Nonlinear Dyn (2023) 111:16467–16482
https://doi.org/10.1007/s11071-023-08712-3

ORIGINAL PAPER

Mix-training physics-informed neural networks for high-order rogue waves of cmKdV equation

Shifang Tian · Zhenjie Niu · Biao Li

Received: 9 March 2023 / Accepted: 6 June 2023 / Published online: 25 July 2023

Abstract Recently, a neural networks method: physics-informed neural networks (PINNs) is proposed to solve nonlinear partial differential equations (NPDEs); this method obtained the predicted solution by minimizing the sum of mean square of initial error, boundary error and residual of equation. It provides a new approach for solving NPDEs by neural networks. The complex modified Korteweg–de Vries (cmKdV) equation is a classical integrable equation which contains plenty of significant properties and occurs in many physical areas. However, due to its complexity, it is difficult to solve the high-order rogue waves of them by PINNs, so how to solve the high-order rogue waves of complex equation by neural networks method has become a hot research direction. In this paper, based on PINNs, we propose two neural networks methods: mix-training physics-informed neural networks (MTPINNs) and prior information mix-training physics-informed neural networks (PMTPINNs). Numerical experiments have shown that the original PINNs are completely unable to simulate the high-order rogue waves of the cmKdV equation, but our proposed models not only simulate these high-order rogue waves, but also significantly improve the simulation ability, increasing prediction accuracy by three to four orders of magnitude. The inverse problem of these models is tested by some noise data, which also proves that these models have good robustness. The above results validate the superiority of our proposed models in simulating high-order rogue waves of cmKdV equation, and these models provide us some new insights for studying the dynamic characteristics of high-order rogue waves using neural networks methods.

Keywords Physics-informed neural networks · cmKdV equation · High-order rogue waves · Mix-training · Prior information

S. Tian · Z. Niu · B. Li (✉)
School of Mathematics and Statistics, Ningbo University, Ningbo 315211, People's Republic of China
e-mail: libiao@nbu.edu.cn

S. Tian
e-mail: 18658277095@163.com

Z. Niu
e-mail: nzj951001@163.com

1 Introduction

Nonlinear problems exist in many fields, such as fluid mechanics, plasma physics and fiber optic communications [1–4]. Nonlinear phenomena in nature are described as simple mathematical physical models and expressed by nonlinear partial differential equations (NPDEs). Because of the importance of nonlinear problems, solving NPDEs has been the focus of researchers in recent decades.

With the development of computer and the arrival of the era of *big data*, due to its powerful "learning" ability, deep learning has had a significant impact in many fields, such as computer vision (unmanned driving, face recognition, VR/AR, medical image analysis, etc.), rec-

ommendation systems (help users find information that interests them in an information overload environment and push it to them) and natural language processing (syntactic semantic analysis, information extraction, machine translation, information retrieval, etc.) [5–8]. An important reason behind the success of deep learning is the use of neural networks. Because the key step to obtain the predictive solution of NPDEs through deep learning is to constrain the neural networks to minimize the loss function. Thus, the use of deep neural networks methods to solve NPDEs has attracted extensive attention, and some models [9–12] optimized on the basis of them have emerged, greatly promoting the development of this field. Among these works, physics-informed neural networks (PINNs) proposed by Raissi et al. [9] are particularly outstanding. This method uses the standard back-propagation (BP) neural networks, generating the function through the automatic differentiation method of the BP neural networks, and then obtain the final predicted solution by minimizing the sum of mean square of initial error, boundary error and residual of NPDEs. In addition, compared with traditional mesh-based methods such as finite difference method and finite element method, PINNs make the method meshless because of its automatic differential method. Therefore, PINNs have attracted much attention because of their good generalization ability and prediction performance in solving NPDEs, Bihlo et al. [13] simulated the shallow-water equations on the sphere by PINNs. Through PINNs, Luo et al. [14] simulated data-driven solutions of the Sasa–Satsuma equation and solved parameter discovery problem. Li et al. [15] solved the forward and inverse problems of the nonlinear Schrödinger equation with the generalized $\mathcal{PT}$-symmetric Scarf-II potential via PINNs. Chen et al. [16] simulated data-driven vector localized waves of Manakov system by PINNs and solved the parameter discovery problem of Manakov system.

