Modern Physics Letters B
Vol. 36, No. 1 (2022) 2150531 (20 pages)
© World Scientific Publishing Company
DOI: 10.1142/S021798492150531X

Physics-informed neural networks method in high-dimensional integrable systems

Zheng Wu Miao* and Yong Chen*,†,‡

*School of Mathematical Sciences,
Shanghai Key Laboratory of Pure Mathematics and Mathematical Practice,
East China Normal University, Shanghai 200241, China
†College of Mathematics and Systems Science,
Shandong University of Science and Technology, Qingdao 266590, China
‡ychen@sei.ecnu.edu.cn

Received 15 July 2021
Revised 29 August 2021
Accepted 21 September 2021
Published 13 December 2021

In this paper, the physics-informed neural networks (PINNs) are applied to high-dimensional system to solve the $(N+1)$-dimensional initial-boundary value problem with $2N+1$ hyperplane boundaries. This method is used to solve the most classic $(2+1)$-dimensional integrable Kadomtsev–Petviashvili (KP) equation and $(3+1)$-dimensional reduced KP equation. The dynamics of $(2+1)$-dimensional local waves such as solitons, breathers, lump and resonance rogue are reproduced. Numerical results display that the magnitude of the error is much smaller than the wave height itself, so it is considered that the classical solutions in these integrable systems are well obtained based on the data-driven mechanism.

Keywords: PINN; high-dimensional integrable systems; KP equation; resonance rogue.

1. Introduction

Nonlinear problems play an important role in mathematical physics and engineering fields such as fluid mechanics, plasma physics and optical fiber communication.[1–4] The nonlinear phenomenon in the natural world is abstracted as simple mathematical and physical models, which are expressed as nonlinear partial differential equations (NPDEs). In recent decades, the precise and numerical solutions of NPDEs have been the focus of researchers.[5–9]

With the explosive growth of data resources, deep learning technologies with the advantage of *big data* have been widely used in various fields including image

‡Corresponding author.

recognition,[10] cognitive science[11] and natural language processing,[12] and made revolutionary progress. Convolutional neural networks (CNNs), recurrent neural networks (RNNs) and other network structures that have been continuously developed are more inclined to solve specific learning tasks.[10,13] However, the high cost of data acquisition makes traditional deep learning algorithms face huge challenges in complex physics and engineering analysis. Recently, the deep learning framework based on physical constraints [physics-informed neural networks (PINNs)] proposed by Raissi *et al.* provides a new idea for the learning of *small data* regime.[14–16] The PINNs, as a supervised deep learning method, can solve the forward and inverse problems of partial differential equations (PDEs) with a small sample of data, and exhibit outstanding generalization capabilities. This deep learning framework suddenly became the focus, and a number of new developments were reported based on the framework mentioned above.[17–23] For example, Yang *et al.* proposed a Bayesian PINN (B-PINN) to solve both forward and inverse nonlinear problems described by PDEs and noisy data.[18] The improved PINN method presented by Karniadakis *et al.* employs adaptive activation functions to accelerate the minimization process of the loss values.[20] Raissi's own team developed the idea of PINN and solved the problems related to fluid visualization of Navier–Stokes equations.[21] The PINN method has even been applied to fractional PDEs,[22] stochastic PDEs,[23] and so on.

Integrable systems are a class of nonlinear systems with excellent properties. The abundant exact solutions in integrable systems provide a huge sample space for the PINN method. In the integrable deep learning community, Chen's team achieved a series of results.[24–28] For example, Pu *et al.* used an improved PINN method to recover the soliton, breathers and rogue of the nonlinear Schrödinger (NLS) equation.[26] Based on the PINN method, Peng *et al.* obtained the data-driven periodic rogue wave and other local wave solutions of the Chen–Lee–Liu (CLL) equation.[27] Lin and Chen fully considered the advantages of the integrable systems and proposed a two-stage PINN method based on conserved quantities to solve Boussinesq–Burgers (BB) equation and Sawada–Kotera (SK) equation.[28] In addition, Wang and Yan discussed the forward and inverse problems of the defocusing NLS equation based on the PINN method.[29] Dai *et al.* used PINN method to solve a variety of femtosecond optical soliton solutions of high-order NLS equation.[30]

Compared with the traditional method, the PINN method is robust under the *small data* regime, which provides a possibility for advancing the solution of high-dimensional problems, and related researches have made attempts in high-dimensional systems.[20,22] But for general nonlinear systems, this approach always sacrifices space cost and time cost to a large extent. The challenge of the inevitable curse of dimensionality is even worse when it comes to solving tasks that are inherently unstable. For example, the trajectory of a chaotic system has the property of exponential separation, so as the error is transmitted in the network, the predicted result will deviate from the true value. This means that chaotic systems have an extremely low tolerance for errors in the PINN. However, the rich local wave solutions in the integrable systems make it reasonable for us to discuss a small training area. Combined with the advantage of the high tolerance of the integrable systems to errors (the trajectories will not be separated exponentially), the curse of dimensionality has the possibility of being overcome. Therefore, the integrable systems may be the most suitable place for the PINN method to work, especially in high-dimensional situations.

In our work, the framework of the PINN method to solve the $(N+1)$-dimensional initial-boundary value problem with $2N+1$ hyperplane boundaries is given, and many feasible model adjustments are considered. And this framework is applied to solve the most classic high-dimensional integrable equations such as: $(2+1)$-dimensional Kadomtsev–Petviashvili (KP) equation, $(3+1)$-dimensional reduced KP equation. More specifically, in Sec. 2, the framework of the PINN method under the high-dimensional system is given; in Sec. 3, the dynamic behaviors of the single soliton, two solitons, breathers and lump of the $(2+1)$-dimensional KP equation are reproduced based on the PINN method; in Sec. 4, the PINN method is applied to solve the interaction solution (resonance rogue) of the $(3+1)$-dimensional reduced KP equation. Finally, conclusion and expectation are given in the last section.

2. The PINN Method in High-Dimensional Space

The general form of a $(N+1)$-dimensional nonlinear evolution equation in real space is expressed as

$$u_t + \mathcal{N}[u] = 0, \quad \mathbf{x} \in \Omega, \ \ t \in [-T, T], \tag{1}$$

where $u = u(\mathbf{x}, t)$ is the real-valued solution of the equation, Ω is a subset of $\mathbb{R}^N$, therefore $\mathbf{x}$ is actually an N-dimensional vector recorded as $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, and $\mathcal{N}[\cdot]$ is a nonlinear differential operator with respect to $\mathbf{x}$. Thus, $\mathcal{N}[u]$ is a nonlinear functional and written as $\mathcal{N}_u(u, u_{x_1}, \ldots, u_{x_N}, u_{x_1x_1}, \ldots, u_{x_Nx_N}, \ldots)$, which generally contains high-order dispersion terms and nonlinear terms. In this section, we will propose a framework for the application of the PINN method in high-dimensional space.

2.1. *Neural network structure and initialization*

Consider building a feedforward neural network (NN) with a depth of D, which includes an input layer (0th layers), an output layer (Dth layers) and $D-1$ hidden layers. The input layer contains $N+1$ nodes, and the output layer has 1 node, which depends on the dimensions of the domain and the range in Eq. (1). Without loss of generality, assume that the number of nodes in the dth layer network is N_d. Regarding the number of network parameters, the increase in dimensionality only affects the input layer, so even considering the possibility of curse of dimensionality, the fully connected network is still reliable. The D-layer feedforward NN with fully connected structure is shown in Fig. 1.

