Journal of Computational Physics 451 (2022) 110884

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

DeLISA: Deep learning based iteration scheme approximation for solving PDEs

Ying Li [a,b,*], Zuojia Zhou [a], Shihui Ying [c]

[a] School of Computer Engineering and Science, Shanghai University, Shanghai, 200444, PR China
[b] Shanghai Institute for Advanced Communication and Data Science, Shanghai University, Shanghai, 200444, PR China
[c] Department of Mathematics, School of Science, Shanghai University, Shanghai 200444, PR China

ARTICLE INFO

Article history:
Received 14 June 2021
Received in revised form 1 November 2021
Accepted 1 December 2021
Available online 7 December 2021

Keywords:
Deep learning
PDE
Adaptive activation
Iteration scheme approximation

ABSTRACT

Solving the high dimensional partial differential equations (PDEs) with the classical numerical methods is a challenge task. As possessing the power of progressing high dimensional data, deep learning is naturally considered to solve PDEs. This paper proposes a deep learning framework based iteration scheme approximation, called DeLISA. First, we adopt the implicit multistep method and Runge-Kutta method for time iteration scheme. Then, such iteration scheme is approximated by a neural network. Due to integrating the physical information of governing equation into time iteration schemes and introducing time-dependent input, our method achieves the continuous time prediction without a mass of interior points. Here, the activation function with adaptive variable adjusts itself during the iteration process. Finally, we present numerical experiments results for some benchmark PDEs, including Burgers, Allen-Cahn, Schrödinger, carburizing and Black-Scholes equations, and verify that the proposed approach is superior to the state-of-the-art techniques on accuracy and flexibility. Moreover, the Frequency Principle is also illustrated by the changes of prediction at different iterations in this paper.

1. Introduction

PDEs describe many phenomena in fields of physics and finance, such as fluid dynamics [1], stochastic optimal control [2] and mean field games [3]. But only a small number of PDEs have analytical solutions. Thus, a variety of numerical methods are established to approximate analytical solution. The solution space of equations is divided into grids, which are discretized by different numerical methods, such as the finite difference method (FDM) [4], the finite element method (FEM) [5], and the finite volume method (FVM) [6]. The advantage of these methods is that they have solid theoretical analysis. However, there still exists the curse of dimensionality in multi-variable, multi-scale and high-dimensional cases. It is remarkable that the neural networks are able to relatively easily deal with high dimensional data and are successfully applied in many fields [7–10]. In the meantime, according to universal approximation theorem [11], neural networks are proved that it can approximate any continuous function to arbitrary accuracy. Thus, it naturally is considered to solve PDEs.

In recent years, the methods solving PDEs via deep learning have achieved significant progress. First, the stationary equations have been done by the following work. Yang et al. [12] propose a Legendre improved extreme learning machine (L-IELM) to solve elliptic PDEs, which chooses Legendre polynomials as basis functions of hidden neurons, and only possesses

* Corresponding author at: School of Computer Engineering and Science, Shanghai University, Shanghai, 200444, PR China.
E-mail addresses: yinglotus@shu.edu.cn (Y. Li), shu_zhou@shu.edu.cn (Z. Zhou).

https://doi.org/10.1016/j.jcp.2021.110884
0021-9991/

a single hidden layer. Cai et al. [13] focus on the study of various loss functions of neural network, and then come up with a deep FOSLS method, which is efficient for solving second-order elliptic PDEs. In chemistry field, Pfau et al. [14] design a neural network, namely Fermionic neural network (FermiNet), to approximate ground-state wave function of many-electron Schrödinger equation. Liao et al. [15] propose a Deep Nitsche Method based on the framework of Deep Ritz Method to deal with the mixed boundary value problems. By minimizing the Nitsche's variational problem, the Deep Nitsche Method can solve high-dimensional problems up to 100 dimensions. Huang et al. [16] exploit augmented Lagrangian method to reformulate the original problem into a minimax problem, and then use two networks to approximate primal and dual variables respectively. Compared to the penalty method, this method is more accurate and robust. In order to solve the issue shown by Frequency Principle [17] (F-Principle) that the Deep Neural Networks (DNNs) learn the low frequency content of data well but high frequency poorly, Liu et al. [18] make the high frequency of solution convert to lower one by considering a scaling operation along the radial direction in the Fourier space, overcoming the inadequacy that DNNs approximate high frequency solution.

In general, the stationary equations are solved more simply. Recently, the nonstationary equations attract more attention of researchers. In this paper, we also focus on nonstationary equations. Raissi et al. [19] introduce two different types of algorithms called continuous time models and discrete time models. The continuous time models are widely used and mainly depend on vast internal observation points for approximating PDEs' solution and discovering PDEs. Sirignano et al. [20] design a new network architecture like long short term memory (LSTM) networks for a class of quasilinear parabolic PDEs. More importantly, it proves the approximation power of neural networks for these PDEs. In addition, Aradi et al. [21] utilize the network in [20] to solve a series of problems with respect to quantitative finance, such as derivative pricing, optimal investment, optimal execution and mean field games. Nevertheless, they fail to solve shock wave problem (e.g. Burgers equation in [19]). Raissi et al. [19] put forward physics informed neural networks (PINNs) merging data observation with PDE models to solve forward and inverse problems. Many searchers [22–26] further develop PINNs. Raissi [22] approximates the numerical solution by two neural networks when nonlinear dynamics is unknown. For multiple outputs, Haghighat et al. [23] propose a multi-network model, where each network controls one output. Raissi et al. [24,25] propose Navier-Stokes informed neural networks to solve external and internal flow problems. Jagtap et al. [26] employ adaptive activation in PINNs for acceleration of convergence as well as improvement of robustness and accuracy. Meanwhile, they use F-Principle to observe the performance of adaptive activation, where one can see the networks capture the solution from low frequency to high frequency in frequency domain. The F-Principle provides a perspective from the Fourier analysis to study the training process of DNNs, which is defined as follows: *DNNs often fit target functions from low to high frequencies during the training process*. In addition, Xu et al. [27,28] put forward a novel approach called DLGA-PDE, which combines the mutation and crossover of genetic algorithm with deep learning for discovering underlying PDEs.

However, the acquisition of large observation points is expensive, which limits the application of continuous time models in physics field. Therefore, the discrete time models are proposed to address above issues.

As far as we know, solving high dimensional PDEs is difficult by classical time discretization methods, while deep learning provides a solution to this problem. PINNs' discrete time model can take very large time steps while retaining stability and high predictive accuracy. Yet it only can get the terminal time prediction upon a single round of training. For solving backward stochastic differential equations (BSDEs), Han et al. [29] propose a discrete time model, which combines deep learning with the explicit Euler scheme and requires one neural network in each time step. Raissi [30] improves the neural network called forward-backward stochastic neural networks (FBSNNs) to solve forward-backward stochastic differential equations (FBSDEs). For approximation of unknown PDEs, the interesting connection between differentiations and convolutions is established by Cai et al. [31,32], which is further applied to neural networks with temporal discretization in Dong et al. [33], called PDE-Net. In order to improve stability in prediction for a long time, PDE-Net stacks multiple δt-blocks into network, where one δt-block predicts one-step dynamics. But it requires to repeat training many times until completing all blocks. Similar to PDE-Net to some extent, Yao et al. [34] also design a finite element analysis (FEA) framework, called FEA-Net, which is learned from data by proposed FEA convolution. Although these approaches have achieved good results, they only predict one step with one network upon a single round of training.

In order to achieve continuous time prediction of discrete time model upon a single round of training, inspired by FBSNNs, we design a novel time iteration scheme to solve PDEs by combining the deep neural network framework, namely DeLISA. Through introducing time-dependent input, we realize continuous time prediction by one network just upon a single round of training. Specifically, we first adopt higher accurate implicit multistep and Runge-Kutta iteration scheme respectively. Then such iteration scheme is approximated by a neural network, which makes time step larger in the case of remaining accuracy. In this method, we apply automatic differentiation [35] for taking the partial derivative to avoid dimensional disasters. The total loss function is composed of the sum of loss function at each time, which is expressed as the square of difference between solution approximated by network and calculated by time iteration schemes. Finally, we test the performance of the proposed iteration scheme with implicit multistep method on one dimensional PDEs, as well as implicit Runge-Kutta method on two and three dimensional PDEs, validating the effectiveness. Simultaneously, we demonstrate the F-Principle in the solution space by the changes of prediction at different iterations.

By integrating the physical information of governing equation into time iteration schemes, our method avoids to sample a mass of interior points during training but only depends on initial and boundary points. On the other hand, although we apply iteration schemes, our method is no need for mesh grids. To be specific, we sample space points randomly at initial time, and then the solution of these points at any time step are calculated by time iteration schemes. In these schemes, each

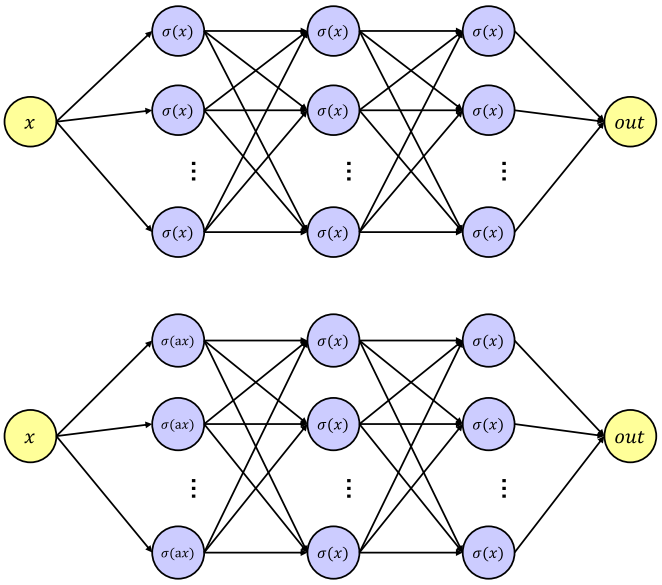


Fig. 1. FNN (top) vs A-FNN (bottom): A-FNN employs adaptive activation function in the first hidden layer, where a is an adaptive variable.

point only depends on itself at previous time step, so as to avoid the generation of mesh grid. In addition, the application of adaptive activation function in the first hidden layer effectively improves convergence rate and accuracy.

This paper is organized as follows: In section 2, we describe the architecture of DeLISA in detail. Section 3 presents the results of numerical experiments, including Burgers, Schrödinger, Allen-Cahn, Carburizing diffusion, Black-Scholes equations. Finally, the conclusions of our work are summarized in Section 4.

2. Methodology

2.1. Problem setup

In this work, we consider a family of PDEs in a bounded domain $\Omega \in \mathbb{R}^d$:

$$u_t(t,x) + \mathcal{L}u(t,x) = 0,\ (t,x) \in [0,T] \times \Omega, \tag{1}$$

where $u(t,x)$ is the latent solution to be solved, $\mathcal{L}$ is differential operator, and u_t denotes first-order derivative of u with respect to t. Initial and boundary conditions are given by practical problem. Our goal is to find above PDEs' solution by the proposed iteration scheme under the neural network framework, which solves high dimensional PDEs with no need for interior points.