Despite certain positive outcomes, PINNs have encountered unforeseen challenges in approximating potential solutions. Certain equations display diminished simulation precision and, at times, fail to be simulated. Consequently, the optimization of neural networks methodologies to enable novel models to simulate equation solutions unattainable by conventional PINNs, or to procure more precise predictive solutions, has emerged as a critical research focus among scholars. Recently, there have been proposals for neural networks methods based on PINNs, involving the optimization of neural networks architectures or the loss function: Mean Squared Error (MSE). For instance, Jagtap et al. [17] introduced an advanced PINNs method by altering the training area of the PINNs loss function and augmenting the energy conservation law constraint, employing this method to simulate the inverse problem in supersonic flow. Mattey et al. [18] put forth a novel sequential PINNs model by incorporating additional loss function constraints, simulating Alan Cahn and Kahn Hilliard equations with this method. Rezaei et al. [19] amalgamated the finite element method, adding the finite element boundary constraint to the loss function to propose a hybrid PINNs, examining issues in heterogeneous domains via this composite PINNs. Chen and his team [20–23] revised the loss function by appending a slope recovery term, proposing an IPNNs method. This approach was used to simulate data-driven rogue waves and soliton solutions of classical mathematical physics equations such as the Sin-Gordon equation, nonlinear Schrödinger equation (NLSE), and derivative NLSE. The team also employed two identical neural networks, incorporating the sum of the mean squared deviations of the prediction solutions acquired from the two neural networks as constraints to the loss function of the second neural networks, they proposed a two-stage PINNs [24] and simulated the rational waves of a class of nonlinear physical equations. Ling et al. [25] altered the training area, selecting points other than boundary errors, initial errors, and equation residuals per the established algorithm. They added the sum of the mean square errors corresponding to these points into the loss function as a constraining condition, and proposed a pre-fixed multi-stage PINNs, thereby obtaining the data-driven vector soliton solution of coupled NLSE. Yan et al. [26–29] proposed an enhanced PINNs by modifying the loss function and incorporating the constraint conditions of the first-order derivative boundary loss function term, predicting the peakon solution and periodic solution of the Camassa–Holm-like equations with this improved PINNs. Building upon the PINNs model, Li et al. [30–32] revised the loss function, setting the coefficients of the boundary loss function and the initial loss function as variables, thereby proposing a gradient-optimized PINNs; they also altered the training area by merging the boundary area with the initial area to create a new training area, subsequently reducing computer error, and proposed a mix-training PINNs. These two models further enhance the accuracy of the simulation solution.

Dai et al. [33–35] proposed the mixed PINNs method by modifying the training region of PINNs loss function and increasing the constraint of the energy conservation law, simulating the three-component coupled NLSE, KdV equation, and mKdV equation.

Building on extant research findings, we contemplate the possibility of proposing a novel neural networks model of superior performance through modifications to the loss function MSE, without altering the structure of the neural networks. Consequently, this study amalgamates the initial sampling region I and boundary sampling region B in the original PINNs model into a single mix-training sampling region IB, thereby proposing a mix-training physics-informed neural networks (MTPINNs) model. Furthermore, predicated on the MTPINNs model, we select certain sample points from the sharp region of the equation solution, a region encompassing specific points that instigate substantial oscillations in preceding and subsequent solution values. These sample points are integrated as prior information into the mixed training sampling region IB, forming a novel sampling region IBP and leading to the proposition of a prior information mix-training PINNs (PMTPINNs).

Rogue waves constitute a compelling physical phenomenon, manifesting in various domains such as optics, ocean dynamics, and plasma, among others. Given the instability and unpredictability of rogue waves, studying them and applying the research findings to practical situations hold significant value. This could, for instance, include strategies to mitigate the damages inflicted on ships by rogue waves during maritime voyages, or measures to prevent harm to seafaring personnel due to these waves. Recently, several scholars have employed neural networks models to simulate rogue waves of certain classical integrable equations, procuring commendable results [18,20,23,27,29]. However, the current neural networks models are unable to simulate the high-order rogue waves of some intricate NPDEs, such as the high-order rogue waves of the cmKdV equation [22]. Thus, our objective is to propose a neural networks methodology capable of simulating high-order rogue waves of the cmKdV equation. In this paper, we will simulate the second-order and third-order rogue waves of the cmKdV equation utilizing our proposed MTPINNs and PMTPINNs models.

