2. PINN for Cahn Hilliard equation

Initially we solve the Cahn Hilliard equation using standard PINN to demonstrate the challenge for this equation. The domain Cahn Hilliard equation on  $\Omega = [-1, 1]$  and periodic boundary conditions. The time domain considered is  $(0, T] = (0, 1]$ . For training the standard PINN, we have used 20,000 collocation points and trained for 100,000 ADAM iterations. The loss function for PINN is described in equation (3), (4) and (5). The solution predicted after training is shown in figure (9) and it can be observed that there is significant mismatch between the PINN and the exact solution.

The two possible reasons for the inaccurate solution are strong non-linearity and the high order derivative terms (fourth order). In PINN the derivatives are approximated using automatic differentiation. It has been shown that as the order of the derivative increases the complexity in automatic differentiation increases and it becomes computationally expensive [9]. In order to overcome the difficulty in approximating the higher order derivative via automatic differentiation, we adopt the phase space representation in the proposed bc-PINN.

### 5.3. bc-PINN for Cahn Hilliard equation

In this section, we introduce the bc-PINN approach with a phase space representation for solving the Cahn Hilliard Equation. The phase space representation is widely used to represent a high order PDE into coupled multiple lower order PDEs. We show that by adopting phase space representation the accuracy of the bc-PINN method improves significantly. The phase

---

<sup>1</sup>The process of phase separation can be observed when a binary alloy is cooled down adequately. This leads to a state of total nucleation which is mainly referred to as spinodal decomposition. In the subsequent stage coarsening occurs in the nucleated microstructure at a much slower rate. This whole phase separation phenomena affects the mechanical properties (eg. strength, hardness and fracture toughness) of the material.