2.2. Architecture of DeLISA

In this section, we construct the architecture of DeLISA. Inspired by adaptive activation functions to accelerate convergence of fully-connected neural network (FNN) [26], an adaptive fully-connected neural network (A-FNN) (see Fig. 1) indicated as $f(t,x;\theta)$ is considered to approximate $u(t,x)$, where θ is parameter of A-FNN. To be specific, the adaptive activation function without regard to scale factor is just applied in the first hidden layer. Next, we consider one-step implicit multistep method [36], namely trapezoidal rule, on one dimension:

$$u^{n+1} = u^n - \frac{1}{2}\Delta t(\mathcal{L}u^{n+1} + \mathcal{L}u^n),\ n = 0, ..., N-1, \tag{2}$$

where u^n, that is $u(t_n,x)$, corresponds to solution u at time $n\Delta t$, Δt is time step. Then the approximation $f(t,x;\theta)$ of u is applied in Eq. (2) to construct a new iteration:

$$u^{n+1} \approx f^n - \frac{1}{2}\Delta t(\mathcal{L}f^{n+1} + \mathcal{L}f^n),\ n = 0, ..., N-1, \tag{3}$$

where f^n is output at time $n\Delta t$. The loss function at each time is designed as:

$$Loss_n(\theta) = \left\| u^n - f^n \right\|^2,\ n = 1, ..., N. \tag{4}$$

Then we give the loss of initial condition u^0:

$$Loss_0(\theta) = \left\| u^0(x) - f^0(x;\theta) \right\|^2. \tag{5}$$

The loss of boundary conditions $u^n(x_b)$ is given by:

$$Loss_b(\theta) = \begin{cases} \sum_{n=1}^{N} \left\| u^n(x_b) - f^n(x_b;\theta) \right\|^2, & \text{Dirichlet boundary} \\ \sum_{n=1}^{N} \left\| u_x^n(x_b) - f_x^n(x_b;\theta) \right\|^2, & \text{Neumann boundary.} \end{cases} \tag{6}$$

In terms of periodic boundary conditions, the loss is reformulated as:

$$Loss_b(\theta) = \begin{cases} \sum_{n=1}^{N} \left\| f^n(x_l;\theta) - f^n(x_u;\theta) \right\|^2, & \text{Dirichlet boundary} \\ \sum_{n=1}^{N} \left\| f_x^n(x_l;\theta) - f_x^n(x_u;\theta) \right\|^2, & \text{Neumann boundary,} \end{cases} \tag{7}$$

where x_l is lower bound, x_u is upper bound. Consequently, the total loss function consists of the sum of following three terms:

$$Loss(\theta) = \sum_{n=1}^{N} Loss_n + Loss_0 + Loss_b. \tag{8}$$

Similarly, we consider q-stage implicit Runge-Kutta method [19] on two dimension:

$$\begin{aligned} u^{n,i} &= u^{n+c_i} - \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{L} u^{n+c_j}, \ i = 1, ..., q, \\ u^{n,q+1} &= u^{n+1} - \Delta t \sum_{j=1}^{q} b_j \mathcal{L} u^{n+c_j}. \end{aligned} \tag{9}$$

The multi-output of A-FNN is described as:

$$[f^{n+c_1}, ..., f^{n+c_q}, f^{n+1}]. \tag{10}$$

We substitute Eq. (10) into Eq. (9) to obtain:

$$\begin{aligned} u^{n,i} &\approx f^{n+c_i} - \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{L} f^{n+c_j}, \ i = 1, ..., q, \\ u^{n,q+1} &\approx f^{n+1} - \Delta t \sum_{j=1}^{q} b_j \mathcal{L} f^{n+c_j}. \end{aligned} \tag{11}$$

The loss function at each time is designed as:

$$Loss_n(\theta) = \sum_{i=1}^{q+1} \left\| f^n - u^{n,i} \right\|^2, \ n = 1, ..., N-1 \tag{12}$$

The initial condition loss is given by:

$$Loss_0(\theta) = \sum_{i=1}^{q+1} \left\| u^0 - u^{0,i} \right\|^2. \tag{13}$$

Then the total loss function consists of the sum of following three terms:

$$Loss(\theta) = \sum_{n=1}^{N} Loss_n + Loss_0 + Loss_b. \tag{14}$$

Based on A-FNN and above iteration scheme approximation (ISA), we construct the architecture of DeLISA, which is depicted as Fig. 2. In this architecture, the A-FNN shares the same set of parameters such that the approximation u is predicted at any time by a single round of training. Algorithm 1 shows the proposed implicit multistep iteration scheme.

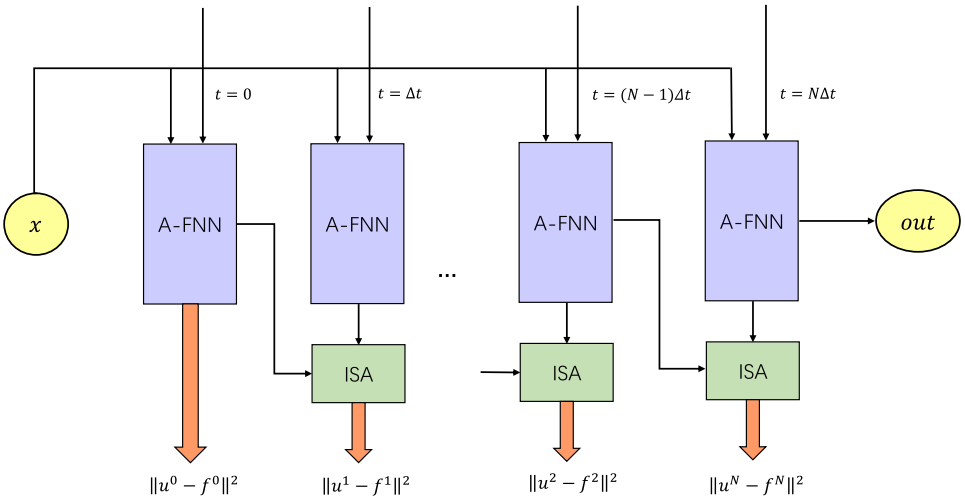


Fig. 2. The architecture of DeLISA.

Algorithm 1 Algorithmic procedure.

Input: Initial training data, (t_0, x); Boundary training data, (t, x_b);
Output: Prediction of A-FNN, $f(t, x; \theta)$;

```
Initialize the parameters of A-FNN;
Compute loss at t_0: Loss_0 = (u^0(x) − f^0(x;θ))^2;
for n = 1 to N do
    get Lf^{n+1}, Lf^n by automatic differential;
    u^{n+1} := f^n − (1/2)Δt(Lf^{n+1} + Lf^n);
    Loss_n = (u^n − f^n)^2;
    Loss_{n,b} = (u^n(x_b) − f^n(x_b;θ))^2;
end for
Minimize the sum of Loss(θ) by gradient descent method;
Get the prediction f(t, x; θ);
return f.
```

3. Numerical experiments

In this section, we consider a series of PDEs to test the performance of our proposed method, including Burgers, Allen-Cahn, Schrödinger, Black-Scholes and carburizing diffusion equations. In addition, we explain F-Principle in the solution space, which reveals deep neural network how to fit target function from low to high frequency during training. Finally, we show the comparison of our method and PINN on accuracy and convergence rate. In our work, we employ L-BFGS-B method to minimize the loss function and NVIDIA RTX 1660 GPU card to train experiments.

3.1. Experiments for the one-dimensional PDEs

3.1.1. Burgers equation

Burgers equations are an important class of nonlinear PDEs, which describe many physical flow phenomena, such as turbulence, boundary layer behavior, shock wave formation. Especially, many researchers pay more attention to the methods for shock wave problems due to its discontinuity. Subsequently, we consider an one-dimensional Burgers equation [19]:

$$\begin{aligned} &u_t = -uu_x + (0.01/\pi)u_{xx},\ x \in [-1, 1],\ t \in [0, 1], \\ &u(0, x) = -sin(\pi x), \\ &u(t, -1) = u(t, 1). \end{aligned} \tag{15}$$

We discretize Eq. (15) by implicit multistep method to obtain:

$$u^{n+1} = u^n + \frac{1}{2}\Delta t[(-u^{n+1}u_x^{n+1} + (0.01/\pi)u_{xx}^{n+1}) + (-u^n u_x^n + (0.01/\pi)u_{xx}^n)],\ n = 0, ..., N. \tag{16}$$

Then applying the output of A-FNN $f(t, x; \theta)$ in Eq. (16):

$$u^{n+1} \approx f^n + \frac{1}{2}\Delta t[(-f^{n+1}f_x^{n+1} + (0.01/\pi)f_{xx}^{n+1}) + (-f^n f_x^n + (0.01/\pi)f_{xx}^n)],\ n = 0, ..., N, \tag{17}$$

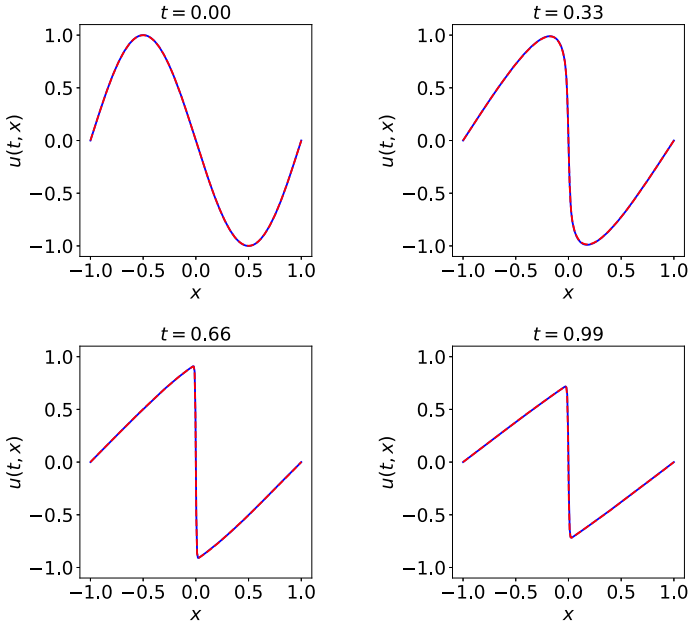


Fig. 3. Burgers equation: The snapshots of predicted solutions (red dashed line) and exact solutions (blue line). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 1
Burgers equation: relative L^2 error between prediction and exact value for different neurons at different time.

t	$l = 10$		$l = 20$		$l = 50$		$l = 100$		$l = 200$	
	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a
0	9.45e-04	1.66	8.16e-04	3.31	7.35e-04	4.62	7.56e-04	3.26	7.67e-04	3.97
0.33	5.99e-03		6.04e-03		2.95e-03		4.90e-03		4.89e-03	
0.66	2.13e-02		3.96e-03		7.35e-03		3.50e-02		2.55e-02	
0.99	1.04e-02		8.61e-03		1.77e-02		4.18e-02		3.32e-02	
All time	1.09e-02		4.87e-03		6.75e-03		2.13e-02		1.53e-02	

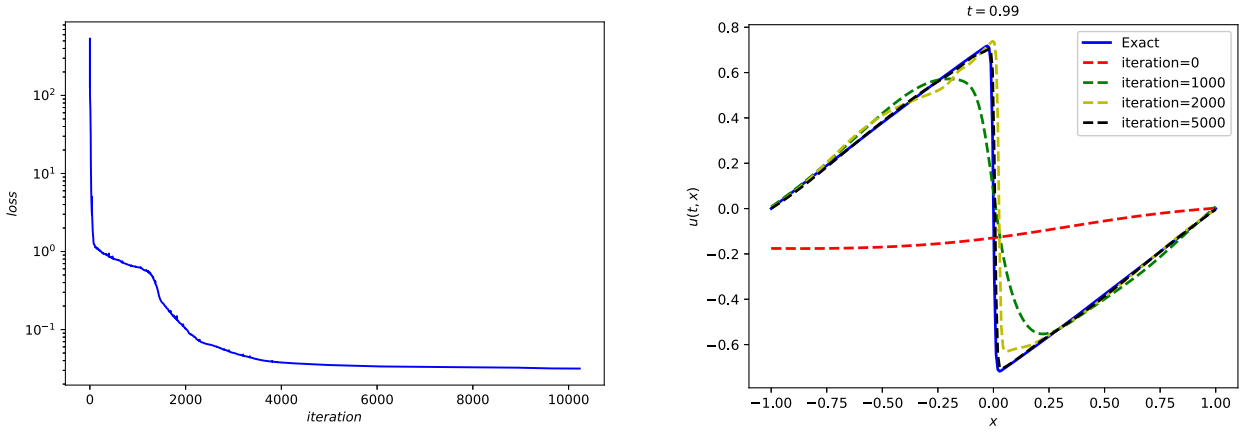
where the partial derivatives of f are calculated by automatic differential. The shared parameters θ of A-FNN are learned by minimizing the following loss function:

$$Loss(\theta) = \sum_{n=1}^{N} \left\| u^n - f^n \right\|^2 + \sum_{n=1}^{N} \left\| f^n(-1;\theta) - f^n(1;\theta) \right\|^2 + \left\| u^0(x) - f^0(x;\theta) \right\|^2. \tag{18}$$

In this experiment, we set 4 hidden layers for A-FNN, where each hidden layer consists of 50 neurons and *tanh* activation function. The adaptive variable a of the first hidden layer is set as 2 and time step $\Delta t = 0.1$. Moreover, the training set is made of initial points $N_0 = 256$ and boundary points N_b at each time step. Fig. 3 shows the comparison of prediction and exact solution at $t = (0, 0.33, 0.66, 0.99)$, which validates that the prediction fits accurately exact solution. Fig. 4 (a) shows the variation trend of loss function as iterations. In order to test the performance of proposed method, we present the errors with different neurons in Table 1, where l represents the size of neurons. Similar to classical discretization methods, we observe the relative L^2 error increases with time. In addition, it is seen from the Fig. 4 (b) that the A-FNN fits solutions from profiles (smoothness) to details (shock waves). As we know, low frequency corresponds to changing slowly area, while high frequency corresponds to changing rapidly area in image processing. Consequently, Fig. 4 (b) also demonstrates F-Principle. Moreover, from Fig. 4, neural network mainly fits shock waves after 5000 iterations, leading to small changes of loss function.