The structure of this paper is as follows: in Sect. 2, we introduce the neural networks' structure; in Sect. 3, we first review the PINNs model, then introduce MTPINNs and PMTPINNs; in Sect. 4, we use PINNs, MTPINNs and PMTPINNs to simulate second-order and third-order rogue waves of the cmKdV equation, and make data-driven coefficient discovery (inverse problem) for the cmKdV equation. Then, by checking the numerical results, we found that the PINNs could not simulate high-order rogue waves of cmKdV equation, but MTPINNs and PMTPINNs we proposed could do so, and these models' performance is good. Finally, Sect. 5 gives the conclusion and discussion.

2 Neural networks

This section provides an overview of neural networks, with a specific focus on Backpropagation (BP) neural networks. The BP neural networks learning algorithm, according to Li et al. [36], is currently the most triumphant methodology utilized in neural networks training. Typically, a BP neural networks comprises input layers, hidden layers, output layers, neurons, and an activation function.

The process of solution derivation and error examination within the BP neural networks model is as follows: Initially, the relevant weights w and biases b are established. Following this, the input values, such as x and t, are set, along with the quantity of neurons in each hidden layer. The corresponding weights w and biases b are then allocated to the inputs x and t. Executing a sequence of weighted summations on the inputs x and t yields $u(x,t)$, which, when applied to the activation function, produces $\hat{u}(x,t)$, this is then transmitted to the subsequent layer within the neural networks. It is worth noting that the implementation of activation function is used to modify the linear relationship of the previous data to provide practicability for the layer of neural networks. This capability enables the networks to learn complex objects and data, representing nonlinear relationships between the inputs and outputs of any complex function mapping. This sequence continues up to the penultimate layer, wherein the activation function is no longer utilized, and the output layer directly generates the final predictive solution $Y(x,t)$. Then, an optimizer is selected to optimize the obtained $Y(x,t)$. If the mean squared error (MSE) derived from the juxtaposition of expected and predicted results exceeds ε or if the iteration count has not reached the maximum value, the predictive solution $Y(x,t)$ is sent back to the