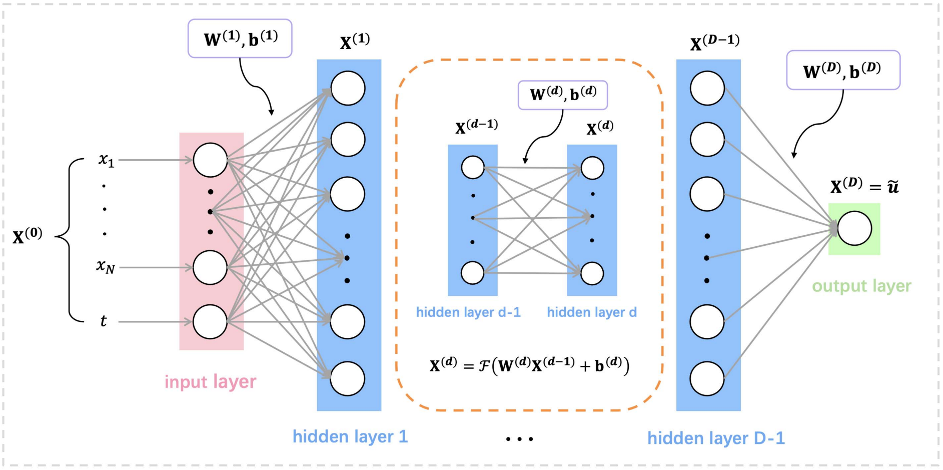

$\mathbf{W^{(1)}}, \mathbf{b^{(1)}}$
$\mathbf{X^{(1)}}$
$\mathbf{W^{(d)}}, \mathbf{b^{(d)}}$
$\mathbf{X^{(d-1)}}$
$\mathbf{X^{(d)}}$
$\mathbf{X^{(D-1)}}$
$\mathbf{W^{(D)}}, \mathbf{b^{(D)}}$
$\mathbf{X^{(D)}} = \tilde{u}$
$\mathbf{X^{(0)}}$
x_1
x_N
t
input layer
hidden layer 1
hidden layer d-1
hidden layer d
$\mathbf{X^{(d)}} = \mathcal{F}(\mathbf{W^{(d)}}\mathbf{X^{(d-1)}} + \mathbf{b^{(d)}})$
hidden layer D-1
output layer

Fig. 1. (Color online) The D-layer feedforward NN with fully connected structure. The network consists of an input layer, an output layer and $D-1$ hidden layers. In order to facilitate the presentation, the number of hidden layers is omitted and replaced by a nonlinear transformation $\mathbf{X}^{(d)} = \mathcal{F}(\mathbf{W}^{(d)}\mathbf{X}^{(d-1)} + \mathbf{b}^{(d)})$ between the hidden layers.

The ith network receives the N_i-dimensional vector input of the previous network, and outputs the N_{i+1}-dimensional vector to the next network, which is clear in the network structure diagram, see Fig. 1. Of course, for problems with high complexity, it is a wise choice to replace a fully connected network with a locality connected network. In order to connect the vectors of different dimensions between the layers of the network, it is necessary to introduce a nonlinear transformation, written as

$$\mathbf{X}^{(d)} = \mathcal{T}_d(\mathbf{X}^{(d-1)}) = \mathcal{F}(\mathbf{W}^{(d)}\mathbf{X}^{(d-1)} + \mathbf{b}^{(d)}), \tag{2}$$

where $\mathbf{W}^{(d)} \in \mathbb{R}^{N_d \times N_{d-1}}$ and $\mathbf{b}^{(d)} \in \mathbb{R}^{N_d}$ represent the weight matrix and bias vector of the dth layer network, respectively. And $\mathbf{X}^{(d)}$ is the state vector of the dth layer node, especially $\mathbf{X}^{(0)} = \mathbf{x} \oplus t = (x_1, x_2, \ldots, x_N, t)$, and $\mathcal{F}$ is a nonlinear activation function, which is usually selected as $Sigmoid$ function, $Tanh$ function or $Relu$ function, etc. However, $\mathcal{T}_d$ is a nonlinear transformation composed of a nonlinear activation function $\mathcal{F}$ and an affine transformation, and the formula (2) tells us that our model fixes the activation function of each layer. The weights and biases of the network are stored in tensor-type variables in tensorflow,[31] so they need to be initialized. For bias terms, they are initialized to tensors with an initial value of zero, but the initialization of weight terms satisfies the conditions:

$$\mathbf{W}^{(d)} \sim N(0, \sigma_d^2), \quad \sigma_d^2 = \frac{2}{N_d + N_{d+1}}, \quad d = 1, 2, \ldots, D, \tag{3}$$

which is the $Xavier$ initialization method[32] that can maintain state variance and gradient variance. And in fact, (3) is a truncated normal distribution, which means that the initial value outside of two standard deviations will be discarded. In addition, methods such as *Kaiming* initialization[33] and *Fixup* initialization[34] may also be used.

2.2. *Initial-boundary value condition*

It is well known that the solution of universal equations (PDEs without definite solution conditions) is uncertain, but for the current deep learning technology, only one fixed solution without parameters can be learned. So, the key is to establish a problem with initial-boundary value. Consider the *Dirichlet boundary* conditions of Eq. (1):

$$\begin{cases} u(\mathbf{x},-T) = u_0(\mathbf{x},-T), & \forall\, \mathbf{x} \in \Omega, \\ u(\mathbf{x},t) = u_0(\mathbf{x},t), & \forall\, \mathbf{x} \in \partial\Omega,\ t \in [-T,T], \end{cases} \tag{4}$$

where the solutions $u_0(\mathbf{x},t)$ of Eq. (1) are our learning goal, and the two formulas in (4) represent the initial value condition and the boundary condition, respectively. Of course, other boundary conditions such as *Neumann Boundary* and *Periodic Boundary* are also feasible at the theoretical level, and even have better performance.

In order to facilitate the acquisition of initial value and boundary value data for the NN to learn, only a simple domain shape is discussed in this paper. Consider the domain Ω is an N-dimensional closed rectangular as

$$\Omega = [x_1^l, x_1^u] \times [x_2^l, x_2^u] \times \cdots \times [x_N^l, x_N^u], \tag{5}$$

where x_i^u and x_i^l are defined as the upper and lower bounds of Ω in the dimension of x_i. Therefore, the boundary of the domain Ω is a regular $(N-1)$-dimensional closed rectangle, combined with the initial value conditions in (4), then the data points needed for deep learning will come from the following $2N+1$ hyperplanes:

$$H_i^l = \{\mathbf{X} \in \mathcal{O} \mid x_i = x_i^l\}, \quad H_i^u = \{\mathbf{X} \in \mathcal{O} \mid x_i = x_i^u\}, \quad 1 \le i \le N, \tag{6}$$

$$H_0 = \{\mathbf{X} \in \mathcal{O} \mid t = -T\}, \quad \mathcal{H} = H_0 \cup \bigcup_{i=1}^{N} (H_i^l \cup H_i^u), \tag{7}$$

where $\mathcal{O} = \Omega \times [-T,T]$, $\mathbf{X} = (x_1, x_2, \ldots, x_N, t)$, and $\mathcal{H}$ represents the union of all $2N+1$ hyperplanes. Previous work has tried to use the PINN method for $(1+1)$-dimensional equations, so only two boundary lines and one initial value line need to be considered. However, the work of this paper is to apply the PINN method to the $(N+1)$-dimensional equation, which is the first time it is proposed in the field of integrable deep learning. This means that the amount of data for the initial-boundary value will increase exponentially. For example, for a $(2+1)$-dimensional system, there will be four boundary surfaces and one initial value surface. Figure 2 displays the initial value surface and boundary surfaces in the case of $N = 2$.

Deep learning requires discrete data points to support the network optimization process, as a result, the data points of the continuous equation (1) must be

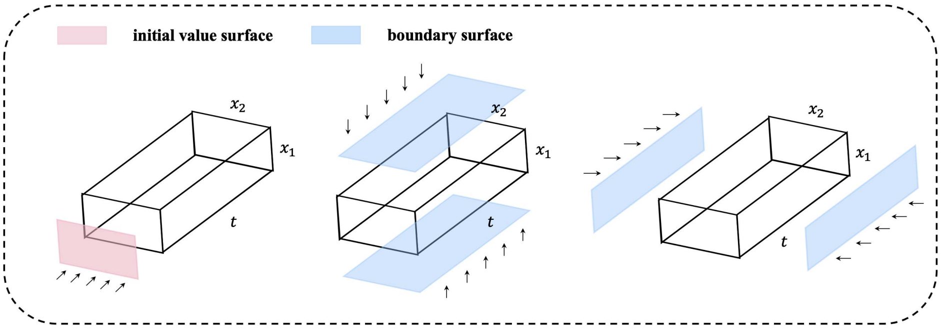


Fig. 2. (Color online) Initial value and boundary surfaces in $(2+1)$ dimension.

discretized. First, divide the ith dimension of the spatial domain Ω into n_i equal parts, and then divide the time domain $[-T, T]$ into n_t equal parts to obtain an $(N+1)$-dimensional grid rectangle. We define these $(n_1+1)\cdots(n_N+1)(n_t+1)$ points as the 2-*type* points, and their intersection with the set $\mathcal{H}$ is additionally specified as the 1-*type* points. In addition, the *Latin hypercube sampling* (LHS) method[35] is used to randomly select n_f points in $\mathcal{O}$, which are defined as the 3-*type* points or internal collocation points. The training of the NN and the calculation of the generalization error of the model are inseparable from these three types of points, and the specific details are discussed in the following section.