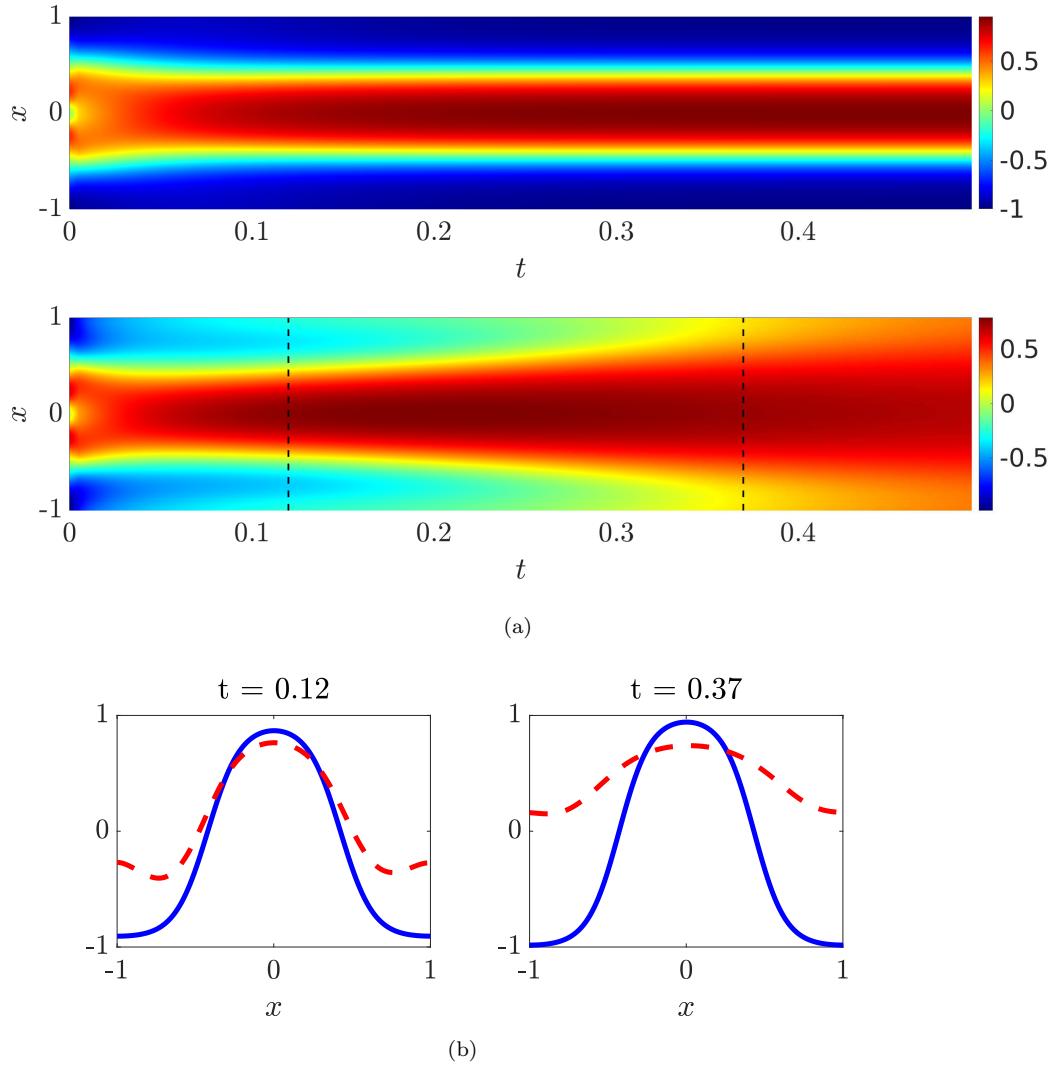


Figure 9: (a): The exact solution (Top) and the PINN solution (Bottom) of the Cahn Hilliard equation for the entire spatio-temporal domain. (b): Time snapshots for the exact solution (—) and the PINN solution (- - -) at  $t = 0.12$  and  $t = 0.37$ .

space representation of the Cahn Hilliard equation (a fourth order PDE, equation (18)) yields two coupled second order PDEs.

$$\begin{aligned}
h_t &= \nabla^2(-\alpha \kappa \mu + \kappa f(h)), \quad \mu = \nabla^2 h \quad t \in (0, T], \quad x \in \Omega \subset \mathbb{R} \\
h(x, 0) &= \cos(\pi x) - \exp(-4(\pi x)^2) \\
h(-x, t) &= h(x, t), \quad (x, t) \in \Gamma \times (0, T] \\
h_x^{(1)}(-x, t) &= h_x^{(1)}(x, t), \quad (x, t) \in \Gamma \times (0, T] \\
\mu(-x, t) &= \mu(x, t), \quad (x, t) \in \Gamma \times (0, T] \\
\mu_x^{(1)}(-x, t) &= \mu_x^{(1)}(x, t), \quad (x, t) \in \Gamma \times (0, T]
\end{aligned} \tag{19}$$

Where,  $f(h) = h(h^2 - 1)$ . Therefore there are two outputs of the neural network  $\hat{h}(x, t)$  and  $\hat{\mu}(x, t)$  in the present method. The input features are the spatio-temporal variables  $(x, t)$ . The modified loss function for the coupled phase space system includes an error on initial condition, error on the boundary conditions and error on the residual. In addition it will have the error for the backward compatibility. Therefore, the total loss function (equation (24)) for any time segment  $\Delta T_n$  is sum of all the aforementioned errors given in equation (20–23).

- Mean squared error on the initial condition for  $h(x, t)$  and  $\mu(x, t)$

$$\text{MSE}_I = \frac{1}{N_i} \left\{ \sum_{k=1}^{N_i} \left( \hat{h}(x_k^i, 0) - h_k^i \right)^2 + \sum_{k=1}^{N_i} \left( \hat{\mu}(x_k^i, 0) - \mu_k^i \right)^2 \right\}, \quad x_k^i \in \Omega \quad (20)$$

Here, the neural network output is  $\hat{h}(x_k^i, 0), \hat{\mu}(x_k^i, 0)$  and the given initial condition is  $h_k^i, \mu_k^i$  at  $(x_k^i, 0)$ . The superscript,  $(\bullet)^i$  stands for initial condition.

- Mean squared error on the boundary Condition

$$\begin{aligned} \text{MSE}_{B_h} &= \frac{1}{N_b} \left\{ \sum_{k=1}^{N_b} \sum_{d=1}^{n_d} \left( \hat{h}^{(d-1)}(x_k^b, t_k^b) - \hat{h}^{(d-1)}(-x_k^b, t_k^b) \right)^2 \right\} \\ \text{MSE}_{B_\mu} &= \frac{1}{N_b} \left\{ \sum_{k=1}^{N_b} \sum_{d=1}^{n_d} \left( \hat{\mu}^{(d-1)}(x_k^b, t_k^b) - \hat{\mu}^{(d-1)}(-x_k^b, t_k^b) \right)^2 \right\} \\ \text{MSE}_B &= \text{MSE}_{B_h} + \text{MSE}_{B_\mu}, \quad (x_k^b, t_k^b) \in \Gamma \times (T_{n-1}, T_n] \end{aligned} \quad (21)$$

where  $n_d$  is the order to which periodicity is enforced. The superscript,  $(\bullet)^b$  stands for boundary condition.