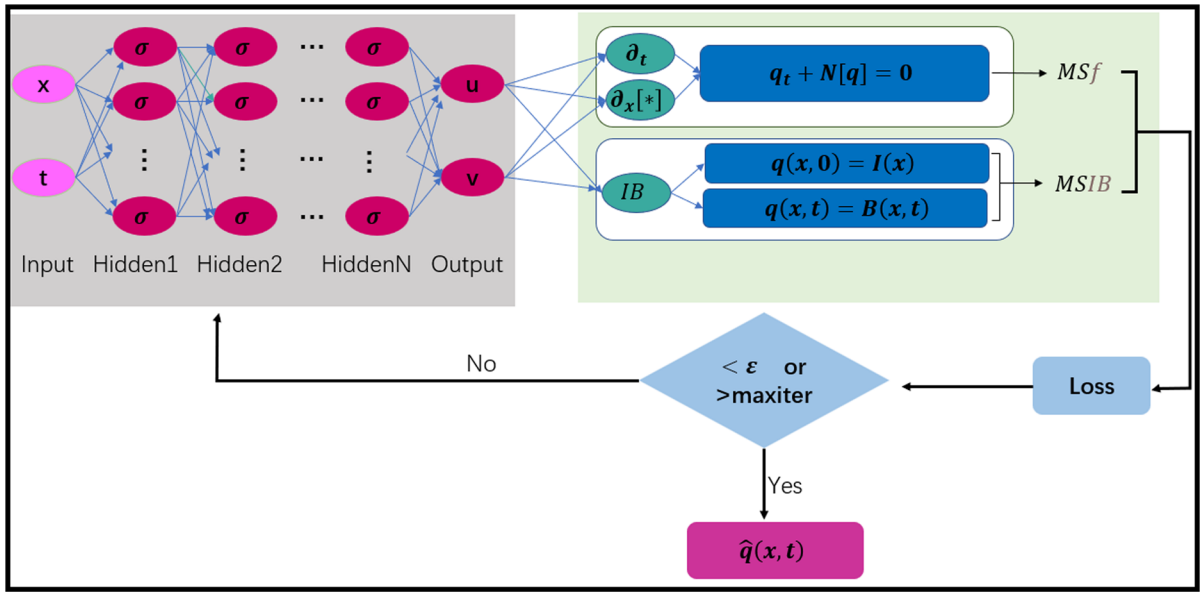

Fig. 1 Mix-training physics-informed neural networks (MTPINNs)

input layer to update the weights w and biases b and repeat the process. The MSE will continue to decrease until it is less than ε or the iteration count reaches the maximum value, thereby concluding the loop and producing the final predictive solution for the equation.

In essence, the BP neural networks as an "error correction" function. In this study, we have employed two optimization algorithms, namely L-BFG-S and Adam, to achieve optimal results. The Adam optimization algorithm, as described in [37], is a traditional form of the stochastic gradient descent algorithm, while the L-BFG-S optimization algorithm, outlined in [38], is a full-batch gradient descent optimization algorithm based on the quasi-Newton method.

As shown in Fig. 1, input and output represent input layer and output layer, and Hidden1, Hidden2, …, HiddenN represent hidden layer. $\mathcal{I}(x)$ is an initial value condition, $\mathcal{B}(x,t)$ is a boundary condition. MSf is the equation residual; $MSIB$ is the loss function on the mixed region IB, the mixed region IB consists of initial region I and boundary region B. ε is a controllable parameter, and maxiter is the largest step of iteration. The above two parameters are used as the loop-continuation condition. Finally, the predicted solution $\hat{q}(x,t)$ is obtained at the end of the loop.

3 Model setup

3.1 PINNs

First, we will briefly introduce the PINNs model. The NPDEs solved by PINNs are as follows [9]:

$$\begin{cases} q_t + N[q] = 0, (x,t) \in \Omega \times T, \\ q(x,t) = \mathcal{I}(x,t), x \in \Omega, t = t_0, \\ q(x,t) = \mathcal{B}(x,t), (x,t) \in \partial\Omega \times T, \end{cases} \tag{1}$$

here T and Ω represent the time domain and spatial domain of the equation, respectively, and $\partial\Omega$ represent the boundary region of the equation in the spatial direction. $N[q]$ is the combination of nonlinear and linear operators of $q(x,t)$, and $\mathcal{I}(x)$, $\mathcal{B}(x,t)$ is the initial condition and boundary condition of the equation, respectively. In addition, in order to calculate the loss function subsequently, we let $f = q_t + N[q]$ is the equation residual.

We need to simulate the original equation through neural networks, so as to simulate the predicted solution $\hat{q}(x,t)$ of the original equation. Using PINNs to solve NPDEs with initial conditions and boundary conditions requires a loss function MSE and gradient descent method is used to process the loss function, so we make the loss function [9] as follows:

$$
\begin{aligned}
MSE_{PINNs} &= MSf + MSI + MSB \\
&= \frac{1}{N_f}\sum_{i=1}^{N_f}[f(x^i,t^i)]^2 \\
&\quad + \frac{1}{N_{\mathcal{I}}}\sum_{i=1}^{N_{\mathcal{I}}}[\hat{q}(x^i_{\mathcal{I}},t_0) - q(x^i_{\mathcal{I}},t_0)]^2 \\
&\quad + \frac{1}{N_{\mathcal{B}}}\sum_{i=1}^{N_{\mathcal{B}}}[\hat{q}(x^i_{\mathcal{B}},t^i_{\mathcal{B}}) - q(x^i_{\mathcal{B}},t^i_{\mathcal{B}})]^2
\end{aligned} \tag{2}
$$

where the loss function consists of three parts, namely, equation residual MSf, initial condition error MSI and boundary condition error MSB. N_f is the number of sampling points in the whole area, $N_{\mathcal{I}}$ is the number of sampling points in the initial area, and $N_{\mathcal{B}}$ is the number of sampling points in the boundary conditions.