3.1.2. Schrödinger equation

For further validating the effectiveness of the proposed method, we solve Schrödinger equation with complex-value solution. Schrödinger equation is the base equation of quantum mechanics, which is a powerful tool to solve non-relativistic problems in atomic physics.



(a) (b)

Fig. 4. Burgers equation: The left (a) is the changes of loss function with iterations. The right (b) is the predicted solution at $t = 0.99$ corresponding to four snapshots $iteration = (0, 1000, 2000, 5000)$.

We consider the following one-dimensional Schrödinger equation along with periodic boundary conditions [19]:

$$\begin{aligned} &i\psi_t + 0.5\psi_{xx} + |\psi|^2\psi = 0,\ x \in [-5, 5],\ t \in [0, \frac{\pi}{2}], \\ &\psi(0, x) = 2sech(x), \\ &\psi(t, -5) = \psi(t, 5), \\ &\psi_x(t, -5) = \psi_x(t, 5), \end{aligned} \tag{19}$$

where $\psi(t, x)$ is the complex-valued solution, which consists of real part u and imaginary part v. Consequently, the solution $\psi(t, x)$ is written as $\psi(t, x) = [u(t, x), v(t, x)]$, and Eq. (19) is reformulated as:

$$\begin{aligned} &u_t = -0.5v_{xx} - (u^2 + v^2)v, \\ &v_t = 0.5u_{xx} + (u^2 + v^2)u, \\ &u(0, x) = 2sech(x), \\ &v(0, x) = 0, \\ &u(t, -5) = u(t, 5), \\ &v(t, -5) = v(t, 5), \\ &u_x(t, -5) = u_x(t, 5), \\ &v_x(t, -5) = v_x(t, 5). \end{aligned} \tag{20}$$

Correspondingly, we set A-FNN to multi-output $f(t, x; \theta) = [U, V]$. Similar to Eq. (2)-Eq. (3), we obtain the following loss function with shared parameters θ:

$$\begin{aligned} Loss_u(\theta) &= \sum_{n=1}^{N} \left(\left\| u^n - U^n \right\|^2 + \left\| U^n(-5; \theta) - U^n(5; \theta) \right\|^2 + \left\| U_x^n(-5; \theta) - U_x^n(5; \theta) \right\|^2 \right) + \left\| u^0(x) - U^0(x; \theta) \right\|^2, \\ Loss_v(\theta) &= \sum_{n=1}^{N} \left(\left\| v^n - V^n \right\|^2 + \left\| V^n(-5; \theta) - V^n(5; \theta) \right\|^2 + \left\| v_x^n(-5; \theta) - V_x^n(5; \theta) \right\|^2 \right) + \left\| v^0(x) - V^0(x; \theta) \right\|^2, \\ Loss(\theta) &= Loss_u + Loss_v, \end{aligned} \tag{21}$$

where $Loss_u$ is the loss of real part, $Loss_v$ is the loss of imaginary part. The resulting total loss is minimized to obtain the parameter θ.

In this part, the A-FNN has 6 hidden layers, each of which has 50 neurons. The *tanh* activation is adopted and the adaptive variable a is 1. The training data is composed of initial points $N_0 = 512$ and boundary points N_b at each time step $\Delta t = \pi \cdot 10^{-3}$. Fig. 5 shows the prediction $|\psi| = \sqrt{U^2 + V^2}$ at $t = (0, 0.31, 0.63, 0.79)$ with time step $4\Delta t$, which

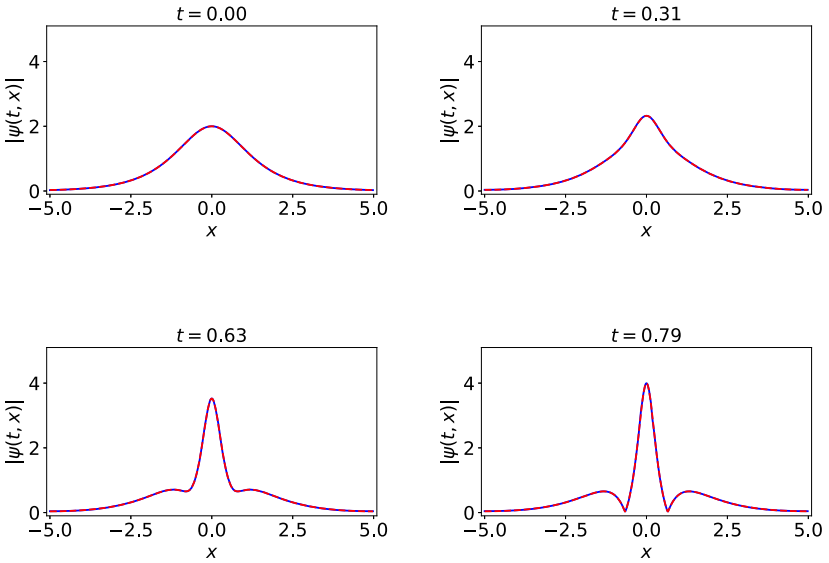
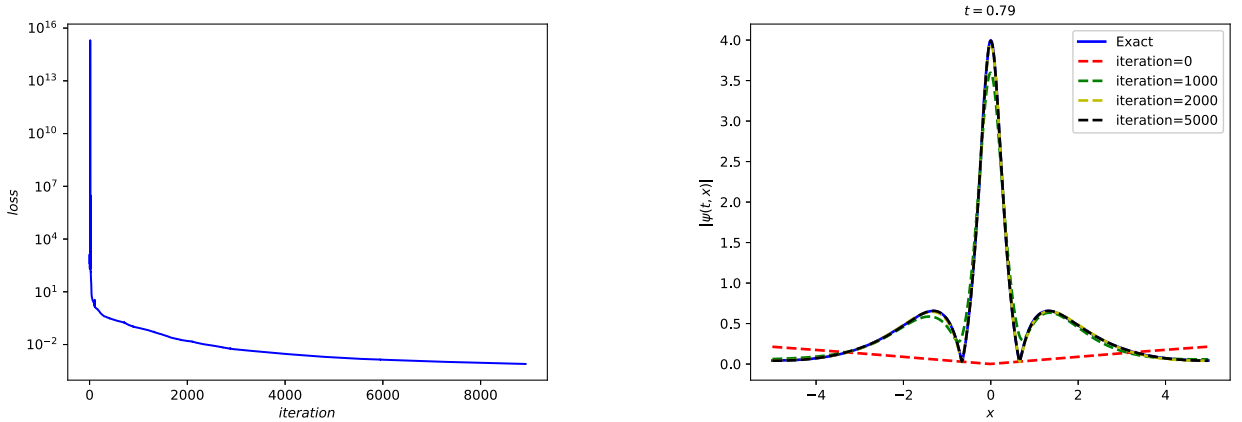


Fig. 5. Schrödinger equation: The snapshots of the predicted solutions (red dashed line) and exact solutions (blue line).



(a) (b)

Fig. 6. Schrödinger equation: The left (a) is the changes of loss function with iterations. The right (b) is the predicted solution at $t = 0.79$ corresponding to four snapshots $iteration = (0, 1000, 2000, 5000)$.

simulates exact solution accurately. Fig. 6 (a) shows the variation trend of loss function as iterations. In order to probe into the impact of time step on accuracy, we employ different time steps for this experiment. Table 2 shows the results with different time steps, which is obvious that the relative L^2 error takes the minimum at time step $4\Delta t$. It is worth noting that the proposed method reflects property different from classical discretization methods on account of applying A-FNN. For classical discretization methods, the smaller the time step, the higher the accuracy, while the method based on A-FNN achieves the optimum at larger time step, which is also an advantage compared with time discretization methods. In addition, Fig. 6 (b) shows the fitting process of A-FNN at $t = 0.79$. We observe that the "sharp turns" are approximated as iterations increases, which is similar to observation of Fig. 4 (b). Consequently, Fig. 6 (b) also demonstrates F-Principle. Moreover, as depicted in Fig. 6, after 2000 iterations, neural network mainly fits "sharp turns", which results in small changes of loss function.

3.1.3. Black-Scholes equation

The Black-Scholes model is a well-known option pricing model for European options, which must be exercised by the buyer on the expiration date. It has laid a foundation for the reasonable pricing of various derivative financial instruments based on market price changes in the emerging derivative financial markets, including stocks, bonds, currencies and commodities.

Table 2
Schrödinger equation: relative L^2 error between prediction and exact value for different time steps at different time.

t	$2\Delta t$		$3\Delta t$		$4\Delta t$		$6\Delta t$	
	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a
0	9.51e-04	2.13	9.05e-04	1.90	7.35e-04	1.57	8.51e-04	1.86
0.31	2.10e-03		1.76e-03		1.07e-03		1.77e-03	
0.63	4.01e-03		3.78e-03		2.74e-03		5.62e-03	
0.79	3.68e-03		3.83e-03		3.05e-03		5.97e-03	
All time	2.68e-03		2.46e-03		1.76e-03		3.15e-03	

Table 3
Black-Scholes equation: relative L^2 error between prediction and exact value for different time steps at different time.

t	Δt		$2\Delta t$		$4\Delta t$		$5\Delta t$	
	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a
0	1.19e-03	1.85	1.73e-03	1.63	2.23e-03	1.38	4.48e-03	0.95
0.33	8.09e-04		1.35e-03		1.39e-03		3.96e-03	
0.66	7.42e-04		1.21e-03		1.30e-03		3.11e-03	
1	2.47e-03		1.61e-03		1.64e-03		2.71e-03	
All time	9.03e-04		1.25e-03		1.42e-03		3.49e-03	

We consider a Black-Scholes equation for European Call Options [21]:

$$\begin{aligned} &g_t + rxg_x + \frac{1}{2}\sigma^2 x^2 g_{xx} = rg,\ x \in [0, 100],\ t \in [0, 1], \\ &g(T, x) = G(x), \\ &g(t, 0) = 0, \\ &g(t, 100) = 50, \end{aligned} \tag{22}$$

where $g(t, x)$ is claim's price function determining the value of the claim at time t, $G(x) = max(0, x - K)$, K is option's strike price, σ is volatility, r is interest rate, T is terminal time. The analytic solution is given by:

$$\begin{aligned} &g(t, x) = x\Phi(d_+) - Ke^{-r(T-t)}\Phi(d_-), \\ &d_+ = \frac{ln(x/K) + (r + \frac{1}{2}\sigma^2)(T - t)}{\sigma\sqrt{T - t}}, \\ &d_- = d_+ - \sigma\sqrt{T - t}, \end{aligned} \tag{23}$$

where Φ denotes the standard normal distribution function.

Similar to Eq. (2)-Eq. (3), we obtain and minimize the following loss function with shared parameters θ:

$$Loss(\theta) = \sum_{n=0}^{N-1} \left(\left\| g^n - f^n \right\|^2 + \left\| g^n(0; \theta) \right\|^2 + \left\| g^n(100; \theta) - 50 \right\|^2 \right) + \left\| g^T(x) - f^T(x; \theta) \right\|^2. \tag{24}$$

In this example, the training dataset consists of $N_T = 101$ terminal data points and boundary points N_b at each time step $\Delta t = 0.01$, and set $r = 5\%$, $\sigma = 25\%$, $K = 50$. We use *tanh* activation for A-FNN with 4 hidden layers and 20 neurons per layer. The adaptive variable a is initialized as 0.5 in the first hidden layer. Fig. 7 presents the comparison between exactness and prediction at $t = (0, 0.33, 0.66, 1)$ with time step Δt, which is obvious that they have the same trajectory. Fig. 8 (a) shows the variation curve of loss function, which gets smoother and smoother during training. Table 3 shows the results with different time steps. Different from previous examples, the terminal condition replaces initial condition, leading to error not increasing with time but decreasing and then increasing. This interesting observation may be caused by terminal condition, which deserves further study. In addition, Fig. 8 (b) portrays the fitting curves of different iterations at T. Analogous to observation in Fig. 4 (b), the "sharp turns" in Fig. 8 (b) are fitted as iterations increase, which also demonstrates F-Principle. Moreover, after 500 iterations, it is obvious that the loss function gradually gets smoother due to neural network mainly fitting "sharp turns" in Fig. 8.