2.3. *Loss function, model training and generalization*

The loss function is the key to network optimization. However, the loss function of PINN method is composed of initial-boundary condition constraints and physical equation constraints, which correspond to the 1-*type* points and the 3-*type* points, respectively. The 1-*type* points are defined as $\{\mathbf{X}_b^i, u_b^i\}_{i=1}^{n_b}$, and the 3-*type* points are defined as $\{\mathbf{X}_f^i\}_{i=1}^{n_f}$, where $\mathbf{X}_b^i = \mathbf{x}_b^i \oplus t_b^i = (x_{b,1}^i, \ldots, x_{b,N}^i, t_b^i)$, $\mathbf{X}_f^i = \mathbf{x}_f^i \oplus t_f^i = (x_{f,1}^i, \ldots, x_{f,N}^i, t_f^i)$ and $u_b^i = u_0(\mathbf{x}_b^i, t_b^i)$. Next, define a function $f(\tilde{\mathbf{x}}, \tilde{t}, u)$ derived from the left side of Eq. (1), i.e.

$$f(\tilde{\mathbf{x}}, \tilde{t}; u) = \mathcal{N}_0[u]|_{\mathbf{x}=\tilde{\mathbf{x}}, t=\tilde{t}}, \quad \mathcal{N}_0[\cdot] = \mathcal{N}[\cdot] + \partial_t[\cdot], \tag{8}$$

where $u = u(\mathbf{x}, t)$ is regarded as a function parameter, especially if u_0 is the solution of Eq. (1), then obviously there is $f(\mathbf{x}, t; u_0) = 0, \forall \mathbf{x} \in \Omega, t \in [-T, T]$. However, the main idea of PINN is to find suitable parameters $\mathbf{W}^{(d)}$, $\mathbf{b}^{(d)}, d = 1, 2, \ldots, D$ from parameter space Θ to make the composite function $\tilde{u}(\mathbf{x}, t, \theta)$ close enough to the solution u_0 of Eq. (1). And $\tilde{u}(\mathbf{x}, t, \theta)$ is actually a function represented by the NN, specifically written as

$$\tilde{u}(\mathbf{x}, t, \theta) = (\mathcal{T}_{D-1} \circ \mathcal{T}_{D-2} \circ \cdots \circ \mathcal{T}_1)(\mathbf{X}), \tag{9}$$

where $\mathbf{X} = \mathbf{x} \oplus t$, $\theta = \{\mathbf{W}^{(d)}, \mathbf{b}^{(d)}\}_{d=1}^{D} \in \Theta$ and "$\circ$" represents the functional composition operator. Each group of parameters θ in the parameter space Θ determines a function $\tilde{u}(\mathbf{x}, t, \theta)$, which is obvious in (9). In order to better measure the gap between $u_0(\mathbf{x}, t)$ and $\tilde{u}(\mathbf{x}, t, \theta)$, a loss function composed of initial-boundary condition constraints and physical equation constraints will be constructed as

$$\text{Loss}(\theta) = \text{Loss}_b(\theta) + \text{Loss}_f(\theta), \tag{10}$$

where

$$\begin{aligned} \text{Loss}_b(\theta) &= \frac{1}{n_b}\sum_{i=1}^{n_b} |\tilde{u}(\mathbf{x}_b^i, t_b^i, \theta) - u_b^i|^2, \\ \text{Loss}_f(\theta) &= \frac{1}{n_f}\sum_{i=1}^{n_f} |f(\mathbf{x}_f^i, t_f^i; \tilde{u}(\mathbf{x}, t, \theta))|^2. \end{aligned} \tag{11}$$

Here, Loss_b represents the initial-boundary condition constraint, which measures the data fitting condition at the 1-*type* points, and Loss_f is the physical equation constraint, which measures whether the equation is satisfied at the 3-*type* points. In addition, the partial derivatives involved in the model can be easily solved in tensorflow with the help of automatic differentiation (AD) technology.[36] Of course, because of the high-cost nature of the data, the training of initial-boundary points usually involves only a part of the 1-*type* points. Obviously, the smaller the loss function is, the closer $\tilde{u}(\mathbf{x}, t, \theta)$ is to $u_0(\mathbf{x}, t)$ (it just means that some of their properties are similar at specific points). The goal of model training is to find the optimal parameter $\theta^* = \{\mathbf{W}^{(d),*}, \mathbf{b}^{(d),*}\}_{d=1}^{D}$, so that the loss function can reach the global minimum (of course, it must be small enough). Here, we utilize the L-BFGS-B algorithm[37] to optimize loss functions, and other gradient descent algorithms such as *stochastic gradient descent* (SGD)[38] and *batch gradient descent* (BGD)[39] can also be considered. Regrettably, the global optimal solution may not always be found, and some local optimal solutions are replaced. These local optimal solutions can still be accepted by us if they have good performance in model generalization.

The 2-*type* points mentioned above play an important role in measuring the generalization performance of the model. They are defined as $\{\mathbf{X}_g^i, u_g^i\}_{i=1}^{n_g}$, where $\mathbf{X}_g^i = \mathbf{x}_g^i \oplus t_g^i = (x_{g,1}^i, \dots, x_{g,N}^i, t_g^i)$ and $u_g^i = u_0(\mathbf{x}_g^i, t_g^i)$. The training of the model does not involve the 2-*type* points, which is obvious from the expressions (10) and (11) of the loss function. Therefore, it is reasonable to define the generalization error of the model at the 2-*type* points as

$$\text{Error} = \frac{\sqrt{\sum_{i=1}^{n_g} |\tilde{u}(\mathbf{x}_g^i, t_g^i, \theta^*) - u_g^i|^2}}{\sqrt{\sum_{i=1}^{n_g} |u_g^i|^2}}, \tag{12}$$

where $\tilde{u}(\mathbf{x}_g^i, t_g^i, \theta^*)$ represents the result of the generalization of the optimal solution of the model at point $(\mathbf{x}_g^i, t_g^i)$, and (12) is actually an error calculation based on the $\mathbb{L}_2$ norm, which mainly measures the average level of error. Other norms can construct different error formulas, such as the error formula based on the $\mathbb{L}_\infty$ norm to measure the maximum error of the model. The generalization error shows the

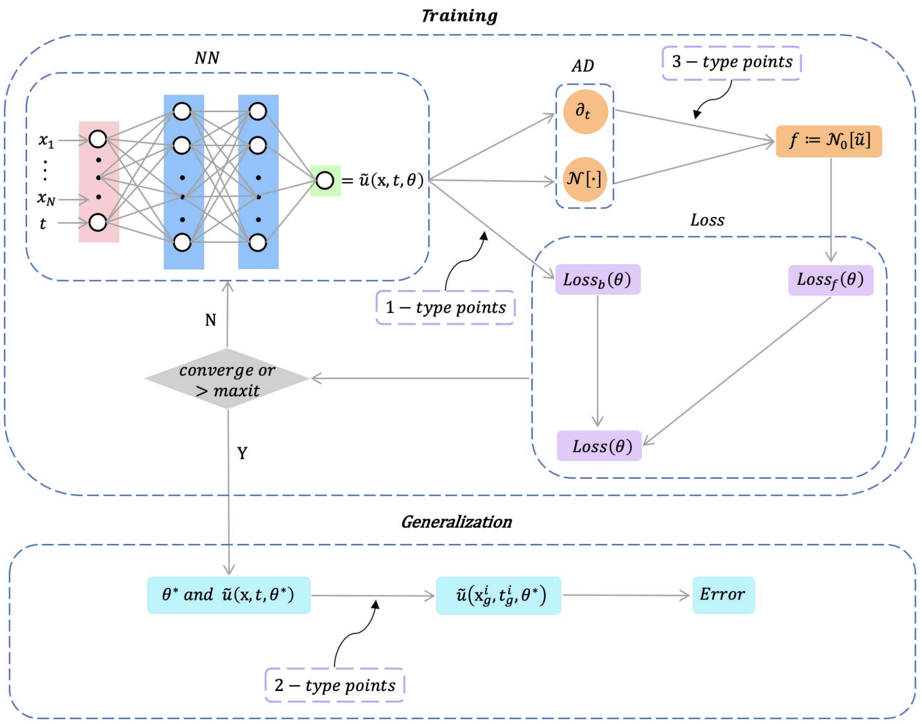


Fig. 3. (Color online) Process sketch of the PINN method in a high-dimensional system. The framework is divided into two parts: training and generalization. First, use the $\tilde{u}(\mathbf{x}, t, \theta)$ constructed by the NN and the AD to calculate the $\text{Loss}_b(\theta)$ and $\text{Loss}_f(\theta)$, and then the model is continuously trained through the convergence judgment. The final model enters the generalization module, and the generalization error is estimated at the 2-*type* points.