The above sampling points are random sampling in a specific area. Finally, the two optimizers use the gradient descent method to minimize the loss function MSE until reach convergence.

3.2 MTPINNs

Although the PINNs has achieved success in many fields, there are still many difficulties in solving some complex NPDEs. Because the loss function of the model is calculated by sampling from the whole area MSf, from the initial value area MSI, and from the boundary area MSB. Finally, the mean square of the three parts is added to form the final loss function MSE. Therefore, when simulating some complex NPDEs, the accuracy may be too low or the solution of the equation cannot be simulated.

In this part, we try to combine the boundary region B and initial value region I into one region IB; that is, we directly proceed sampling point training from this new region. In theory, we can continue to reduce the loss function MSE, and the error of the corresponding boundary conditions and initial conditions also changes from $MSB+MSI$ to $MSIB$. At this time, the new loss function [32] is:

$$
\begin{aligned}
MSE_{MTPINNs} &= MSf + MSIB \\
&= \frac{1}{N_f}\sum_{i=1}^{N_f}[f(x^i,t^i)]^2 \\
&\quad + \frac{1}{N_{\mathcal{IB}}}\sum_{i=1}^{N_{\mathcal{IB}}}[\hat{q}(x^i_{\mathcal{IB}},t^i_{\mathcal{IB}}) \\
&\quad - q(x^i_{\mathcal{IB}},t^i_{\mathcal{IB}})]^2.
\end{aligned} \tag{3}
$$

where the loss function is composed of two parts, namely equation residuals MSf, error $MSIB$ in the mixed region IB composed of initial conditions and boundary conditions. $\hat{q}(x^i_{\mathcal{IB}},t^i_{\mathcal{IB}})$ is our predicted solution and $q(x^i_{IB},t^i_{IB})$ is the true solution of the equation; N_f is the number of sampling points in the whole region, and $N_{\mathcal{IB}}$ is the number of sampling points in the mixed region. In order to better compare the quality of methods, we make $N_{\mathcal{IB}}$ and $N_{\mathcal{I}}$, $N_{\mathcal{B}}$ of PINNs to meet the following conditions:

$$
N_{\mathcal{IB}} = N_{\mathcal{I}} + N_{\mathcal{B}} \tag{4}
$$

The sampling methods are the same, but the sampling areas are different.

3.3 PMTPINNs

When simulating the existing research results, we found that sometimes the sampling area is too random, so many points with large errors are not selected as training points. Therefore, we consider whether it is possible to add the constraint conditions for selecting training points, so that some points with large errors can be selected for training. At this time, we list the sampling points of the sharp area of the equation solution in advance, and then, they are added as prior information to the previous IB training area, generating a new area IBP. Then, when sampling again, it will be carried out in a new area, so that the method can be further optimized. This idea combines the idea of adaptive grid used in the finite difference method. The idea of adaptive grid is to reduce the number of grid nodes in the area with small error, and densify the grid in the area with large error. At this time, the new loss function [32] is defined as:

$$
\begin{aligned}
MSE_{PMTPINNs} &= MSf + MSIBP \\
&= \frac{1}{N_f}\sum_{i=1}^{N_f}[f(x^i,t^i)]^2 \\
&\quad + \frac{1}{N_{\mathcal{I}BP}}\sum_{i=1}^{N_{\mathcal{I}BP}}[\hat{q}(x^i_{\mathcal{I}BP},t^i_{\mathcal{I}BP}) \\
&\quad - q(x^i_{\mathcal{I}BP},t^i_{\mathcal{I}BP})]^2. \qquad (5)
\end{aligned}
$$

where the loss function MSE is composed of two parts, namely equation residuals MSf and loss function $MSIB$. $\hat{q}(x^i_{\mathcal{I}BP}, t^i_{\mathcal{I}BP})$ is our predicted solution and $q(x^i_{\mathcal{I}BP}, t^i_{\mathcal{I}BP})$ is the true solution of the equation; N_f is the number of sampling points in the whole region, and $N_{\mathcal{I}BP}$ is the number of sampling points in the mixed region. Similarly, we order $N_{\mathcal{I}BP} = N_{\mathcal{I}B}$. In this way, a kind of prior information mix-training model PMTPINNs is generated by adding the sampling points of the sharp area of the equation solution to the training domain in advance.