- Mean squared error on the Residual of the partial differential equation

$$\begin{aligned} R_1 &:= \hat{h}_t - \nabla^2 \hat{h}(-\mu + f(\hat{h})) \\ R_2 &:= \hat{\mu} - \nabla^2 \hat{h} \\ \text{MSE}_R &= \frac{1}{N_r} \left\{ \sum_{k=1}^{N_r} R_1(x_k^r, t_k^r)^2 + \sum_{k=1}^{N_r} R_2(x_k^r, t_k^r)^2 \right\}, \quad (x_k^r, t_k^r) \in \Omega \times (T_{n-1}, T_n] \end{aligned} \quad (22)$$

The superscript,  $(\bullet)^r$  stands for residual of the PDE.

- Mean squared error for backward compatibility

$$\text{MSE}_S = \frac{1}{N_s} \sum_{k=1}^{N_s} \left( \hat{h}(x_k^s, t_k^s) - \underline{h}(x_k^s, t_k^s) \right)^2, \quad (x_k^s, t_k^s) \in \Omega \times [0, T_{n-1}] \quad (23)$$

where,  $\hat{h}(x, t)$  is the neural network prediction and  $\underline{h}(x, t)$  is the known solution through the neural network from the previous time steps  $\Omega \times [0, T_{n-1}]$ . The superscript,  $(\bullet)^s$  stands for the backward compatible solution.

- The total mean squared error or loss is given as

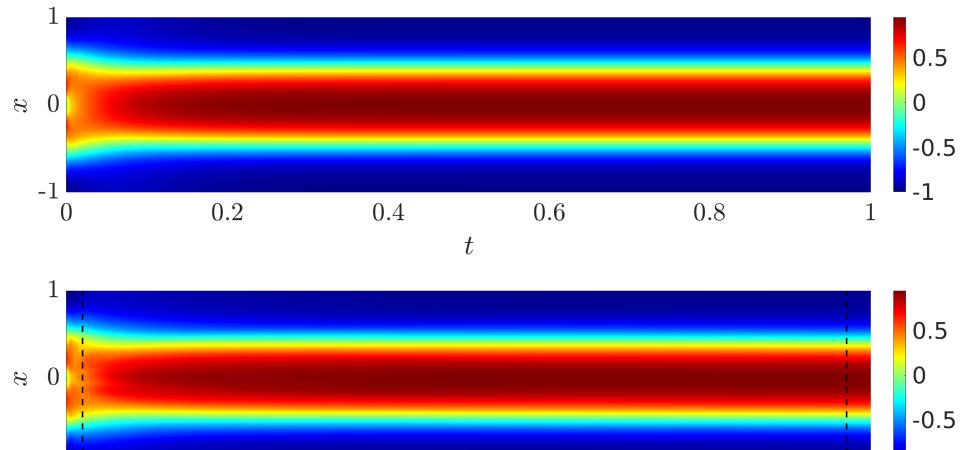
$$\text{MSE}_{\Delta T_n} = \text{MSE}_I + \text{MSE}_B + \text{MSE}_R + \text{MSE}_S \quad (24)$$

Variable	Description	Number
$N_i$	Initial datapoints	512
$N_b$	Boundary datapoints	10/segment
$N_r$	Collocation points	5000/segment
$N_{iter}$	Number of ADAM iterations	10000/segment

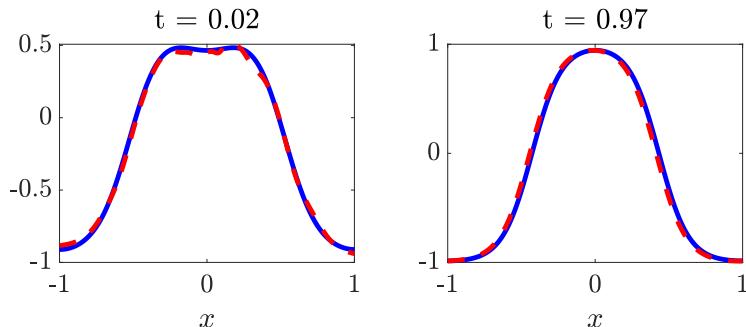
Table 3: Description of training data for Cahn Hilliard equation. 10 time steps/segment have been considered and the amount of collocation points generated remains same and doesn't increase as we progress through time.

The boundary loss (equation (21) is applied for  $n_d = 1$  on  $\hat{h}$  and  $\hat{\mu}$ , to represent periodic boundary conditions. Equation (22) describes two components of residual for two PDEs in the phase space form of the Cahn Hilliard equation (equation (19)). Table 3 describes the values of the hyper-parameters used in bc-PINN.  $N_i$  and  $N_b$  refers to the number of points considered to enforce the initial and boundary condition respectively.  $N_r$  is the number of collocation points per time segment and  $N_{iter}$  is the number of ADAM iterations used to train the neural network per time segment .