generalization ability of the model. However, deep learning is to balance model training errors and model generalization errors to avoid overfitting. Therefore, it is a good choice to add parameter regularization items to the loss function, such as L_1 or L_2 regularization.[40]

Figure 3 is a process sketch of the PINN method in a high-dimensional system. Without loss of generality, this paper assumes that the number of nodes in the hidden layer is equal ($N_d = N_0, d = 1, 2, \ldots, D$), and selects the hyperbolic function tanh as the activation function (if there are no special instructions). All codes in this paper are based on Python 3.7 and tensorflow 1.15, and all numerical examples reported here are run on a DELL Precision 7920 Tower computer with 2.10 GHz 8-core Xeon Silver 4110 processor and 64 GB memory.

3. The Application of PINN Method to (2 + 1)-Dimensional KP Equation

In this section, an example of $N = 2$, that is, the (2 + 1)-dimensional KP equation has become the object of our discussion. The non-dimensional form of the

$(2+1)$-dimensional KP equation[41] is written as

$$(u_t + u_{xxx} + 6uu_x)_x + \sigma^2 u_{yy} = 0, \tag{13}$$

where σ^2 is the equation parameter and $u = u(x, y, t)$. In particular, (13) is called the KPI when $\sigma^2 = -1$, and is called the KPII when $\sigma^2 = 1$, and the KP equation can be reduced to the KdV equation when u is independent of y. Equation (13) was proposed to study the evolution of small amplitude long ion acoustic waves propagating in plasma under long lateral disturbances,[41] and was widely used in almost all fields of physics, such as shallow water waves[42] and nonlinear optics.[43] The fully integrable KP equation is a classic high-dimensional model that is often used to test new mathematical techniques.

Consider an initial-boundary value problem with Dirichlet condition for the $(2+1)$-dimensional KP equation:

$$\begin{cases} (u_t + u_{xxx} + 6uu_x)_x + \sigma^2 u_{yy} = 0, \\ u(x, y, -T) = u_0(x, y, -T), \quad \forall (x, y) \in \Omega, \\ u(x, y, t) = u_0(x, y, t), \quad \forall (x, y) \in \partial\Omega,\ t \in [-T, T], \end{cases} \tag{14}$$

where $u_0(x, y, t)$ is usually taken as a known solution of the KP equation and $\Omega = [x^l, x^u] \times [y^l, y^u]$. In addition, there are two special explanations. First, because of the scale transformation, the domain $[-T, T]$ of t does not lose its generality. Second, the equation in (14) is not the standard form in (1), but this does not affect our results.

The well-known transformation

$$u = 2(\ln f)_{xx}, \tag{15}$$

converts the KP equation (13) into bilinear form

$$\begin{aligned}(D_t D_x + D_x^4 + \sigma^2 D_y^2) f \cdot f = f_{tx} f - f_t f_x + f f_{xxxx} - 4 f_x f_{xxx} \\ + 3 f_{xx}^2 + \sigma^2 (f f_{yy} - f_y^2) = 0, \end{aligned} \tag{16}$$

where $D_t^n D_x^m f \cdot g$ is the Hirota bilinear operator.[8] It is easy to obtain the multiple solitons solution[44] of the KP equation from the bilinear equation (16) as follows:

$$u_n = 2\left(\frac{f_{n,xx}}{f_n} - \frac{f_{n,x}^2}{f_n^2}\right), \quad f_n = \sum_{\mu=0,1} \exp\left(\sum_{i<j}^{n} \mu_i \mu_j A_{ij} + \sum_{i=1}^{n} \mu_i \eta_i\right), \tag{17}$$

where

$$\eta_i = k_i[x + p_i y + (-k_i^2 - \sigma^2 p_i^2)t] + \xi_i^{(0)}, \quad e^{A_{ij}} = \frac{3(k_i - k_j)^2 - \sigma^2 (p_i - p_j)^2}{3(k_i + k_j)^2 - \sigma^2 (p_i - p_i)^2}, \tag{18}$$

with the wave numbers k_i, p_i, original positions $\xi_i^{(0)}$. Here, the summation symbol $\sum_{\mu=0,1}$ is all possible combinations of $u_i = 0,\ 1 (j = 1, 2, \ldots)$, so it means the addition of 2^n terms.

Within the acceptable error, the single soliton, two solitons and breathers of the $(2+1)$-dimensional KP equation based on (17) with (18) have been well numerically explained through the PINN method in Secs. 3.1–3.3. However, the dynamic behavior of lump in NNs is discussed in Sec. 3.4.

3.1. *The data-driven single soliton solution*

First, when the parameters of the soliton solution (17) are set as: $n = 1, \sigma^2 = 3, k_1 = 3, p_1 = 2, \xi_1^{(0)} = 0$, the single soliton solution of the KP equation (13) with the parameter $\sigma^2 = 3$ is written as

$$u_1(x,y,t) = \frac{18e^{3x+6y-63t}}{(1+e^{3x+6y-63t})^2}, \tag{19}$$

which is an exact solution. Let $[-T,T] = [-0.2,0.2], \Omega = [-2,2]\times[-2,2]$, the initial-boundary value problem will be determined when $u_0(x,y,t) = u_1(x,y,t)$ is substituted into (14). A 11-layer feedforward NN with 40 nodes in each hidden layer is constructed to find a data-driven solution to the above initial-boundary value problem. The three dimensions $x \in [-2,2]$, $y \in [-2,2]$, $t \in [-0.2,0.2]$ of the spatio-temporal data set are divided into $n_1 = n_2 = 65$ and $n_t = 33$ discrete points to obtain the 1-*type* and 2-*type* of points mentioned in Sec. 2.2. The actual data training is carried out by randomly selecting $n_b = 6000$ points from the 1-*type* points and combining with the $n_f = 50000$ 3-*type* points selected by the LHS method. The PINN reduces the loss to 6.5699×10^{-5} after 2310 iterations in 1482.33 s, and has a good generalization effect (Error $= 5.7423\times10^{-4}$).

Figure 4(a) displays the time snapshot of the three-dimensional profile and two-dimensional cross-sectional view of the data-driven single soliton solution of the KP equation, which is based on the PINN method. The linear soliton evolves in the positive direction along the x- and y-axes, which is obvious in the figure. In addition, the consistency of the predicted solution and the exact solution on the cross-sectional exhibit that the model has achieved good results in the learning of single soliton solution. More specifically, the order of magnitude displayed by the color bars of the error density graph indicates that the difference between the predicted solution and the exact solution at the three moments is quite small. Figure 4(b) shows that the generalization error of the model mainly comes from the soliton rather than the background wave, which means that the learning of the NN at large gradients needs to be improved. In general, the NN has successfully learned the dynamic behavior of the single soliton solution, which can be derived from the above evidence and referring to the average generalization error (Error $=$ 5.7423×10^{-4}).