4 Numerical experiment

In this section, we test the performance of PINNs, MTPINNs, and PMTPINNs by high-order rogue waves of the cmKdV equation [39]. In all cases, without losing generality, using a deep neural networks with 6 hidden layers with 80 neurons in each hidden layer, we define the hyperbolic tangent function (tanh) as the nonlinear activation equation of the model. By default, L-BFGS algorithm [38] is used to train 40,000 steps of the networks, and then Adam optimizer [37] with default learning rate is used to optimize 40,000 steps of the networks. The weights w and deviations b in all models are initialized using the Xavier method [9], without any additional regularization techniques. It is worth noting that we use $N_f = 10,000$, $N_{\mathcal{I}} = 50$ and $N_{\mathcal{B}} = 50$ sample points, respectively, on equation residuals, initial conditions and boundary conditions to test. For consistency and convenience of subsequent comparison of method performance, we use $N_{\mathcal{IB}} = 100$ sample points on the mixed training dataset from Dirichlet boundary conditions and $N_{\mathcal{IBP}} = 100$ sample points on the mixed dataset from sharp regions and Dirichlet boundary conditions.

Consider cmKdV equation with Dirichlet boundary conditions [39]:

$$
\begin{cases}
q_t + 6|q|^2 q_x + q_{xxx} = 0, (x,t) \in \Omega \times T, \\
q(x,t) = \mathcal{I}(x,t), x \in \Omega, t = t_0, \\
q(x,t) = \mathcal{B}(x,t), (x,t) \in \partial\Omega \times T,
\end{cases} \qquad (6)
$$

where $q(x,t)$ is an equation containing variables x and t, which are space variables and time variables, respectively.

4.1 Second-order rogue waves

In this section, we take the second-order rogue waves of cmKdV equation as an example, and implement two sets of numerical experiments according to the different parameters of the rogue waves. The form of the second-order rogue waves of the cmKdV equation [39] is as follows:

$$
q^{[2]} = ce^{ia[x+t(a^2-6c^2)]}\frac{B(x,t)}{C(x,t)}, \qquad (7)
$$

where $B(x,t)$ and $C(x,t)$ are polynomials containing terms a, c, x, t, s_0, s_1, and the specific values are given in Appendix A.

4.1.1 $a = 1.44, c = 1, s_0 = 0, s_1 = 100$

First, when $a = 1.44, c = 1, s_0 = 0, s_1 = 100$, $T \in [-0.6, 0.8]$, $X \in [-8, 8]$, the corresponding cmKdV equation with Dirichlet boundary conditions [39] is as follows:

$$
\begin{cases}
q_t + 6|q|^2 q_x + q_{xxx} = 0, (x,t) \in \Omega \times T, \\
q(x,-0.6) = ce^{ia[x-0.6(a^2-6c^2)]}\frac{B(x,-0.6)}{C(x,-0.6)}, x \in [-8,8], \\
q(-8,t) = ce^{ia[-8+t(a^2-6c^2)]}\frac{B(-8,t)}{C(-8,t)}, t \in [-0.6,0.8], \\
q(8,t) = ce^{ia[8+t(a^2-6c^2)]}\frac{B(8,t)}{C(8,t)}, t \in [-0.6,0.8].
\end{cases} \qquad (8)
$$

First, we use PINNS, MTPINNS and PMTPINNs models, combined with the training conditions and optimization methods described in Sect. 3 to conduct numerical experiments. However, we find that PINNs model cannot simulate second-order rogue waves, so PINNs is not suitable for this. Then we conduct numerical experiments on MTPINNs and find that MTPINNs has a qualitative leap in simulation ability compared with PINNs. Table 1 provides a more intuitive and detailed assessment of MTPINNs' simulated performance.