The accuracy of the proposed bc-PINN approach is shown by comparing it against the exact solution obtained by the chebfun method in figure (10). This shows that the phase space representation with bc-PINN can closely match the exact solution for the Cahn Hilliard equation. The relative total error ( $\varepsilon_{total}$ ) obtained for the bc-PINN solution is 0.036 whereas for the standard PINN solution the error is 0.8594. It is evident from the error plots given in figure (11) that a more accurate solution is obtained by using the bc-PINN compared to standard PINN. The higher accuracy can be accredited to the fact that approximating lower order derivatives using automatic differentiation is much simpler. One key observation to note is that the solution in the  $n^{\text{th}}$  time segment takes the solution at time  $T_{n-1}$  from the  $(n-1)^{\text{th}}$  time segment as initial condition. Thus, only the error at the end point in a time segment is propagated to the next time segment. For instance, only the error at the time  $T_{n-1}$  in  $(n-1)^{\text{th}}$  time segment is propagated to the next time segment. Errors at all other time steps in  $(n-1)^{\text{th}}$  time segment does not propagate to the  $n^{\text{th}}$  time segment. This can be observed in figure (11b), even though the error at time 0.01 is quite high but since this is not the end point of the time segment  $[0, 0.05]$  it does not propagate with time. The error in the first time segment can be further reduced by using more iterations and the accuracy of the total solution can be improved. To further demonstrate the effectiveness of the current phase space backward compatible training approach, we have taken different values of the parameter ( $\alpha\kappa$ ) and compared the predicted solutions with the exact solutions generated using chebfun which is shown in figure (12). The proposed phase space representation with bc-PINN approach can be extended to any partial differential equation consisting higher order derivatives and non-linearity.

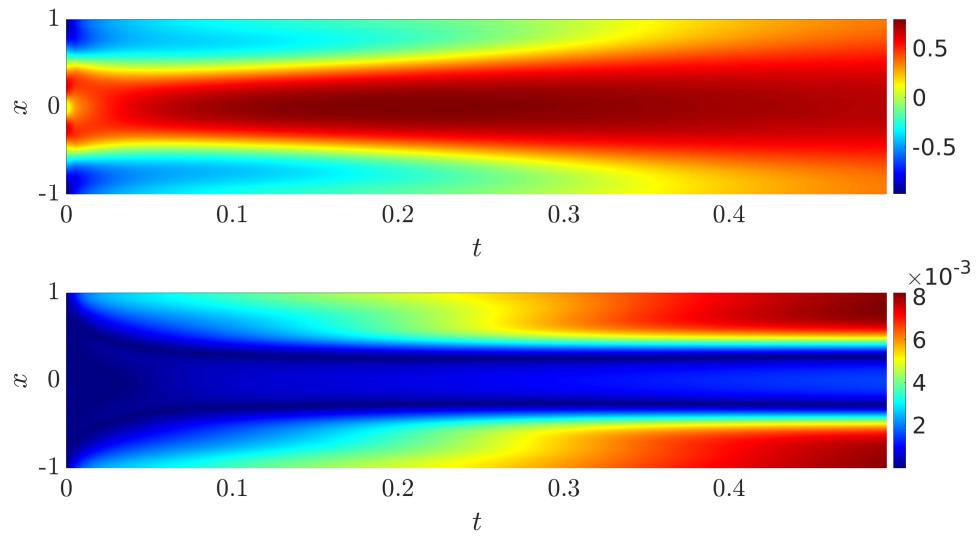


(a)

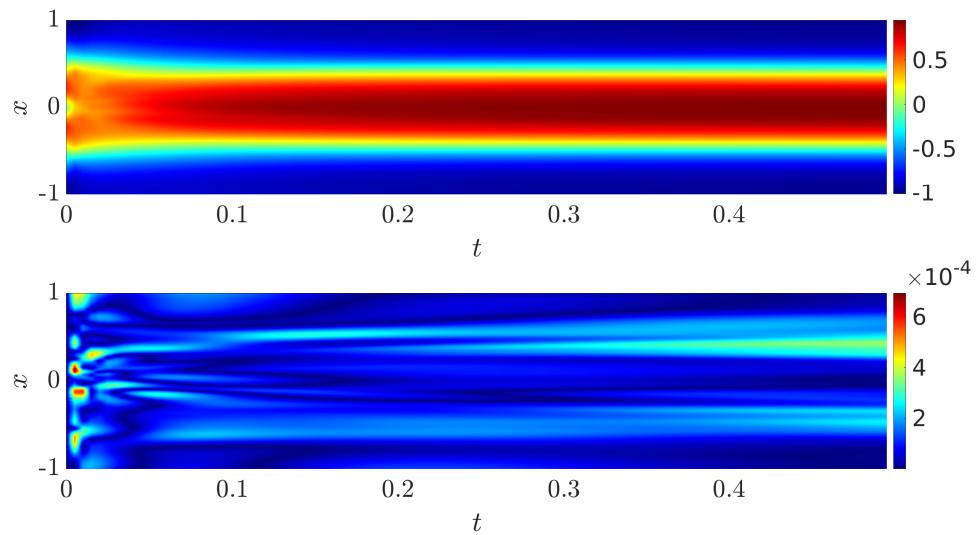


(b)