3.2. *The data-driven two-soliton solution*

Naturally, after taking the free parameter as $n = 2, \sigma^2 = 3, k_1 = -2, k_2 = 3, p_1 = \frac{2}{5}, p_2 = -\frac{2}{5}, \xi_1^{(0)} = -3, \xi_2^{(0)} = 0$ in (17), the two-soliton solution of the KP equation

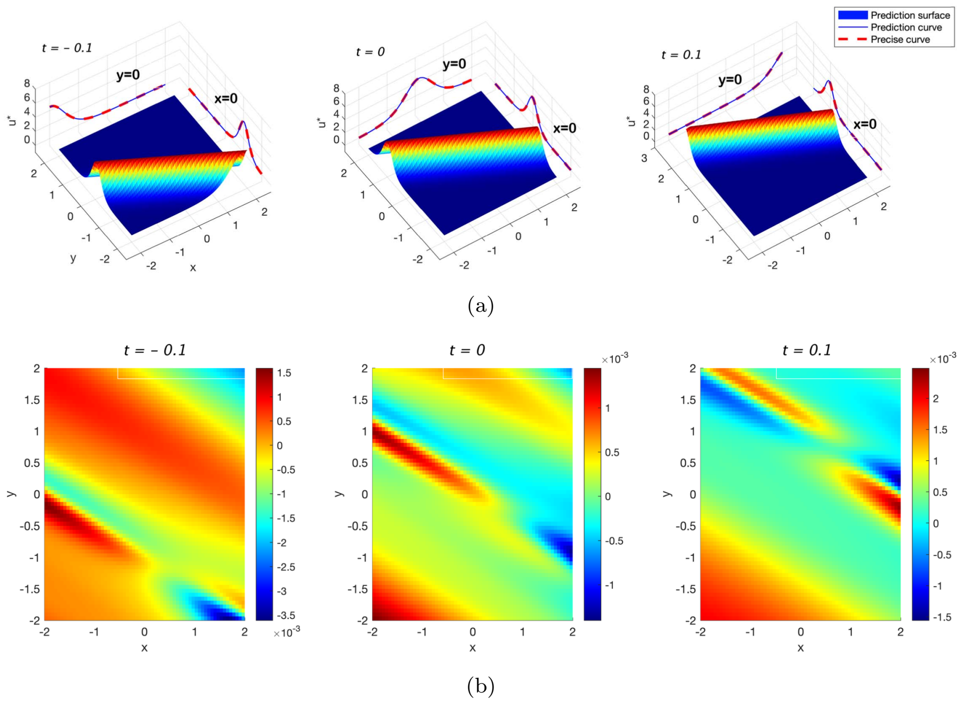


Fig. 4. (Color online) The data-driven single soliton solution of the KP equation: (a) Three time snapshots of the three-dimensional diagram and the corresponding two-dimensional cross-sectional diagrams at $x = 0$ and $y = 0$ (the blue solid line and the red dashed line correspond to the predicted solution and the exact solution, respectively); (b) density map of error (predicted solution minus exact solution).

with the parameter $\sigma^2 = 3$ is expressed as

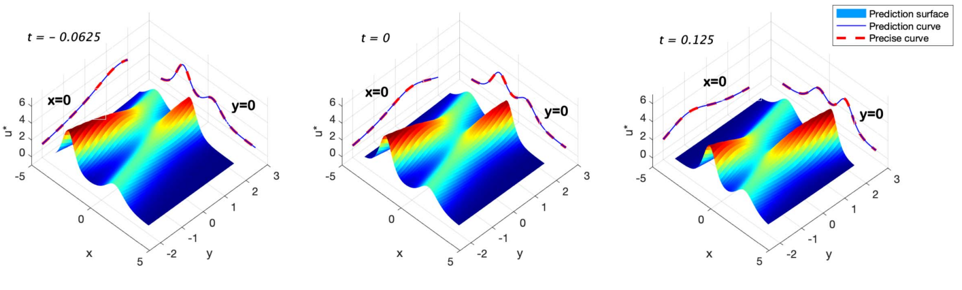
$$u_2(x,y,t) = \frac{2(f_{2,xx}f_2 - f_{2,x}^2)}{f_2^2}, \tag{20}$$

with

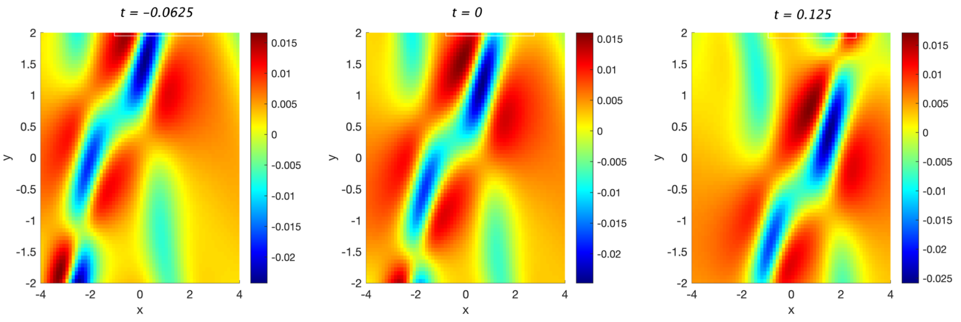
$$f_2 = 1 + e^{-2x-\frac{4}{5}y+\frac{224}{25}t-3} + e^{3x-\frac{6}{5}y-\frac{711}{25}t} + \frac{203}{3}e^{x-2y-\frac{487}{25}t-3}. \tag{21}$$

A 12-layer feedforward NN with 40 nodes per hidden layer will solve the initial-boundary value problem (14) with $[-T,T] = [-0.5, 0.5], \Omega = [-4,4]\times[-2,2]$ and $u_0(x,y,t) = u_2(x,y,t)$. The discrete training data are similar to Sec. 3.1, and the space-time dimensions are also discretely divided into $n_1 = n_2 = 65, n_t = 33$, and the corresponding number of randomly selected points of 1-*type* and 3-*type* is $n_b = 6000, n_f = 50000$. The PINN has also achieved great success in the two-soliton learning of the KP equation.

The three-dimensional contour diagram, two-dimensional cross-section diagram and the corresponding error density diagram of the data-driven solution of the two solitons are clearly shown in Fig. 5. The collision of the two linear solitons



(a)



(b)

Fig. 5. (Color online) The data-driven two-soliton solution of the KP equation: (a) Three time snapshots of the three-dimensional diagram and the corresponding two-dimensional cross-sectional diagrams at $x = 0$ and $y = 0$ (the blue solid line and the red dashed line correspond to the predicted solution and the exact solution, respectively); (b) density map of error (predicted solution minus exact solution).

in the $(2 + 1)$ dimension is vividly displayed. The cross-sectional views at $y = 0$ at three moments fully display the process of the tall-thin soliton chasing and surpassing the short-fat soliton. From the error density map, it can be found that the main error comes from the soliton with a high wave number, which also shows that a large gradient may bring a large prediction error. The error of 10^{-2} level is almost negligible compared to the wave height of the soliton, so the NN once again demonstrated excellent performance. Looking back at the more specific model results, the PINN passed $11,718$ iterations in 7868.21 s, and obtained a local optimal solution so that the loss and generalization error were reduced to 7.4780×10^{-4} and 4.7135×10^{-3}, respectively, which demonstrated extremely low training error and excellent generalization effect.

3.3. *The data-driven breather solution*

Parametric complexation of the two-soliton solution is a common method to obtain breathers. Let $n = 2$, (17) becomes a two-soliton solution, then the wave number and initial position are complexed to $k_1 = k + i\kappa, k_2 = k - i\kappa, p_1 = p + iP,$

$p_2 = p - iP, \xi_1^{(0)} = \xi + i\Xi, \xi_2^{(0)} = \xi - i\Xi$, thereby deriving the breathers of the KP equation[45]:

$$u_{\text{breather}} = 2\frac{\partial^2}{\partial x^2}\ln[2e^{\Omega}(\cos\omega + e^{\Lambda}\cosh(\Omega + \Lambda))], \tag{22}$$

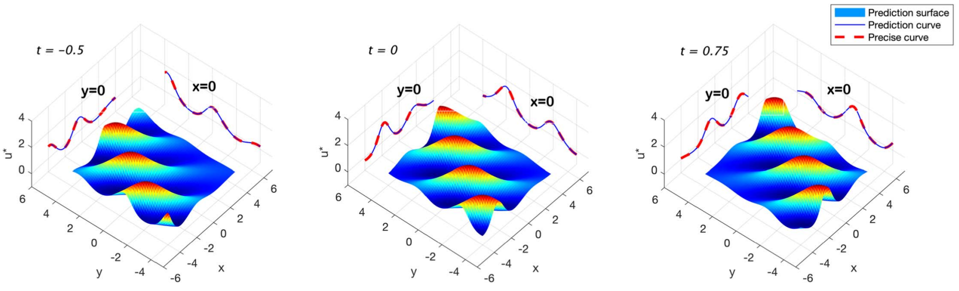
with

$$\Omega = kx + (kp - \kappa P)y + (2\kappa P p\sigma^2 + P^2 k\sigma^2 - kp^2\sigma^2 + 3\kappa^2 k - k^3)t + \xi, \tag{23}$$