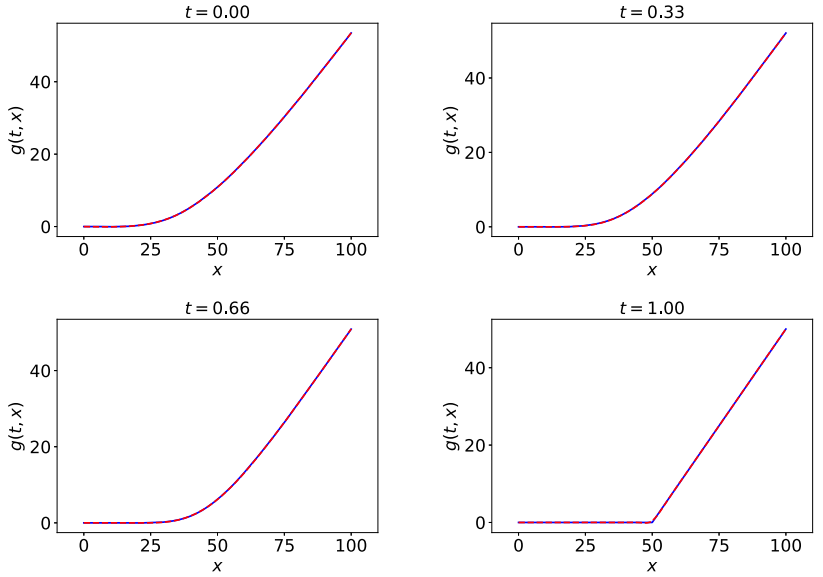
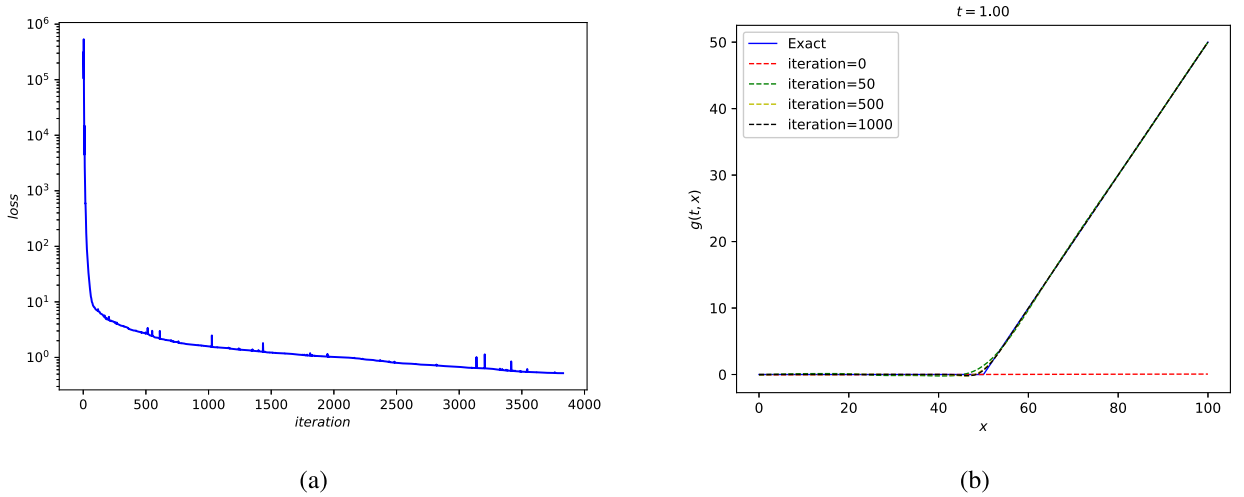


Fig. 7. Black-Scholes equation: The snapshots of the predicted solutions (red dashed line) and exact solutions (blue line).



(a) (b)

Fig. 8. Black-Scholes equation: The left (a) is the changes of loss function with iterations. The right (b) is the predicted solution at $t = 0$ corresponding to four snapshots $iteration = (0, 50, 100, 500)$.

3.1.4. Allen-Cahn equation

Allen-Cahn equations describe the antiphase boundary motion in a crystal. They are a kind of important equations for describing fluid dynamics and reaction diffusion problems in material science. Next, let us consider an Allen-Cahn equation with periodic boundary conditions [19]:

$$\begin{aligned} &u_t - 0.0001u_{xx} + 5u^3 - 5u = 0,\ x \in [-1, 1],\ t \in [0, 1], \\ &u(0, x) = x^2 cos(\pi x), \\ &u(t, -1) = u(t, 1), \\ &u_x(t, -1) = u_x(t, 1). \end{aligned} \tag{25}$$

Similar to Eq. (2)-Eq. (3), we obtain and minimize the following loss function with shared parameters θ:

$$Loss(\theta) = \sum_{n=1}^{N} \left(\left\| u^n - f^n \right\|^2 + \left\| f^n(-1; \theta) - f^n(1; \theta) \right\|^2 + \left\| f_x^n(-1; \theta) - f_x^n(1; \theta) \right\|^2 \right) + \left\| u^0(x) - f^0(x; \theta) \right\|^2. \tag{26}$$

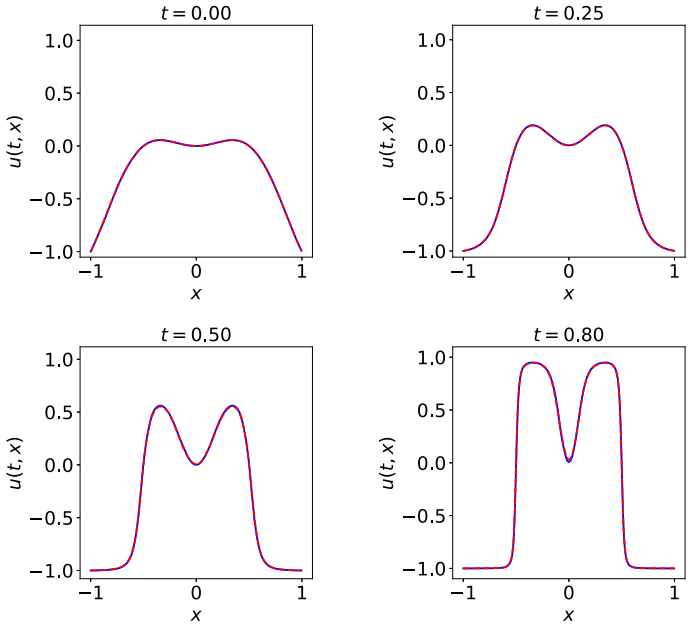


Fig. 9. Allen-Cahn equation: The snapshots of the predicted solutions (red dashed line) and exact solutions (blue line).

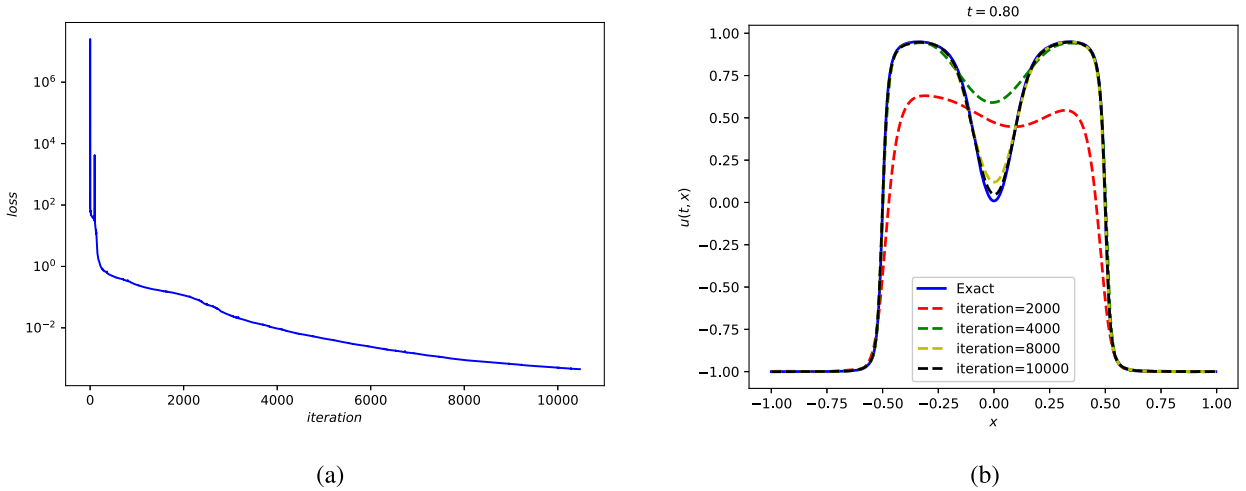


Fig. 10. Allen-Cahn equation: The left (a) is the changes of loss function with iterations. The right (b) is the predicted solution at $t = 0.8$ corresponding to four snapshots $iteration = (2000, 4000, 8000, 10000)$.

In this experiment, the A-FNN architecture consists of 4 hidden layers with 200 neurons in each layer and *tanh* activation function. The adaptive variable a of the first hidden layer is set as 2 and time step $\Delta t = 5 \cdot 10^{-3}$. Moreover, the training set is made of initial points $N_0 = 512$ and boundary points N_b at each time step. In order to further test the performance under the influence of time step, the mixed time steps are employed in a single training. Here, we define the mixed time steps as adopting different time steps on the time domain divided by some time points. Fig. 9 illustrates the accuracy of the prediction u at $t = (0, 0.25, 0.5, 0.8)$ with time step $10\Delta t$ and $5\Delta t$, where $10\Delta t$ is used before $t = 0.5$, $5\Delta t$ is used after $t = 0.5$. Fig. 10 (a) plots the variation trend of loss function as iterations. Table 4 shows the results with different time steps. Obviously, the relative L^2 error takes the minimum at mixed time steps $10\Delta t$ and $5\Delta t$, indicating the mixed time steps are superior to the single time step. The reason for this situation is that large time step gets small loss in the smooth area, while small time step gets small loss in the "sharp turns" area. That means if the point-in-time at "sharp turns" is determined, we employ the different time steps to deal with it for acquiring the smaller error. In addition, Fig. 10 (b) shows the fitting process of A-FNN at $t = 0.8$. We obtain the observation from it similar to Fig. 4 (b). Consequently, Fig. 10 (b) also demonstrates F-Principle. After 8000 iterations, it is clear from previous examples that neural network mainly approximates "sharp turns", leading to slow decrease of loss function.

Table 4
Allen-Cahn equation: relative L^2 error between prediction and exact value for different time steps at different time.

t	$10\Delta t\&5\Delta t$		$5\Delta t$		$10\Delta t$		$2\Delta t$	
	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a
0	1.30e-03	3.29	1.23e-03	3.37	2.89e-03	2.28	2.40e-03	2.48
0.25	4.21e-03		6.69e-03		1.19e-02		3.78e-03	
0.5	8.08e-03		1.77e-02		3.16e-02		1.01e-03	
0.8	8.58e-03		5.09e-02		8.36e-02		2.11e-03	
All time	7.25e-03		2.61e-02		4.45e-02		1.29e-01	

Table 5
Carburizing diffusion equation with constant: relative L^2 error between prediction and exact value for different time steps at different time.

t	$\Delta t\&4\Delta t$		$\Delta t\&8\Delta t$		$\Delta t\&10\Delta t$		$\Delta t\&20\Delta t$	
	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a
0.1h	1.59e-02	5.07	1.47e-02	5.30	1.49e-02	4.96	1.08e-02	4.88
0.5h	5.49e-03		6.65e-03		6.27e-03		7.59e-03	
1h	5.35e-03		8.72e-03		5.61e-03		6.60e-03	
5h	4.02e-03		5.20e-03		2.31e-03		3.18e-03	
All time	9.25e-03		1.04e-02		8.44e-03		8.28e-03	

3.1.5. Carburizing diffusion equation

The carburizing treatment refers to a common chemical heat treatment process, in which the steel is heated and insulated in a carburizing medium. This process increases the carbon content of the surface layer of the steel and forms a certain carbon concentration gradient. The carburizing treatment process is expressed as a diffusion equation with constant diffusion coefficient given by [37]:

$$\begin{aligned} &u_t - (D(u)u_x)_x = 0,\ x \in [0, 2.5],\ t \in [0, 18000], \\ &u(0, x) = u^0, \\ &u(t, 0) = u_l, \\ &u(t, 2.5) = u_b, \end{aligned} \tag{27}$$

where u is the carbon concentration, $D(u)$ is the carbon diffusivity depending on concentration, which is given by:

$$D(u) = D_0 e^{-Q/(RT)}, \tag{28}$$

where D_0 is the pre-exponential factor, Q is the activation energy of carbon, R denotes gas constant, and T means carburizing temperature. The analytical solution is:

$$u(t, x) = c_p - (c_p - c_0) erf\left(\frac{x}{2\sqrt{Dt}}\right). \tag{29}$$

Similar to Eq. (2)-Eq. (3), we obtain and minimize the following loss function with shared parameters θ:

$$Loss(\theta) = \sum_{n=1}^{N} \left(\left\| u^n - f^n \right\|^2 + \left\| f^n(0; \theta) - u_l \right\|^2 + \left\| f_x^n(2.5; \theta) - u_b \right\|^2 \right) + \left\| u^0(x) - f^0(x; \theta) \right\|^2. \tag{30}$$

In this experiment, we use $D_0 = 16.2mm^2/s$, $Q = 137800 J/mol$, $R = 8.314 J/(K \cdot mol)$, $c_p = 1.2$, $c_0 = 0.2$, $T = 1123K$. u^0, u_l and u_b is generated by Eq. (29). The architecture of A-FNN consists of 4 hidden layers with 100 neurons and *tanh* activation function. The adaptive variable a in the first hidden layer is initialized as 3 and time step $\Delta t = 18s$. Moreover, the initial points $N_0 = 201$ and boundary points N_b at each time step are used during training. Fig. 11 exhibits the numerical solutions u fit accurately ground truth at $t = (0.1h, 0.5h, 1h, 5h)$ with mixed time steps $\Delta t\&20\Delta t$, where Δt is used before $t = 1h$, $20\Delta t$ is used after $t = 1h$. Fig. 12 (a) shows the variation trend of loss function as iterations. Table 5 shows the results with different time steps and the relative L^2 error takes the minimum at mixed time steps Δt and $20\Delta t$. It means the large time step is superior to the small time step in the smooth area. Fig. 12 (b) presents the trend of approximating solution at $t = 0.1h$, which is similar to Fig. 4 (b). Consequently, F-Principle is also explained by Fig. 12 (b). Moreover, being attributed to neural network mainly fitting "sharp turns" in Fig. 12 after 2000 iteration, the loss changes little.