Figure 10: (a): Exact (Top) and bc-PINN (Bottom) solutions of the Cahn–Hilliard equation for the entire spatio–temporal domain. (b): The exact (—) and the bc-PINN (---) solutions at time  $t = 0.02$  and  $t = 0.97$ .

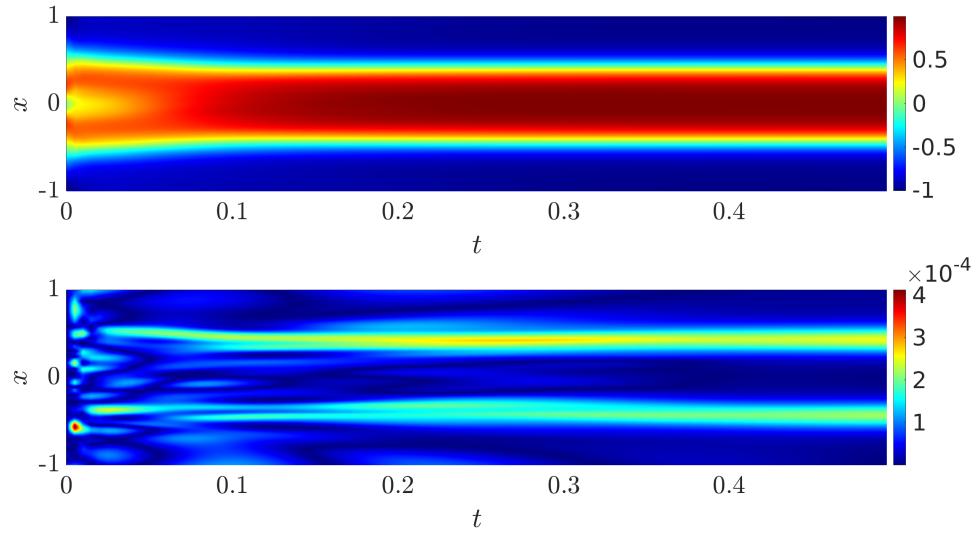


(a) Solution (top) and relative error (bottom) via standard PINN

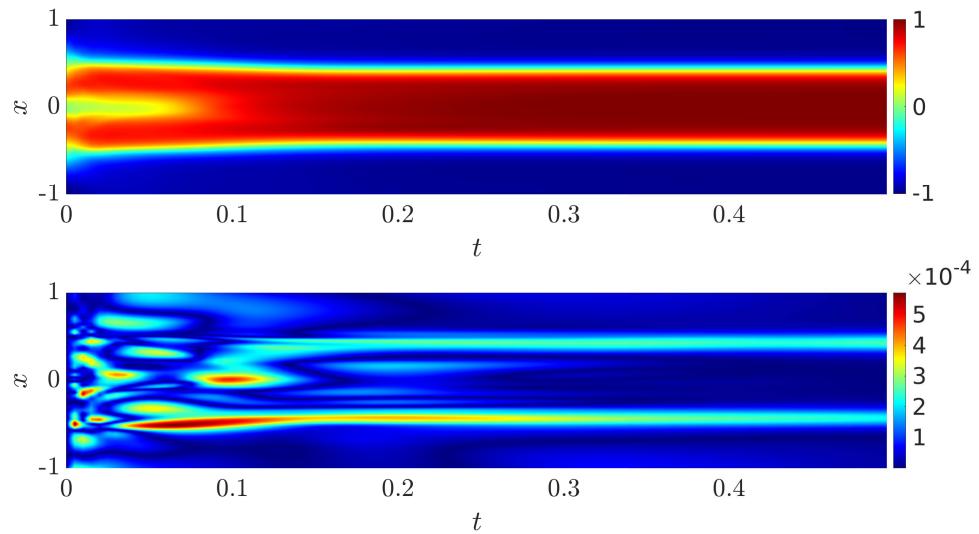


(b) Solution (top) and relative error (bottom) via bc-PINN

Figure 11: Solution and error associated with respect to the true solution for Cahn Hilliard equation.