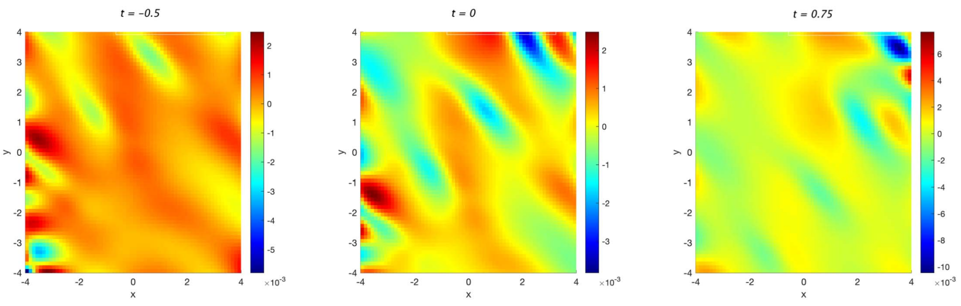
$$\omega = -\kappa x + (-\kappa p - Pk)y + (-\kappa^3 - P^2\sigma^2\kappa + p^2\sigma^2\kappa + 3k^2\kappa + 2kPp\sigma^2)t - \Xi, \tag{24}$$

$$\Lambda = \ln(\Lambda_0), \quad \Lambda_0 = \sqrt{\frac{P^2\sigma^2 - 3\kappa^2}{P^2\sigma^2 + 3k^2}}, \tag{25}$$

among them, if and only if $e^{\Lambda} > 1$, (22) becomes the analytical breather solution. After selecting the parameter as $\sigma^2 = -3, k = \frac{1}{2}, \kappa = \frac{7}{5}, p = 1, P = \frac{3}{5}, \xi = -\frac{6}{5}, \Xi = 0$, a specific breather solution $u_3(x, y, t)$ is obtained. Take $[-T, T] = [-1, 1], \Omega = [-4, 4] \times [-4, 4]$ and $u_0(x, y, t) = u_3(x, y, t)$ to determine the specific initial-boundary value problem. After discretizing each space-time dimension into $n_1 = n_2 = 65$ and $n_t = 33$, the LHS method is used to randomly



(a)



(b)

Fig. 6. (Color Online) The data-driven breather solution of the KP equation: (a) Three time snapshots of the three-dimensional diagram and the corresponding two-dimensional cross-sectional diagrams at $x = 0$ and $y = 0$ (the blue solid line and the red dashed line correspond to the predicted solution and the exact solution, respectively); (b) density map of error (predicted solution minus exact solution).

select $n_b = 6000$ points and $n_f = 50000$ points as the training data in the 1-*type* and 3-*type* of points, respectively. In addition, because of the periodic nature of the breathers, the sine function sin, which also has periodicity, is more suitable as the activation function. A 10-layer feedforward NN with 40 nodes per hidden layer performs 12,605 iterations within 7849.77 s to reach the local optimal solution. The loss value and generalization error of this optimal solution are 1.7919×10^{-5} and 1.9162×10^{-3}, respectively. Figure 6 shows that PINN has learned the breather solution well.

Figure 6 displays the three-dimensional profile, the time snapshot of the two-dimensional profile and the error density map of the data-driven breather solution. The "breathing" feature of the breathers is very obvious in Fig. 6(a), and the breather wave moves to the positive direction of the x-axis and the negative direction of the y-axis as time progresses. Although the error density map is slightly blurred, it can still be found that the large gradient may be the weakness of the NN, but the performance on the overall generalization error is impeccable. All in all, the PINN once again successfully learned the breather solution of the KP equation.

3.4. *The data-driven lump solution*

This section will discuss the lump solution based on the PINN. Ma used the bilinear form (16) of the KP equation to construct an analytical lump solution,[46] and f_{lump} has the following form:

$$f_{\text{lump}} = (a_1x + a_2y + a_3t + a4)^2 + (a_5x + a_6y + a_7t + a_8)^2 + \frac{3(a_1^2 + a_5^2)^3}{(a_1a_6 - a_2a_5)^2}, \quad (26)$$

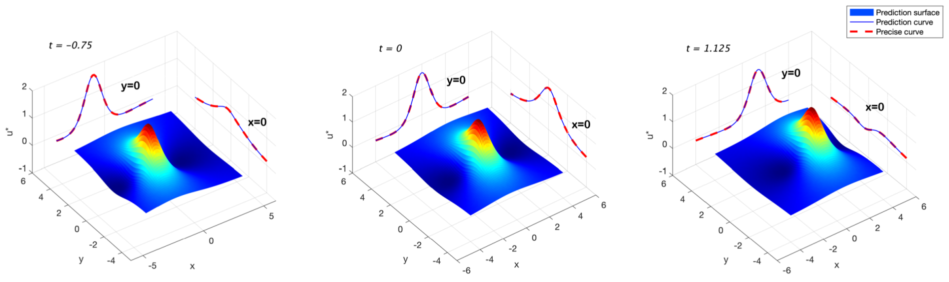
with

$$a_3 = \frac{a_1a_2^2 - a_1a_6^2 + 2a_2a_5a_6}{a_1^2 + a_5^2}, \quad a_7 = \frac{2a_1a_2a_6 - a_2^2a_5 + a_5a_6^2}{a_1^2 + a_5^2}, \quad (27)$$

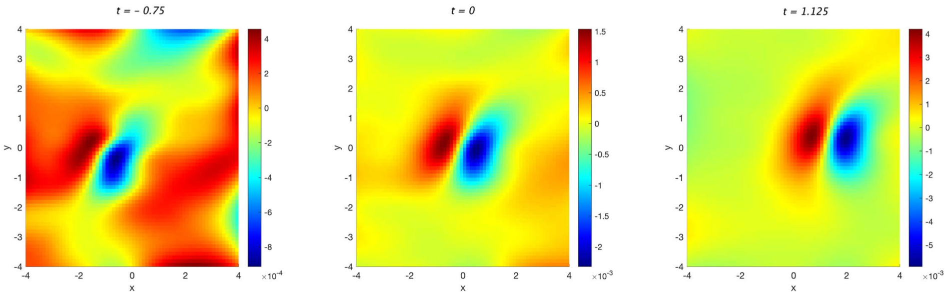
where $a_1, a_2, a_4, a_6, a_7, a_8$ are six free parameters, and need to satisfy a determinant condition $a_1a_6 - a_2a_5 \neq 0$. And Eq. (26) is actually the solution of bilinear form with parameter $\sigma^2 = -1$ (KPI). Taking the free parameter as $a_1 = 3, a_2 = 4, a_4 = 0, a_5 = -4, a_6 = 4, a_8 = 0$, the lump solution of the KPI equation is denoted as $u_4(x, y, t)$.

In the initial-boundary value problem (14), let $[-T, T] = [-1.5, 1.5], \Omega = [-4, 4] \times [-4, 4]$ and $u_0(x, y, t) = u_4(x, y, t)$. A 10-layer NN with 40 nodes per hidden layer is constructed to learn the lump solution of the KP equation. The parameters involved in the discretization and random sampling of data are selected as $n_1 = n_2 = 65, n_t = 33, n_b = 5000, n_f = 50000$.

Figure 7 displays the three-dimensional contour map, two-dimensional cross-sectional view and the corresponding error density map of the data-driven lump solution. A lump with a peak and two adjacent troughs moves in the positive direction along the x-axis and y-axis, which is evident in Fig. 7(a). The two main error regions in the error density map correspond to the points near the lump



(a)



(b)

Fig. 7. (Color online) The data-driven lump solution of the KP equation: (a) Three time snapshots of the three-dimensional diagram and the corresponding two-dimensional cross-sectional diagrams at $x = 0$ and $y = 0$ (the blue solid line and the red dashed line correspond to the predicted solution and the exact solution, respectively); (b) density map of error (predicted solution minus exact solution).

extreme, which once again confirms that the generalization of large gradient regions is a challenge for PINN. The NN undergoes 3833 iterations in 2149.91 s, and obtains a local optimal solution with a generalization error of 1.7836×10^{-3}. Combined with the magnitude of the error displayed by the density map, the dynamic behavior of the lump solution is well reproduced.

4. The Application of PINN Method to $(3 + 1)$-Dimensional Reduced KP Equation

Rogue waves were first discovered in the ocean as a natural disaster. Afterwards, the rogue wave phenomenon was discovered to exist in various fields such as optics,[47] Bose–Einstein condensates[48] and even finance,[49] so it has become one of the most active and important research fields in experimental observation and theoretical analysis. Because of the rogue wave's characteristic of "coming without a shadow and going without a trace", it has always been difficult to observe it. Therefore, scientists have been working to explain the mysterious dynamic behavior and formation mechanism of rogue waves through NPDEs. In this section, we use the PINN method to reproduce the dynamic behavior of a resonance rogue, which is the first discovery under the data-driven mechanism.