Next, we consider the other Carburizing diffusion equation with variable diffusion coefficient simulating the diffusion process of Cu-Ni system for advanced steels:

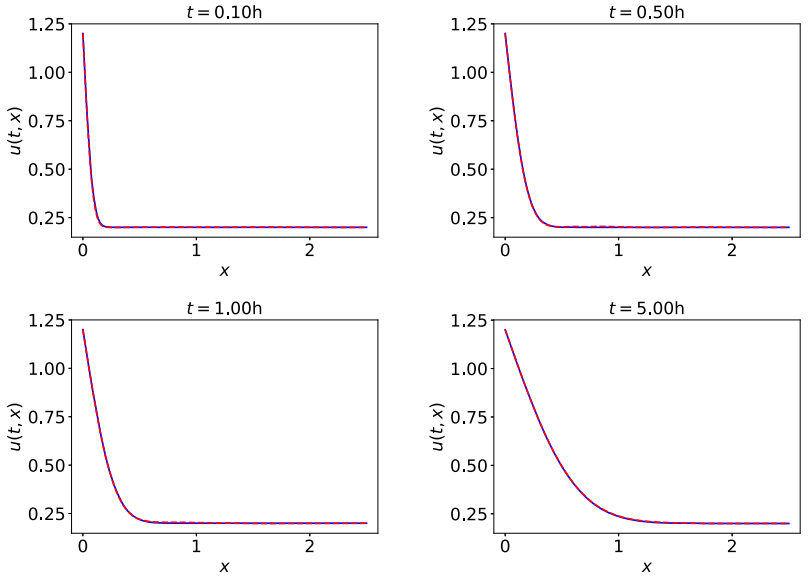
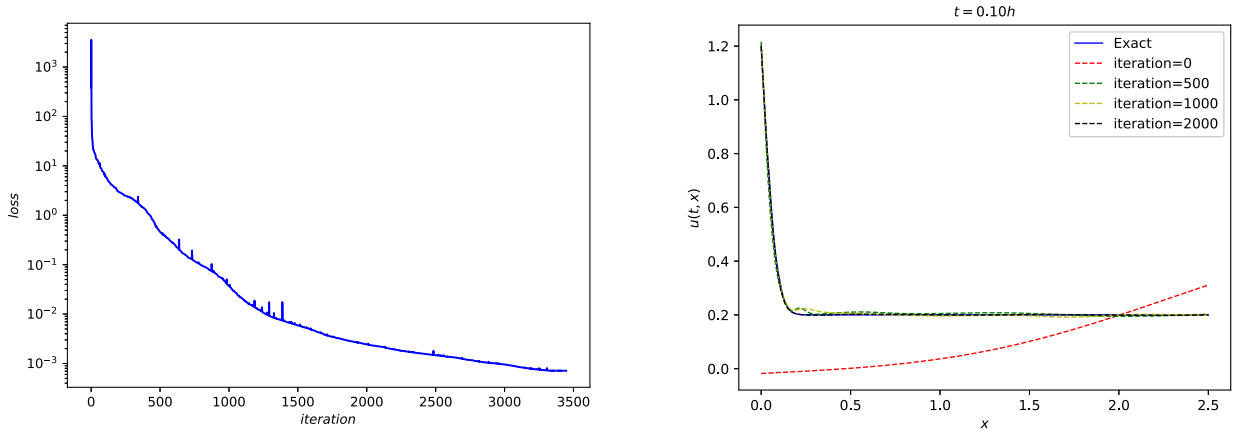


Fig. 11. Carburizing diffusion equation with constant: The snapshots of the predicted solutions (red dashed line) and exact solutions (blue line).



(a) (b)

Fig. 12. Carburizing diffusion equation with constant: The left (a) is the changes of loss function with iterations. The right (b) is the predicted solution at $t = 0.1h$ corresponding to four snapshots $iteration = (0, 500, 1000, 2000)$.

$$\begin{aligned} &u_t - (D(u)u_x)_x = s,\ x \in [-\pi, \pi],\ t \in [0, 1], \\ &u(0, x) = u^0, \\ &u(t, 0) = u_l, \\ &u(t, 1) = u_b, \end{aligned} \tag{31}$$

where $D(u) = \cos u$, the source term is given by:

$$s(t, x) = sin(e^{-dt})e^{-2dt}cos^2x - de^{-dt}sinx + cos(e^{-dt}sinx)e^{-dt}sinx, \tag{32}$$

where $d = 0.5$, and initial and boundary points are obtained by the following analytic solution:

$$u(t, x) = e^{-dt}sinx. \tag{33}$$

In this example, the A-FNN possesses 4 hidden layers with 100 neurons per layer and *tanh* activation function. We set the initial value of the adaptive variable a in the first hidden layer as 2 and time step $\Delta t = 0.005$. Moreover, the initial points $N_0 = 201$ and boundary points N_b at each time step are used as training data. Fig. 13 compares the prediction to

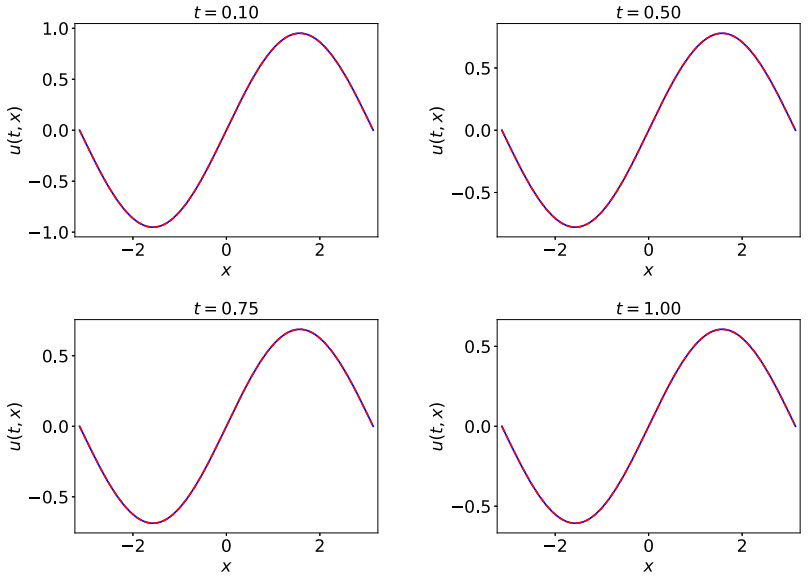


Fig. 13. Carburizing diffusion equation with variable: The snapshots of the predicted solutions (red dashed line) and exact solutions (blue line).

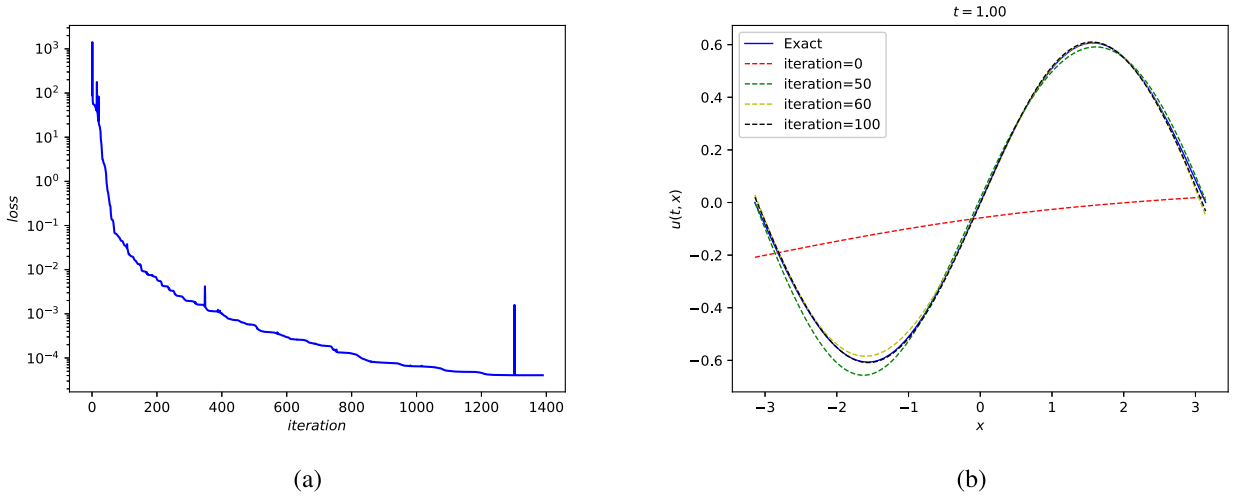


Fig. 14. Carburizing diffusion equation with variable: The left (a) is the changes of loss function with iterations. The right (b) is the predicted solution at $t = 0.1h$ corresponding to four snapshots $iteration = (0, 50, 60, 100)$.

ground truth at $t = (0.1, 0.5, 0.75, 1)$ with time step $20\Delta t$, demonstrating the proposed method is effective. As is shown in Fig. 14 (a), the loss function decreases as iterations, and Fig. 14 (b) shows the A-FNN how to fit exact solution at $t = 1$. Table 6 shows that the relative L^2 error takes the minimum at time step $20\Delta t$. Obviously, the error by the large time step is superior to by the small time step in the smooth area. We observe that the network convergences rapidly only about 1500 iterations, and "turns" are approximated as iterations increase, which is similar to observation of Fig. 4 (b). Consequently, Fig. 14 (b) also demonstrates F-Principle. Moreover, from Fig. 14, we observe that neural network mainly fits "turns" after 100 iterations, which causes slow decrease of loss function.