From Table 1, we can obtain the experimental results of MTPINNs' prediction performance for second-order abnormal waves under all neural network conditions.

Table 1 The relative errors of the two models with respect to the first kind of second-order rogue waves

Numerical experimental model		
Neural network information	PINNs	MTPINNs
40 Neuron|6 hidden layer	2.47e+00	2.2e−02
60 Neuron|6 hidden layer	2.04e+00	1.665e−02
80 Neuron|6 hidden layer	2.47e+00	7.532e−03
40 Neuron|8 hidden layer	3.56e+00	1.87e−02
60 Neuron|8 hidden layer	2.18e+00	1.041e−02
80 Neuron|8 hidden layer	2.47e+00	4.34e−03

Press the Tab key. From Table 1, we find that MTPINNs was superior to PINNs when the number of neurons in the hidden layer and each layer was the same. MTPINNs can not only complete the tasks that PINNs has not completed, but also has a high accuracy rate. In order to better and more clearly observe the performance of MTPINNs, we give the exact solution, predictive solution and absolute error of MTPINNs in Fig. 2, where the black marks in the figure are we selected training points. In addition, in order to observe the iterative optimization process of the optimizer we set up, we also draw the loss function shown in Fig. (3). As can be seen from Fig. 3, the value of the loss function gradually decreases with the increase of the number of iterations. And we believe that with the increase of the number of iterations, the error of the model will be smaller and smaller, and the accuracy will be higher and higher.

In addition, in order to test PMTPINNs, we choose a neural networks with 6 hidden layers and 80 neurons in each layer, and add some training points in the sharp area of the rogue waves (Fig. 4 shows the new sampling area). Through Table 2, we find that PMTPINNs is an order of magnitude more accurate than MTPINNs, which is very meaningful. Based on the results, the concept of model optimization rate is introduced, It meets the following conditions: $1 - \frac{PMTPINNs\ \mathbb{L}_{2\ errors}}{MTPINNs\ \mathbb{L}_{2\ errors}}$. We