(a) Solution (top) and error (bottom) for  $\alpha\kappa = 0.001$



(b) Solution (top) and error (bottom)  $\alpha\kappa = 0.0005$

Figure 12: Solution and relative errors of the Cahn Hilliard equation for different parameters ( $\alpha\kappa$  of equation (19)) obtained by the bc-PINN method.

## 6. Conclusions

We have proposed a new PINN approach (named as bc-PINN) for solving nonlinear and higher order PDEs. The bc-PINN re-trains the neural network over successive time segments while satisfying the solution for all previous time segments.

Additionally, bc-PINN incorporates different techniques such as logarithmic residual, phase space representation of the PDE to improve its accuracy. The key advantages of bc-PINN are summarized below.

The proposed bc-PINN method can provide accurate solution for nonlinear and higher-order PDEs such as Cahn Hilliard and Allen Cahn equations, where standard PINN faces difficulties.

Moreover, the proposed method can achieve high accuracy by using less number of collocation points. The phase space technique used in the bc-PINN significantly reduces the time required to compute the derivatives in a higher order PDE. Despite the segmentation of the time domain, it requires only one neural network and provides a continuous solution for the entire spatio-temporal domain. The proposed backward compatibility scheme may enhance many other machine learning approaches applied to complex systems represented by time dependent PDEs.

**Acknowledgments:** SG acknowledges the financial support by NSF (CMMI MoMS) under grant number 1937983. We acknowledge Superior, a high-performance computing facility at MTU and Google Colab, a cloud service hosted by Google. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) (allocation number MSS200004), which is supported by the NSF grant number ACI-1548562.

## Appendix A. Hyper-parameter selection for bc-PINN

As discussed in section 4 & 5, the proposed method there have a number of hyper-parameters like number of ADAM iterations ( $N_{iter}$  per segment), time steps per segment, number of collocation points ( $N_r$ ) etc. In the current section we choose the Cahn Hilliard equation as the canonical example for all the analysis performed. The accuracy of the bc-PINN's solution depends on proper choice of these hyper-parameters. To optimize each of the hyper-parameters, we considered various cases and metrics like computational time and accuracy. Table (A.4) describes the optimum parameters required to train 100 steps. The optimum parameters are chosen to achieve an accurate solution while balancing the computational cost as shown in figure (A.13). It can be also seen that as the number of collocation points and number of iterations are increased the accuracy increases. So, we choose an segment size of 10 steps with 5000 collocation points and 10000 iterations per time segment as the computational time required in this case is comparably less than the other cases.

## Appendix B. bc-PINN with a logarithmic residual for Allen Cahn Equation

In this section we show how the bc-PINN with logarithmic residual compares against the standard PINN and bc-PINN without the logarithmic residual. The loss function for the bc-PINN with a logarithmic residual is same as the bc-PINN except the equation(15) is replaced by the following loss term for the residual of the PDE:

$$R := \hat{h}_t - c_1^2 \nabla^2 \hat{h} + f(\hat{h})$$

$$\text{MSE}_R^{(\ln)} = \frac{1}{N_r} \sum_{k=1}^{N_r} \ln (1 + (R(x_k^r, t_k^r))^2) , \quad (x_k^r, t_k^r) \in \Omega \times (T_{n-1}, T_n] \quad (\text{B.1})$$

Model	Time steps/segment	$N_r$	$N_{iter}$
A	10	5000	10000
B	10	5000	20000
C	10	10000	10000
D	10	10000	20000
E	25	5000	10000
F	25	5000	20000
G	25	10000	10000
H	25	10000	20000

Table A.4: Parameter combinations for choosing the optimum segment size, collocation points and number of ADAM iterations to apply the bc-PINN technique for Cahn Hilliard equation.

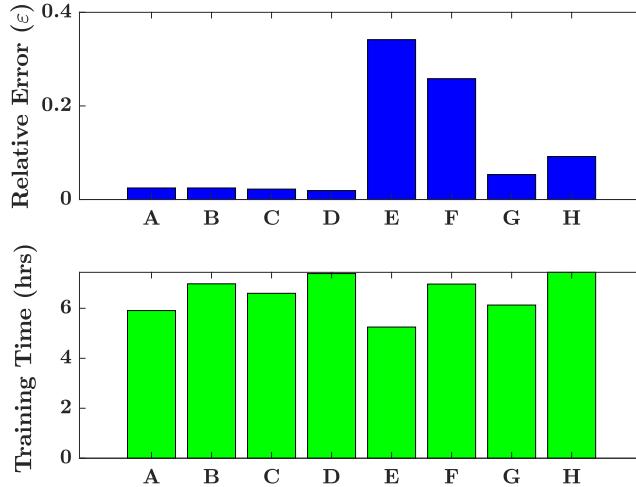


Figure A.13: Relative error ( $\varepsilon$ ) and time taken for various models given in table (A.4).