3.2. Experiments for two-dimensional PDEs

3.2.1. Burgers equation

For exploring the availability to other time discretization method, we apply q stage implicit Runge-Kutta method to solve two-dimensional PDEs. Here, we first consider a two-dimensional Burgers equation [38]:

Table 6
Carburizing diffusion equation with variable: relative L^2 error between prediction and exact value for different time steps at different time.

t	Δt		$4\Delta t$		$5\Delta t$		$10\Delta t$		$20\Delta t$	
	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a
0.1	1.03e-03	1.75	4.11e-04	1.79	6.72e-04	1.85	3.90e-04	1.76	1.61e-04	1.89
0.5	1.87e-03		2.13e-04		9.56e-04		8.11e-04		2.04e-04	
0.75	2.32e-03		1.10e-04		1.10e-03		5.17e-04		2.69e-04	
1	2.30e-03		4.90e-05		1.62e-03		8.53e-04		3.29e-04	
All time	1.73e-03		8.41e-04		6.72e-04		4.39e-04		2.06e-04	

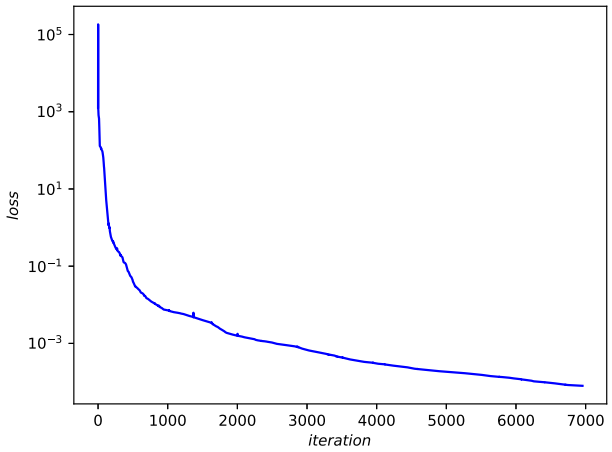


Fig. 15. Two-dimensional Burgers equation: The changes of loss function with iterations.

$$\begin{aligned} &u_t + u(u_x + u_y) = 0.1(u_{xx} + u_{yy}),\ (x, y) \in [0, 1] \times [0, 1],\ t \in [0, 2] \\ &u(0, x, y) = u^0, \\ &u(t, x_b, y_b) = u_b, \end{aligned} \tag{34}$$

where (x_b, y_b) is composed of $(0, y), (x, 0), (1, y), (x, 1)$, the initial condition u^0 and boundary conditions u_b are generated by analytical solution:

$$u(x, y, t) = \frac{1}{1 + e^{\frac{x+y-t}{0.2}}}. \tag{35}$$

Similar to Eq. (9)-Eq. (11), we obtain and minimize the following loss function with shared parameters θ:

$$Loss(\theta) = \sum_{n=1}^{N} \sum_{i=1}^{q+1} \left\| f^n - u^{n,i} \right\|^2 + \sum_{n=1}^{N} \left\| u_b - f^n(x_b, y_b; \theta) \right\|^2 + \sum_{i=1}^{q+1} \left\| u^0(x, y) - u^{0,i}(x, y; \theta) \right\|^2. \tag{36}$$

In this example, the A-FNN contains 6 hidden layers with 50 units per layer and *tanh* activation function. We determine the initial adaptive variable a as 0.4 in the first hidden layer, the stage $q = 4$ and time step $\Delta t = 0.01$. Moreover, the initial points $N_0 = 30 \times 30$ and boundary points N_b at each time step are used during training. Fig. 16 shows the prediction accuracy at $t = (0, 0.5, 1.2, 2)$ with time step $10\Delta t$. As depicted in Fig. 15, we see that the curve of loss function decreases from steep to gentle as iterations. Table 7 shows the results with different time steps, from which the relative L^2 error takes the minimum at time step $10\Delta t$. It also reveals when the proposed approach is applied in the smooth domain, using the large time step performs better than the small time step. In addition, Table 8 shows the relative L^2 error at different stages, and the stage q has little impact on total error. It seems to be explained by the increasing complexity of the model due to high stage. To be specific, high stage increases the number of terms of loss function and cancels out the high precision of the model itself.

3.2.2. Allen-Cahn equation

Next, let us consider a two-dimensional Allen-Cahn equation with the double well potential, which is the fundamental equation of phase-field models [39]:

$$\phi_t - \epsilon^2 \Delta\phi + f(\phi) = 0,\ x \in \Omega, \tag{37}$$

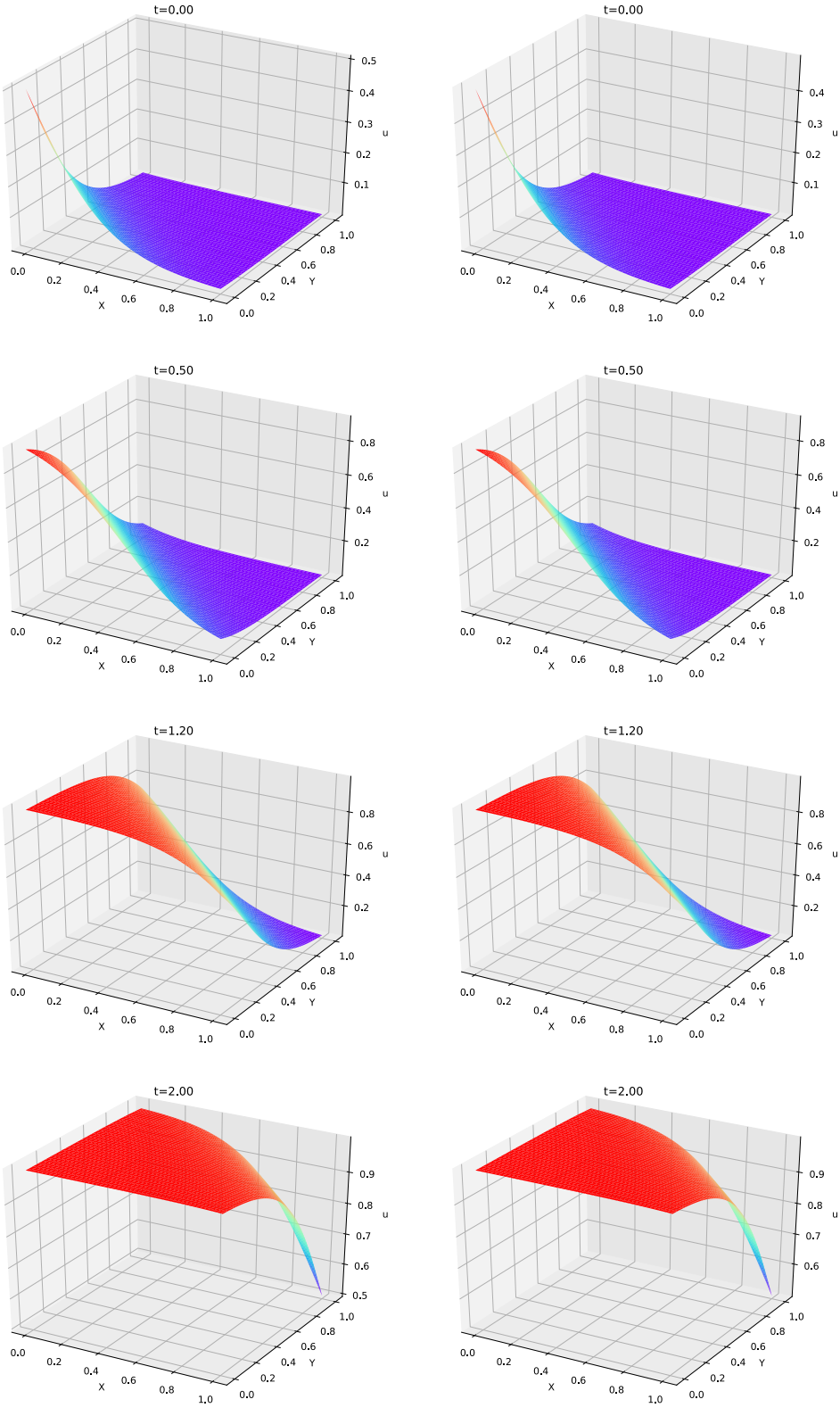


Fig. 16. Two-dimensional Burgers equation: The snapshots of predicted solutions (left column) and exact solutions (right column).

Table 7
Two-dimensional Burgers equation: relative L^2 error between prediction and exact value for different time steps at different time.

t	Δt		$4\Delta t$		$5\Delta t$		$10\Delta t$		$20\Delta t$	
	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a
0	1.09e-03	0.46	3.98e-04	0.49	5.26e-04	0.42	3.13e-04	0.42	7.04e-04	0.57
0.5	2.78e-04		2.13e-04		2.19e-04		1.15e-04		1.60e-04	
1.2	1.98e-04		1.10e-04		9.59e-05		5.48e-05		4.64e-05	
2	7.63e-05		4.90e-05		7.88e-05		5.05e-05		6.70e-05	
All time	2.15e-04		1.59e-04		1.53e-04		8.51e-05		1.23e-04	

Table 8
Two-dimensional Burgers equation: relative L^2 error between prediction and exact value for different stages q with time step Δt.

t	$q=1$		$q=2$		$q=4$		$q=6$	
	L^2 error	a	L^2 error	a	L^2 error	a	L^2 error	a
0	1.67e-03	0.33	1.12e-03	0.38	1.09e-03	0.46	9.39e-04	0.28
0.5	2.88e-04		4.09e-04		2.78e-04		2.75e-04	
1.2	1.60e-04		1.45e-04		1.98e-04		1.21e-04	
2	1.30e-04		1.25e-04		7.63e-05		1.09e-04	
All time	2.08e-04		2.75e-04		2.15e-04		2.10e-04	

where ϵ is a positive constant measuring the interfacial width, ϕ is the phase variable and $f(\phi) = F'(\phi)$ is the chemical potential. The free energy function $F(\phi)$, that is double well potential, is given by:

$$F(\phi) = \frac{1}{4}(1-\phi^2)^2,\ \phi \in [-1, 1]. \tag{38}$$

In this example, we endow A-FNN with 8 hidden layers with 50 neurons per layer and *tanh* activation. Meanwhile, the initial adaptive variable is set as 9, the stage $q = 4$. Next, we consider two cases according to different initial conditions respectively. One is the square [40]:

$$\phi_0 = \begin{cases} 1, & |x| \le 0.5,\ |y| \le 0.5, \\ -1, & \text{else.} \end{cases} \tag{39}$$

For this problem, we set $\epsilon = 0.5$, time step $\Delta t = 0.1$, and initial points 50×50 in Ω, which is depicted in Fig. 17 (a). Fig. 17 illustrates the shrinking process predicted by A-FNN from square to circle. And the result of adaptive variable is adjusted as 8.31868. In comparison with classical numerical method employing small time step [41], DeLISA achieves good results even in the case of taking larger time step.

The other initial condition is two kissing bubbles:

$$\phi_0 = \begin{cases} 1, & (x-\pi+1)^2 + (y-\pi)^2 \le 1 \text{ or } (x-\pi-1)^2 + (y-\pi)^2 \le 1, \\ -1, & \text{else.} \end{cases} \tag{40}$$

For this problem, we set $\epsilon = 0.25$, time step $\Delta t = 0.1$, and initial points 80×80 in Ω, which is depicted in Fig. 18 (a). Fig. 18 illustrates the coalescence process predicted by A-FNN from two bubbles to one circle. And the adaptive variable is adjusted as 8.32643. Similarly, although taking the large time step, DeLISA still remains behaving well and is verified the effectiveness.

3.3. Experiments for three-dimensional PDE

3.3.1. Burgers equation

In order to illustrate our method can solve high dimensional problems, we consider the following three-dimensional Burgers equation [38]:

$$\begin{aligned} &u_t + uu_x + vu_y + wu_z = \varepsilon(u_{xx} + u_{yy} + u_{zz}), \\ &v_t + uv_x + vv_y + wv_z = \varepsilon(v_{xx} + v_{yy} + v_{zz}), \\ &w_t + uw_x + vw_y + ww_z = \varepsilon(w_{xx} + w_{yy} + w_{zz}),\ (x, y, z, t) \in [a, b]^3 \times [0, T], \end{aligned} \tag{41}$$

where $\varepsilon = 0.01$, $a = 0$, $b = 1$, $T = 0.1$, the exact solutions are given by:

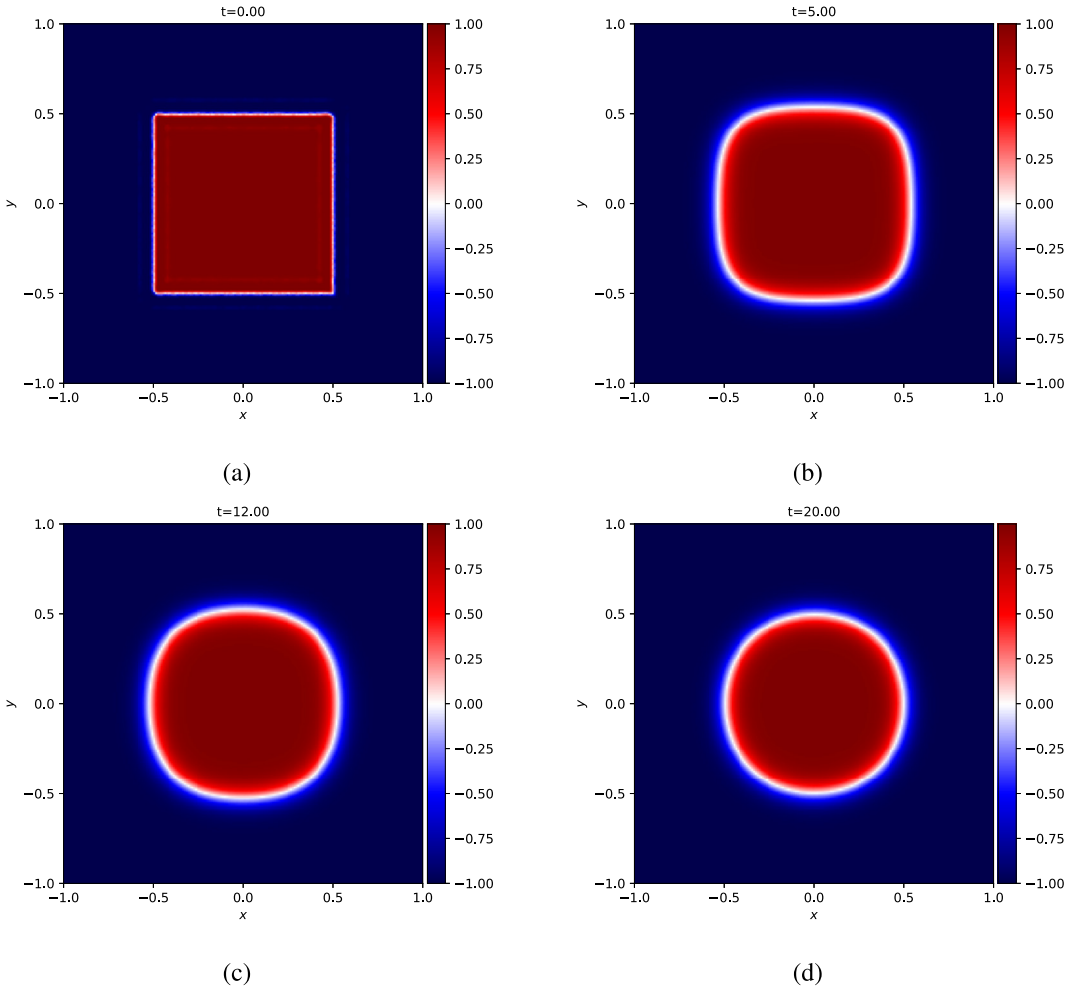


Fig. 17. Two-dimensional Allen-Cahn equation: the shrinking process of the phase variable predicted by the three temporal snapshots.

$$
\begin{aligned}
u(x,y,z,t) &= -2\varepsilon\left[\frac{1+e^{-t}cos(x)sin(y)sin(z)}{1+x+e^{-t}sin(x)sin(y)sin(z)}\right],\\
v(x,y,z,t) &= -2\varepsilon\left[\frac{1+e^{-t}sin(x)cos(y)sin(z)}{1+x+e^{-t}sin(x)sin(y)sin(z)}\right],\\
w(x,y,z,t) &= -2\varepsilon\left[\frac{1+e^{-t}sin(x)sin(y)cos(z)}{1+x+e^{-t}sin(x)sin(y)sin(z)}\right].
\end{aligned}
\tag{42}
$$

In this example, the A-FNN contains 2 hidden layers with 100 units per layer and *tanh* activation function. We determine the initial adaptive variable a as 15 in the first hidden layer, the stage $q=2$ and time step $\Delta t=0.001$. Moreover, the initial points $N_0=11\times 11$ and boundary points N_b at each time step are used during training. Fig. 19 shows the predictions and exact solutions in the xoy plane, $z=0.4$ at T. As can be seen from the figures, the predictions approximate exact solutions well. In order to determine the choice of adaptive variable, we examine the effect of initial adaptive variable with respect to three-dimensional Burgers equation, as is shown in Fig. 20. We can see that the error goes down as a goes up until $a=15$. Meanwhile, we also observe the error with a is lower than without a. Thus, the way we choose a is that increasing a until we get the smallest error. In Table 9, we set the layers as 2, neurons as 100 in each layer, $q=2$, and obtain the best result when $a=15$.

3.4. Comparison of PINN and DeLISA

To demonstrate the merit of DeLISA, the relative L^2 error comparison of PINN, DeLISA and DeLISA with adaptive variable is conducted in various PDEs, which is listed in Table 10, where DeLISA with adaptive variable is written as DeLISA(a). For

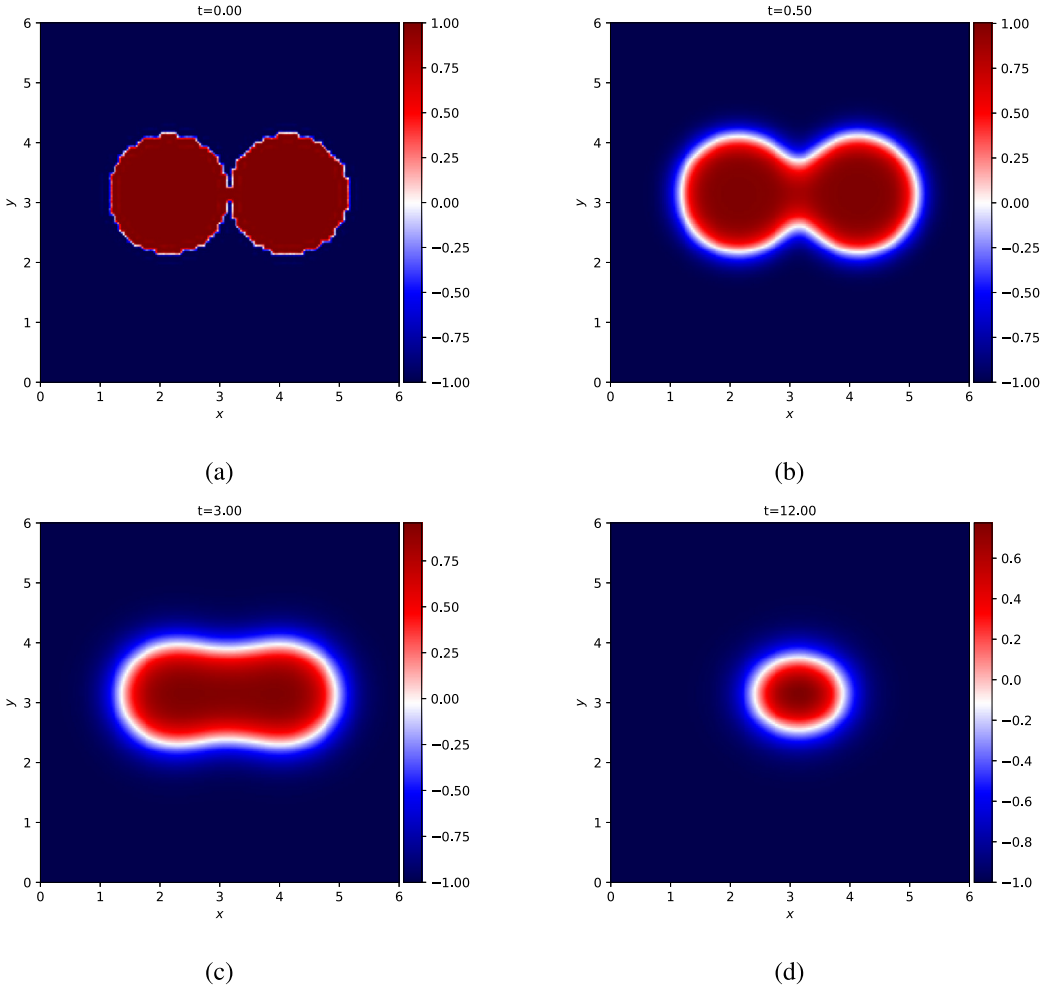


Fig. 18. Two-dimensional Allen-Cahn equation: the coalescence process of the phase variable predicted by the three temporal snapshots.

Table 9
Three-dimensional Burgers equation: the relative L^2 error between prediction and exact value for different initial adaptive variable a with time step $2\Delta t$.

t	$a=1$			$a=15$			$a=20$		
	L_u^2 error	L_v^2 error	L_w^2 error	L_u^2 error	L_v^2 error	L_w^2 error	L_u^2 error	L_v^2 error	L_w^2 error
0	1.08e-02	7.398e-03	8.83e-03	2.64e-03	1.56e-03	1.96e-03	2.75e-03	2.57e-03	1.97e-03
0.05	1.01e-02	6.81e-03	7.99e-03	3.43e-03	1.40e-03	2.23e-03	3.41e-03	2.15e-03	2.35e-03
0.08	9.98e-03	6.61e-03	7.57e-03	4.03e-03	1.52e-03	2.81e-03	4.05e-03	2.48e-03	3.02e-03
0.1	9.94e-03	6.57e-03	7.38e-03	4.44e-03	1.72e-03	3.26e-03	4.52e-03	2.92e-03	3.54e-03
All time	1.02e-02	6.87e-03	8.03e-03	3.50e-03	1.48e-03	2.39e-03	3.52e-03	2.37e-03	2.53e-03

the cases in Table 10, we show the relative L^2 error with respect to different methods in the same cases of the number of layers and neurons of network. Also, we remain the same convergence condition. The results suggest that DeLISA(a) performs better overall. For Burgers equation, Schrödinger equation and carburizing equation with variable coefficient, they are no significant difference on the relative L^2 errors calculated by different methods. To verify our method converges faster in above cases, we supplement the Fig. 21 and Table 11. Fig. 21 depicts PINN's prediction and DeLISA(a)'s prediction at a certain iteration, which presents DeLISA(a)'s prediction is closer to "sharp turns". In addition, Table 11 also shows the number of iterations of different methods and can be seen that DeLISA(a)'s iterations are less than PINN. To sum up, DeLISA(a) converges faster than PINN.

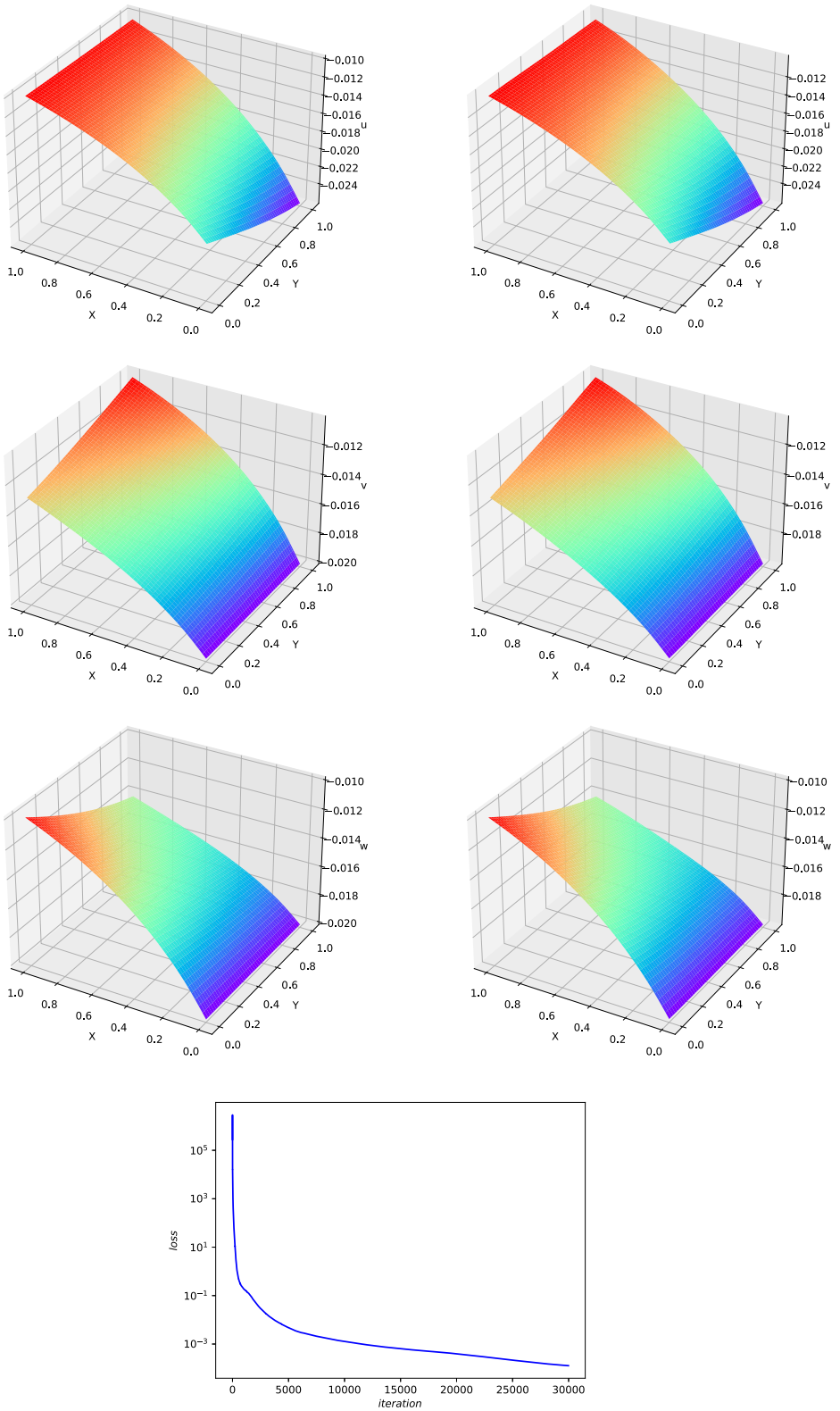
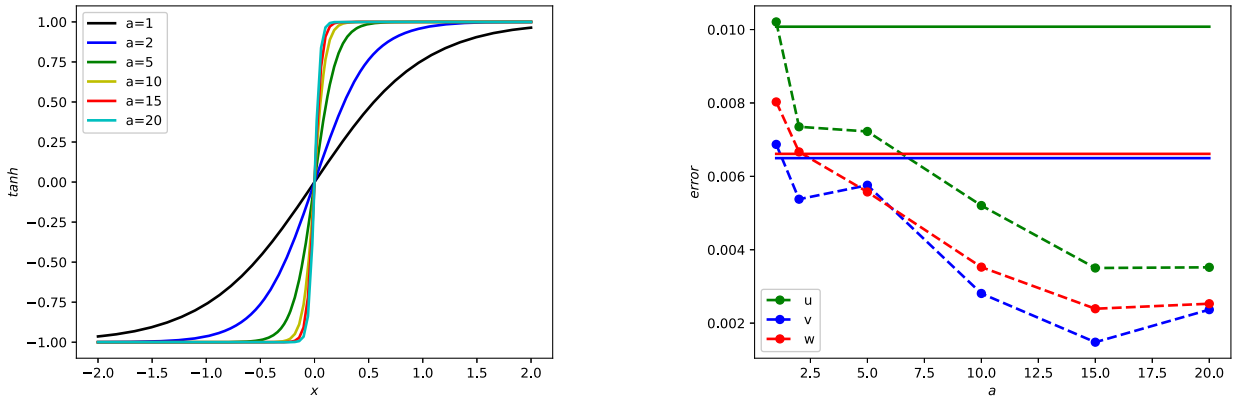