First of all, consider a $(3+1)$-dimensional KP equation[50]

$$(u_t + h_1 uu_x + h_2 u_{xxx} + h_3 u_x)_x + h_4 u_{yy} + h_5 u_{zz} = 0, \tag{28}$$

where $u = u(x, y, z, t)$ and $h_i, 1 \leq i \leq 6$ are the equation parameters. Equation (28) has a wide range of applications in plasma.[51,52] Let the parameters $h_1 = -1$, $h_2 = -\frac{1}{3}, h_3 = 1, h_4 = 1, h_5 = -\frac{2}{3}$, and take $x = z$, then Eq. (28) is reduced to

$$\left(u_t - uu_x - \frac{1}{3}u_{xxx} + \frac{1}{3}u_x\right)_x + u_{yy} = 0, \tag{29}$$

through variable transformation $u = 4(\ln f)_{xx}$, its bilinear equation is obtained

$$\begin{aligned}(D_x D_t - \frac{1}{3}D_x^4 + \frac{1}{3}D_x^2 + D_y^2)f \cdot f \\ = 2f_{xy}f - 2f_x f_t - \frac{2}{3}f_{xxxx}f \\ + \frac{8}{3}f_{xxx}f_x - 2f_{xx}^2 + \frac{2}{3}f_{xx}f - \frac{2}{3}f_x^2 + 2f_{yy}f - 2f_y^2 = 0,\end{aligned} \tag{30}$$

where $D_x^n D_y^m D_t^l\ f \cdot g$ is the Hirota bilinear operator.[8] Rational solutions account for the vast majority of the known rogue wave theories. Zhang *et al.* received inspiration from the interaction between lump and single soliton and proposed a new nonlinear mechanism for generating rogue waves,[53] recently. They introduced a new combination of positive quadratic function and hyperbolic function, and obtained the resonance rogue of the (3+1)-dimensional reduced KP equation. This resonance rogue is actually the interaction solution between lump and resonance two solitons, based on the bilinear form (30), f is written as

$$f = g^2 + h^2 + \lambda\cosh(c_1 x + c_2 y + c_3 t) + b_9, \tag{31}$$

with

$$g = b_1 x + b_2 y + b_3 t + b_4, \quad h = b_5 x + b_6 y + b_7 t + b_8, \tag{32}$$

where the lump part is expressed as positive quadratic functions, and the two-soliton part is expressed as a hyperbolic function. Substitution of (31) into (30) leads to

$$\begin{aligned} b_6 &= \frac{b_2 b_5 \pm b_1^2 c_1 \pm b_5^2 c_1}{b_1}, \quad b_9 = \frac{\lambda^2 c_1^4 + 4b_1^4 + 8b_1^2 b_5^2 + 4b_5^4}{4c_1^2(b_1^2 + b_5^2)}, \\ c_2 &= \frac{c_1(b_2 \pm b_5 c_1)}{b_1}, \quad c_3 = \frac{c_1(b_1^2 c_1^2 - 3b_5^2 c_1^2 \mp 6b_2 b_5 c_1 - b_1^2 - 3b_2^2)}{3b_1^2}, \\ b_3 &= \frac{3b_1^2 c_1^2 + 3b_5^2 c_1^2 - b_1^2 - 3b_2^2}{3b_1}, \\ b_7 &= -\frac{3b_1^2 b_5 c_1^2 + 3b_5^3 c_1^2 \pm 6b_1^2 b_2 c_1 \pm 6b_2 b_5^2 c_1 + b_1^2 b_5 + 3b_2^2 b_5}{3b_1^2}. \end{aligned} \tag{33}$$

Therefore, under the first type of solution, the solution u is

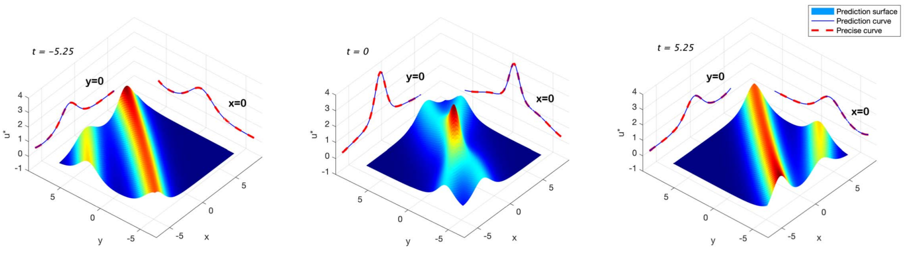
$$u = \frac{4(2b_1^2 + 2b_5^2 + \lambda\cosh(c_1x + c_2y + c_3t)c_1^2)}{f} - 4\frac{(2b_1g + 2b_5h + \lambda c_1\sinh(c_1x + c_2y + c_3t))^2}{f^2}, \tag{34}$$

where f, g, h satisfy the constraints in (31)–(33). Take the parameter $b_1 = \frac{3}{5}, b_2 = -\frac{3}{5}, b_4 = b_5 = b_8 = 0, \lambda = \frac{6}{5}, c_1 = \frac{6}{5}$ to obtain a specific interaction solution $u_5(x, y, t)$.

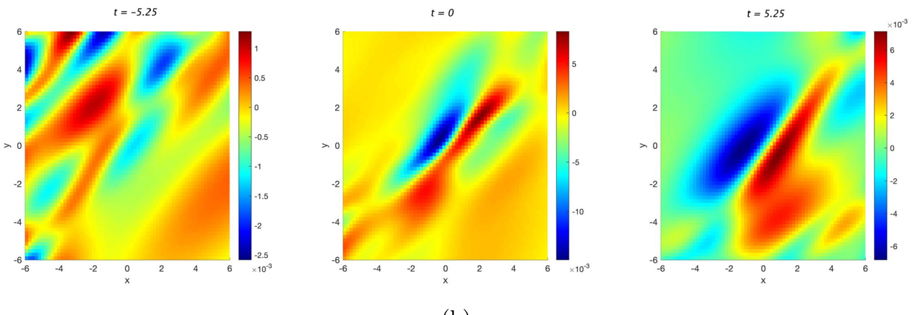
The peaks of rogue waves usually have a larger gradient, so the learning of rogue waves is more difficult than other local waves. Next, we will discuss the performance of the PINN method on the high-dimensional rogue. First, we construct an initial-boundary value problem for the $(3 + 1)$-dimensional reduced KP equation

$$\begin{cases} \left(u_t - uu_x - \dfrac{1}{3}u_{xxx} + \dfrac{1}{3}u_x\right)_x - \dfrac{2}{3}u_{yy} = 0, \\ u(x, y, -T) = u_0(x, y, -T), \quad \forall\,(x, y) \in \Omega, \\ u(x, y, t) = u_0(x, y, t), \quad \forall\,(x, y) \in \partial\Omega,\ t \in [-T, T], \end{cases} \tag{35}$$

where we select $[-T, T] = [-6, 6], \Omega = [-6, 6] \times [-6, 6]$ and $u_0(x, y, t) = u_5(x, y, t)$. A 11-layer NN with 40 nodes per hidden layer is constructed to learn the interaction



(a)



(b)

Fig. 8. (Color online) The data-driven interaction solution of the reduced KP equation: (a) Three time snapshots of the three-dimensional diagram and the corresponding two-dimensional cross-sectional diagrams at $x = 0$ and $y = 0$ (the blue solid line and the red dashed line correspond to the predicted solution and the exact solution, respectively); (b) density map of error (predicted solution minus exact solution).

solution of the $(3+1)$-dimensional reduced KP equation. The parameters involved in the discretization and random sampling of data are selected as $n_1 = n_2 = n_t = 65, n_b = 10000, n_f = 50000$. The NN undergoes 3616 iterations in 2368.08 s, and obtains a local optimal solution with a generalization error of 2.6201×10^{-3}. The data-driven interaction solution and the corresponding error are shown as

Figure 8(a) shows that the lump is hidden like a ghost soliton and only appears in the middle of the two linear solitons, that is to say, the resonance rogue appears near $x = y = t = 0$. The main reason for this phenomenon is that the algebraic decay of lump is slower than the exponential decay of the resonance two solitons. The generalization error mainly comes from the larger gradient area, which is also obvious in (b). The magnitude of the error tells us that PINN successfully reproduced the dynamic behavior of the resonance rogue. This proves once again that the combination of integrable system and PINN will have unexpected results.