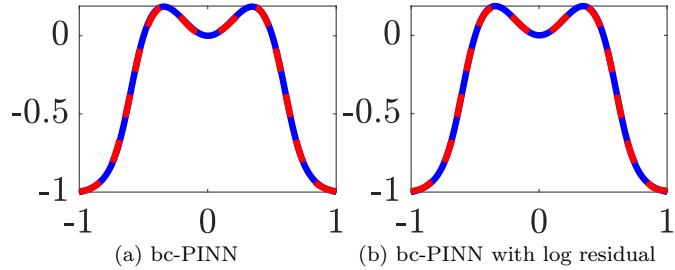


Figure B.14: Exact (—) and Predicted (- - -) solution at time  $t = 0.25$

It turns out that the bc-PINN with a logarithmic residual is more accurate than the bc-PINN. A possible explanation is that the logarithmic function reduces the relative weight on the  $MSE_R$ , which would have larger inaccuracy due to its derivative and nonlinear terms. Thus the initial and boundary terms are satisfied more accurately, which in turn yields a more accurate solution.

Method	Error( $\varepsilon$ )
Standard PINN	0.9919
bc-PINN	0.0701
bc-PINN with logresidual	0.03

Table B.5: Relative errors (equation (7)) over the entire domain with respect to Chebfun solution for different methods.

This explanation is substantiated by the fact that when the logarithmic function is used on all of the four terms in the loss function then the accuracy decreases.

### Appendix C. Minimization of the bc-PINN loss function

In section 3.2 we have described we have mentioned about the learning rates and stopping criteria for the ADAM and LBFGS optimizer are utilized to train the bc-PINN. Here, the minimization of loss function (equation (24)) for training the bc-PINN in time segment [0.45, 0.5] is given in figure (C.15).

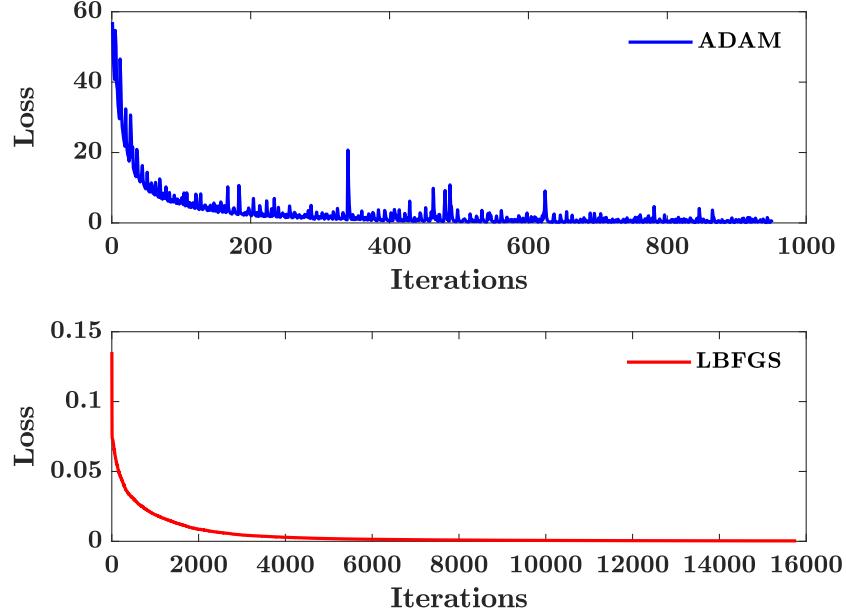


Figure C.15: Loss vs Iterations (Top); using ADAM optimizer (Bottom); using the LBFGS optimizer for training the time segment [0.45, 0.5] of the Cahn Hilliard equation.