Fig. 19. Three-dimensional Burgers equation: The snapshots of predicted solutions (left column) and exact solutions (right column) at $t = T$ in the xoy plane, $z = 0.4$.

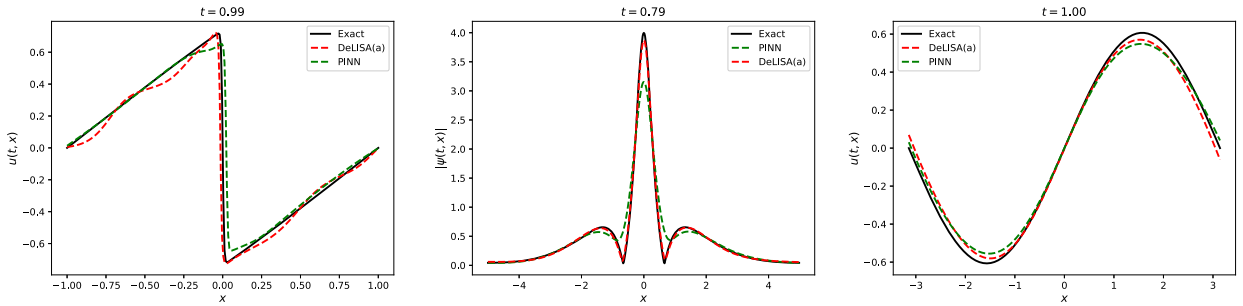


(a) (b)

Fig. 20. The left (a) shows the activations with different a. The right (b) is the comparison of error with different a, where the solid lines are the errors without a.

Table 10
The comparison of different methods with respect to relative L^2 error, where a is initial adaptive variable.

PDE	PINN	DeLISA	DeLISA(a)
Burgers	1.74e-03	4.15e-03	3.18e-03 (a=15)
Black-Scholes	1.48e+00	9.44e-04	9.03e-04 (a=0.5)
Allen-Cahn	9.49e-01	2.96e-02	7.25e-03 (a=2)
Schrödinger	2.41e-03	1.58e-03	1.43e-03 (a=1)
Carburizing_Const	3.82e+00	8.29e-03	8.28e-03 (a=2)
Carburizing_Var	3.28e-04	5.21e-04	2.70e-04 (a=3)
2D-Burgers	4.38e-04	2.43e-04	8.51e-05 (a=0.4)



(a) (b) (c)

Fig. 21. The comparison of PINN's prediction (green) and DeLISA(a)'s prediction (red): Burgers equation (a) at $iteration = 300$; Schrödinger equation (b) at $iteration = 1000$; Carburizing equation with variable (c) at $iteration = 50$.

Table 11
The number of iterations of different methods.

PDE	PINN	DeLISA	DeLISA(a)
Burgers	7307	7052	5895
Schrödinger	8491	7855	6928
Carburizing_Var	1360	1697	1058

4. Conclusions

In this paper, we propose a novel iteration scheme based on deep neural network framework for solving underlying solutions, namely DeLISA. In this method, by integrating the physical information of governing equation into time iteration schemes, our method avoids to sample a mass of interior points during training. And the application of time iteration schemes does not generate the mesh grids because of each sampled point only depending on itself at previous time step. The adaptive activation function is applied in the first hidden layer. And it improves convergence rate and accuracy. In addition, the L-BFGS-B method is adopted to minimize the loss function defined by the sum of squared errors.

For various numerical experiments, we test the performance with different time discretization methods, including implicit multistep method on one-dimensional problems, implicit Runge-Kutta method on two and three dimensional problems. The results demonstrate that DeLISA behaves well for a series of classical PDEs. Furthermore, the effects of hidden layer of neural network, time step, initial adaptive variable and stage of Runge-Kutta method are examined respectively, and the results present that the number of neurons, time steps and initial adaptive variable influence accuracy significantly. Especially for the changes of time step, the large time step can also obtain the smaller error. On the other hand, the stage of Runge-Kutta affects slightly for our proposed method. In addition, the F-Principle is illustrated via the changes of prediction with "sharp turns" at different iterations.

It is remarkable that there are some valuable issues remaining to further study. For example, we need large memory consumption due to A-FNN possessing parameters in quantity. Therefore, we could consider to replace it with convolutional neural network (CNN), which can significantly reduce the size of parameters. On the other hand, because the data points can be seen as a time sequence, we could consider other network architectures, such as recurrent neural network (RNN), long short term memory (LSTM) network. In our future work, we further consider the above tasks.

CRediT authorship contribution statement

Ying Li: Conceptualization, Formal analysis, Methodology, Supervision, Validation, Writing – review & editing. **Zuojia Zhou:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Shihui Ying:** Formal analysis, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors acknowledge support from NSFC 11971296, National Natural Science Foundation of China (No. 61873156).

References

[1] X. Feng, Fully discrete finite element approximations of the Navier-Stokes-Cahn-Hilliard diffuse interface model for two-phase fluid flows, SIAM J. Numer. Anal. 44 (3) (2006) 1049–1072.
[2] N. Touzi, Optimal stochastic control, stochastic target problems, and backward SDE, Fields Inst. Monogr. 29 (1) (2013).
[3] M. Huang, R.P. Malhame, P.E. Caines, Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle, Commun. Inf. Syst. 6 (3) (2006) 221–252.
[4] Z. Yang, A finite difference method for fractional partial differential equation, Appl. Math. Comput. 215 (2) (2009) 524–529.
[5] C.A. Taylor, T. Hughes, C.K. Zarins, Finite element modeling of blood flow in arteries, Comput. Methods Appl. Mech. Eng. 158 (1–2) (1998) 155–196.
[6] R. Eymard, G. Thierry, H. Raphaele, Finite volume methods, Handb. Numer. Anal. 7 (4) (2000) 713–1018.
[7] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.
[8] Y. LeCun, B. Yoshua, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[9] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proc. IEEE Comput. Conf. Comput. Vision Pattern Recognit., 2014.
[10] N. Wang, D.Y. Yeung, Learning a deep compact image representation for visual tracking, Adv. Neural Inf. Process. Syst. 26 (2013) 809–817.
[11] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[12] Y. Yang, M. Hou, H. Sun, T. Zhang, F. Weng, J. Luo, Neural network algorithm based on Legendre improved extreme learning machine for solving elliptic partial differential equations, Soft Comput. 24 (2) (2020) 1083–1096.
[13] Z. Cai, J. Chen, M. Liu, X. Liu, Deep least-squares methods: an unsupervised learning-based numerical method for solving elliptic PDEs, J. Comput. Phys. 420 (2020) 109707.
[14] D. Pfau, J.S. Spencer, A. Matthews, W. Foulkes, Ab initio solution of the many-electron Schrödinger equation with deep neural networks, Phys. Rev. Res. 2 (3) (2020) 033429.
[15] Y. Liao, P. Ming, Deep Nitsche method: deep Ritz method with essential boundary conditions, Commun. Comput. Phys. 29 (2021) 1365–1384.
[16] J. Huang, H. Wang, T. Zhou, An augmented Lagrangian deep learning method for variational problems with essential boundary conditions, arXiv preprint arXiv:2106.14348, 2021.
[17] Z.J. Xu, Y. Zhang, T. Luo, Y. Xiao, Z. Ma, Frequency principle: Fourier analysis sheds light on deep neural networks, arXiv preprint arXiv:1901.06523, 2019.
[18] Z. Liu, W. Cai, Z.J. Xu, Multi-scale deep neural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains, Commun. Comput. Phys. 28 (2020) 1970–2001.

[19] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[20] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[21] A. Al-Aradi, A. Correia, D. Naiff, G. Jardim, Y. Saporito, Solving nonlinear and high-dimensional partial differential equations via deep learning, arXiv preprint arXiv:1811.08782v1, 2018.
[22] M. Raissi, Deep hidden physics models: deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. 19 (1) (2018) 932–955.
[23] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A deep learning framework for solution and discovery in solid mechanics, arXiv preprint arXiv:2003.02751v2, 2020.
[24] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.
[25] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: a Navier-Stokes informed deep learning framework for assimilating flow visualization data, arXiv preprint arXiv:1808.04327, 2018.
[26] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136.
[27] H. Xu, H. Chang, D. Zhang, DL-PDE: deep-learning based data-driven discovery of partial differential equations from discrete and noisy data, arXiv preprint arXiv:1908.04463, 2019.
[28] H. Xu, H. Chang, D. Zhang, DLGA-PDE: discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm, J. Comput. Phys. (2020) 109584.
[29] J. Han, A. Jentzen, E. Weinan, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. 115 (34) (2018) 8505–8510.
[30] M. Raissi, Forward-backward stochastic neural networks: deep learning of high-dimensional partial differential equations, arXiv preprint arXiv:1804.07010, 2018.
[31] J. Cai, B. Dong, S. Osher, Z. Shen, Image restoration: total variation, wavelet frames, and beyond, J. Am. Math. Soc. 25 (4) (2012) 1033–1089.
[32] B. Dong, Q. Jiang, Z. Shen, Image restoration: wavelet frame shrinkage, nonlinear evolution pdes, and beyond, Multiscale Model. Simul. 15 (1) (2017) 606–660.
[33] Z. Long, Y. Lu, B. Dong, PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network, J. Comput. Phys. 399 (2019) 108925.
[34] H. Yao, Y. Gao, Y. Liu, FEA-Net: a physics-guided data-driven model for efficient mechanical response prediction, Comput. Methods Appl. Mech. Eng. 363 (2020) 112892.
[35] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (1) (2017) 5595–5637.
[36] A. Iserles, A First Course in the Numerical Analysis of Differential Equations, Cambridge University Press, 2009.
[37] Y. Li, F. Mei, Deep learning-based method coupled with small sample learning for solving partial differential equations, Multimed. Tools Appl. 5 (2020).
[38] X. Yang, Y. Ge, L. Zhang, A class of high-order compact difference schemes for solving the Burgers' equations, Appl. Math. Comput. 358 (2019).
[39] S.M. Allen, J.W. Cahn, A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, Acta Metall. 27 (6) (1979) 1085–1095.
[40] X. Wang, J. Kou, H. Gao, Linear energy stable and maximum principle preserving semi-implicit scheme for Allen-Cahn equation with double well potential, Commun. Nonlinear Sci. 98 (2) (2021) 105766.
[41] L. Bu, L. Mei, Y. Hou, Stable second-order schemes for the space-fractional Cahn-Hilliard and Allen-Cahn equations, Comput. Math. Appl. (2019).