5. Conclusion

The PINN method based on physical constraints provides a possibility for solving high-dimensional problems. But the instability of the system itself will exacerbate the difficulties caused by the curse of dimensionality. As a class of NPDEs with excellent properties, integrable systems have abundant local wave solutions and high tolerance to errors (the trajectories will not be exponentially separated).

Here, the PINN method is applied to integrable systems, after fully considering the advantages of both. The framework for applying the PINN method to solve the $(N+1)$-dimensional initial-boundary value problem with $2N+1$ hyperplane boundaries is presented in this paper. And at low time and space cost, the dynamic behavior of various local waves in the high-dimensional integrable system is reproduced, such as single soliton, two solitons, breathers, lump of the $(2+1)$-dimensional KP equation, and the interaction solution (resonance rogue) of the $(3+1)$-dimensional reduced KP equation. From the error results of the model, the PINN shows outstanding generalization ability even in the high-dimensional integrable systems. This proves that the integrable systems will be suitable places for the PINN method to function, especially in the case of high-dimensional.

However, from the error density map, we discovered that the error mainly comes from the region of large gradient, which means that network optimization for the region of large gradient may be the direction of future efforts. In addition, more interesting structures in integrable systems may speed up the process of minimizing the loss function or improve the accuracy of generalization. These will become our future work.

Acknowledgments

The project is supported by the National Natural Science Foundation of China (No. 12175069) and Science and Technology Commission of Shanghai Municipality (Nos. 21JC1402500 and 18dz2271000).

References

1. T. Gustafsson, K. R. Rajagopal, R. Stenberg and J. Videman, *Appl. Math. Model.* **39** (2015) 5299.
2. D. J. Kaup and A. C. Newell, *J. Math. Phys.* **19** (1978) 798.
3. A. D. Polyanin and A. I. Zhurov, *Int. J. Non-Linear Mech.* **67** (2014) 267.
4. A. S. Parkins and D. F. Walls, *Phys. Rep.* **303** (1998) 1.
5. V. B. Matveev and M. A. Salle, *Darboux Transformations and Solitons* (Springer Verlag, 1991).
6. M. J. Ablowitz and P. A. Clarkson, *Solitons, Nonlinear Evolution Equations and Inverse Scattering* (Cambridge University Press, Cambridge, 1991).
7. C. Rogers and W. K. Schief, *Bäcklund and Darboux Transformations: Geometry and Modern Applications in Soliton Theory* (Cambridge University Press, Cambridge, 2002).
8. R. Hirota, *The Direct Method in Soliton Theory* (Cambridge University Press, Cambridge, 2004).
9. B. Wongsaijai, T. Aydemir, T. Ak and S. Dhawan, *Numer. Methods Partial Differ. Equ.* (2020), doi:10.1002/num.22693.
10. A. Krizhevsky, I. Sutskever and G. E. Hinton, ImageNet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems*, 2012.
11. B. M. Lake, R. Salakhutdinov and J. B. Tenenbaum, *Science* **350** (2015) 1332.
12. Y. LeCun, Y. Bengio and G. Hinton, *Nature* **521** (2015) 436.
13. W. Zaremba, I. Sutskever and O. Vinyals, *Recurrent Neural Network Regularization* arXiv:1409.2329.
14. M. Raissi, P. Perdikaris and G. E. Karniadakis, *J. Comput. Phys.* **378** (2019) 686.
15. M. Raissi, P. Perdikaris and G. E. Karniadakis, *J. Comput. Phys.* **335** (2017) 736.
16. M. Raissi, P. Perdikaris and G. E. Karniadakis, *J. Comput. Phys.* **348** (2017) 683.
17. Y. B. Yang and P. Perdikaris, *J. Comput. Phys.* **394** (2019) 136.
18. L. Yang, X. H. Meng and G. E. Karniadakis, *J. Comput. Phys.* **425** (2021) 109913.
19. X. W. Jin, S. Z. Cai, H. Li and G. E. Karniadakis, *J. Comput. Phys.* **426** (2021) 109951.
20. A. D. Jagtap, K. Kawaguchi and G. E. Karniadakis, *J. Comput. Phys.* **404** (2020) 109136.
21. M. Raissi, A. Yazdani and G. E. Karniadakis, *Science* **367** (2020) 1026.
22. G. Pang, L. Lu and G. E. Karniadakis, *SIAM J. Sci. Comput.* **41** (2019) A2603.
23. D. Zhang, L. Guo and G. E. Karniadakis, *SIAM J. Sci. Comput.* **42** (2020) A639.
24. J. Li and Y. Chen, *Commun. Theor. Phys.* **72** (2020) 105005.
25. J. C. Pu, J. Li and Y. Chen, *Chin. Phys. B* **30** (2021) 060202.
26. J. C. Pu, J. Li and Y. Chen, *Nonlinear Dyn.* **105** (2021) 1723.
27. W. Q. Peng, J. C. Pu and Y. Chen, *PINN deep learning for the Chen–Lee–Liu equation: rogue wave on the periodic background*, arXiv:2105.13027.
28. S. N. Lin and Y. Chen, *A two-stage physics-informed neural network method based on conserved quantities and applications in localized wave solutions*, arXiv:2107.01009.
29. L. Wang and Z. Y. Yan, *Phys. Lett. A* **404** (2021) 127408.
30. Y. Fang, G. Z. Wu, Y. Y. Wang and C. Q. Dai, *Data-driven femtosecond optical soliton excitations and parameters discovery of the high-order NLSE using the PINN*, arXiv:2103.16297.
31. S. Pattanayak, *Pro Deep Learning with TensorFlow* (Springer, 2017).
32. A. Ghatak, *Initialization of Network Parameters* (Springer Verlag, 2019).

33. K. M. He, X. Y. Zhang, S. Q. Ren and J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, in *2015 IEEE Int. Conf. Computer Vision* (*ICCV*) (IEEE, 2015), pp. 1026–1034.
34. H. Y. Zhang, Y. N. Dauphin and T. Y. Ma, Fixup initialization: Residual learning without normalization, in *7th Int. Conf. Learning Representations* (*ICLR*), 2019.
35. M. Stein, *Technometrics* **29** (1987) 143.
36. A. G. Baydin, B. A. Pearlmutter, A. A. Radul and J. M. Siskind, *J. Mach. Learn. Res.* **18** (2018) 1.
37. D. C. Liu and J. Nocedal, *Math. Program.* **45** (2989) 503.
38. L. Bottou, *Neural Networks: Tricks of the Trade* (Springer, 2012).
39. S. Ruder, *An overview of gradient descent optimization*, arXiv:1609.04747.
40. F. Girosi, M. Jones and T. Poggio, *Neural Comput.* **7** (1995) 219.
41. B. B. Kadomtsev and V. I. Petviashvili, *Sov. Phys. Dokl.* **15** (1970) 539.
42. J. Hammack, N. Scheffner and H. Segur, *J. Fluid Mech.* **209** (1989) 567.
43. D. E. Pelinovsky, Y. A. Stepanyants and Y. S. Kivshar, *Phys. Rev. E* **51** (1995) 5016.
44. R. Hirota and J. Satsuma, *J. Phys. Soc. Jpn.* **40** (1976) 286.
45. Z. D. Dai, S. L. Li, Q. Y. Dai and J. Huang, *Chaos Solitons Fractals* **34** (2007) 1148.
46. W. X. Ma, *Phys. Lett. A* **379** (2015) 1975.
47. D. R. Solli, C. Ropers, P. Koonath and B. Jalali, *Nature* **450** (2007) 1054.
48. Y. V. Bludov, V. V. Konotop and N. Akhmediev, *Phys. Rev. A* **80** (2009) 033610.
49. Z. Y. Yan, *Phys. Lett. A* **375** (2011) 4274.
50. J. M. Tu, S. F. Tian, M. J. Xu, X. Q. Song and T. T. Zhang, *Nonlinear Dyn.* **83** (2016) 1199.
51. U. K. Samanta, A. Saha and P. Chatterjee, *Phys. Plasmas* **20** (2013) 022111.
52. A. Saha, N. Pal and P. Chatterjee, *Braz. J. Phys.* **45** (2015) 325.
53. X. E. Zhang, Y. Chen and X. Y. Tang, *Comput. Math. Appl.* **76** (2018) 1938.