## References

- [1] Z. Mao, A. D. Jagtap, G. E. Karniadakis, Physics-informed neural networks for high-speed flows, *Computer Methods in Applied Mechanics and Engineering* 360 (2020) 112789.
- [2] H. Arbabi, J. E. Bunder, G. Samaey, A. J. Roberts, I. G. Kevrekidis, Linking machine learning with multiscale numerics: Data-driven discovery of homogenized equations, *JOM* 72 (12) (2020) 4444–4457.
- [3] M. Raissi, P. Perdikaris, G. E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, *Journal of Computational Physics* 335 (2017) 736–746.
- [4] M. Raissi, P. Perdikaris, G. E. Karniadakis, Machine learning of linear differential equations using Gaussian processes, *Journal of Computational Physics* 348 (2017) 683–693.
- [5] M. Raissi, G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, *Journal of Computational Physics* 357 (2018) 125–141. [arXiv:1708.00588](#).
- [6] S. Atkinson, N. Zabaras, Structured bayesian gaussian process latent variable model: Applications to data-driven dimensionality reduction and high-dimensional inversion, *Journal of Computational Physics* 383 (2019) 166 – 195.
- [7] I. Bilionis, N. Zabaras, B. A. Konomi, G. Lin, Multi-output separable gaussian process: Towards an efficient, fully bayesian paradigm for uncertainty quantification, *Journal of Computational Physics* 241 (2013) 212 – 239.
- [8] 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, *Journal of Computational Physics* 378 (2019) 686–707.
- [9] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, *Journal of machine learning research* 18 (2018).
- [10] D. Zhang, L. Lu, L. Guo, G. E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, *Journal of Computational Physics* 397 (2019) 108850.
- [11] L. Yang, D. Zhang, G. E. Karniadakis, Physics-informed generative adversarial networks for stochastic differential equations, *SIAM Journal on Scientific Computing* 42 (1) (2020) A292–A317.
- [12] X. Meng, G. E. Karniadakis, A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems, *Journal of Computational Physics* 401 (2020) 109020.
- [13] A. Jagtap, E. Kharazmi, G. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, *Computer Methods in Applied Mechanics and Engineering* 365 (2020) 113028.
- [14] S. Karumuri, R. Tripathy, I. Bilionis, J. Panchal, Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks, *Journal of Computational Physics* 404 (2020) 109120. [arXiv:1902.05200](#).

- [15] R. K. Tripathy, I. Bilionis, Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification, *Journal of Computational Physics* 375 (2018) 565–588.
- [16] Y. Yang, P. Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, *Journal of Computational Physics* 394 (2019) 136–152.
- [17] L. Yang, X. Meng, G. E. Karniadakis, B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data (2020). [arXiv:2003.06097](https://arxiv.org/abs/2003.06097).
- [18] Q. He, D. Barajas-Solano, G. Tartakovsky, A. M. Tartakovsky, Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport, *Advances in Water Resources* 141 (2020) 103610.
- [19] M. Liu, L. Liang, W. Sun, A generic physics-informed neural network-based constitutive model for soft biological tissues, *Computer Methods in Applied Mechanics and Engineering* 372 (2020) 113402.
- [20] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks, *Computer Methods in Applied Mechanics and Engineering* 358 (2020) 112623.
- [21] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, E. Kuhl, Physics-informed neural networks for cardiac activation mapping, *Frontiers in Physics* 8 (2020) 42.
- [22] H. Abels, H. Garcke, G. Grün, Thermodynamically consistent, frame indifferent diffuse interface models for incompressible two-phase flows with different densities, *Mathematical Models and Methods in Applied Sciences* 22 (03) (2012) 1150013.
- [23] K. Deckelnick, G. Dziuk, C. M. Elliott, Computation of geometric partial differential equations and mean curvature flow, *Acta Numerica* 14 (2005) 139–232.
- [24] J. Lowengrub, L. Truskinovsky, Quasi incompressible cahn hilliard fluids and topological transitions, *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454 (1978) (1998) 2617–2654.
- [25] Geometric evolution laws for thin crystalline films: Modeling and numerics, *Communications in Computational Physics* 6 (3) (2009) 433–482.
- [26] L. N. Trefethen, N. Hale, T. A. Driscoll, *Chebfun Guide*, Pafnuty Publications, Oxford, 2014.
- [27] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization (2017). [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [28] S. Cox, P. Matthews, Exponential time differencing for stiff systems, *Journal of Computational Physics* 176 (2) (2002) 430–455.
- [29] S. M. Allen, J. W. Cahn, A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, *Acta Metallurgica* 27 (6) (1979) 1085–1095.
- [30] S. Bartels, *Numerical methods for nonlinear partial differential equations*, Vol. 47, Springer, 2015.
- [31] J. Shen, X. Yang, Numerical approximations of allen-cahn and cahn-hilliard equations, *Discrete & Continuous Dynamical Systems-A* 28 (4) (2010) 1669.

- [32] J. W. Cahn, J. E. Hilliard, Free energy of a nonuniform system. i. interfacial free energy, *The Journal of chemical physics* 28 (2) (1958) 258–267.
- [33] A. Miranville, The cahn-hilliard equation and some of its variants, *AIMS Mathematics* 2 (2017) 479–544.
- [34] J. Kim, S. Lee, Y. Choi, S.-M. Lee, D. Jeong, Basic principles and practical applications of the cahn–hilliard equation, *Mathematical Problems in Engineering* 2016 (2016).