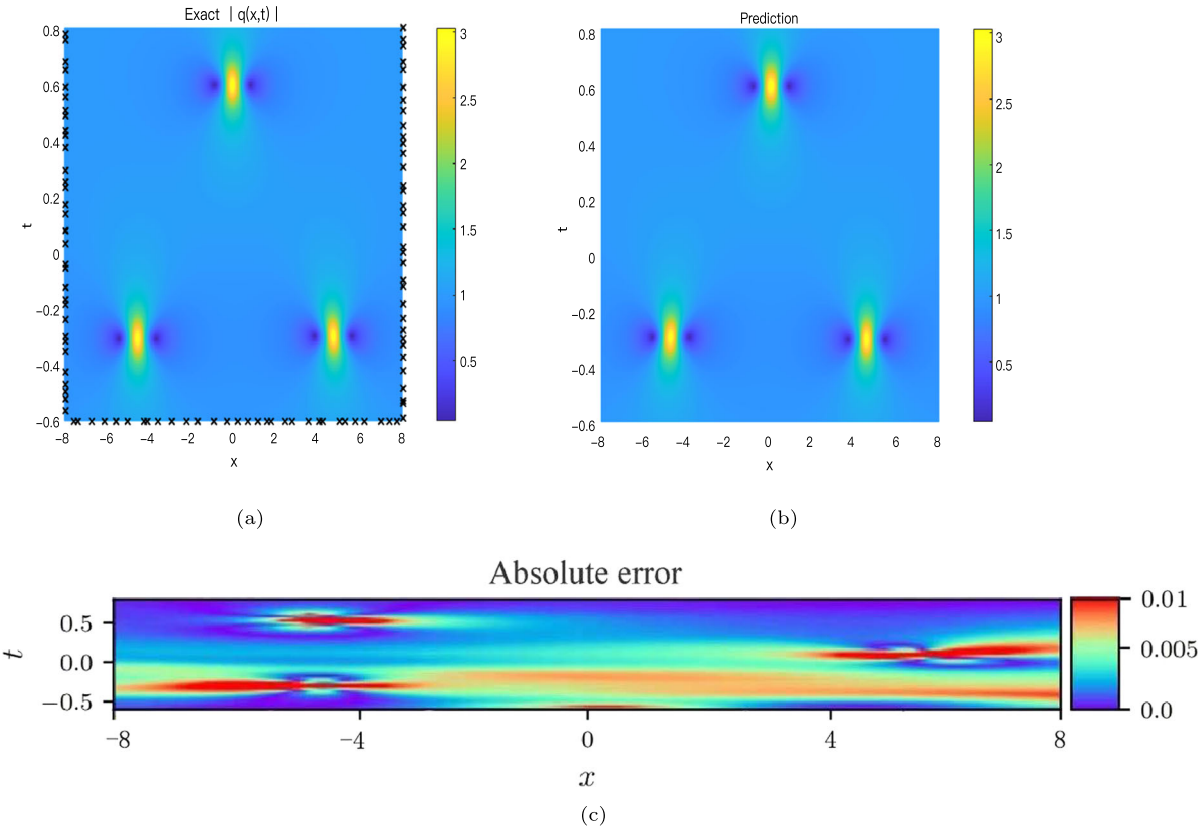

Fig. 2 **a** Exact solution of equation (8), **b** predicted solution of equation (8), **c** absolute error of the second-order rogue waves (8) simulated by MTPINNs (relative error is 7.532e−03)

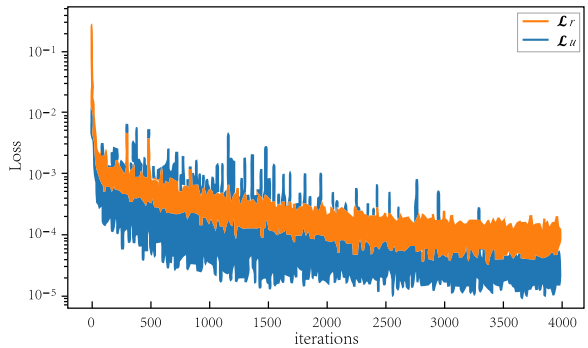

Fig. 3 Training error of the second-order rogue waves simulated by MTPINNs, where $\mathcal{L}_r$ and $\mathcal{L}_u$, respectively, represent the mean value of function residual MSf and the sum of the mean value of the initial error and the mean value of the boundary error

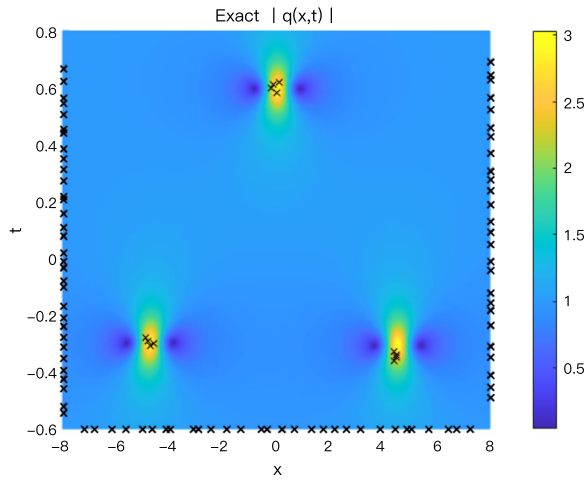

Fig. 4 New sampling area of the second-order rogue waves by PMTPINNs

Table 2 Compared with MTPINNs, the relative errors and optimization rate of PMTPINNs

Numerical experimental model	MTPINNs	PMTPINNs
Neural network information	80 Neuron|6 hidden layer	80 Neuron|6 hidden layer
relative error	7.532e−03	8.415e−04
Model optimization rate	–	88.8%

find the model optimization rate of PMTPINNs is also quite good.

4.1.2 $a = 1.44, c = 1, s_0 = 0, s_1 = 0$

Secondly, when $a = 1.44, c = 1, s_0 = 0, s_1 = 0$, $T \in [-0.5, 0.5]$, $X \in [-5, 5]$, the corresponding cmKdV equation [39] and the three-dimensional diagram in Fig. 5 are as follows:

$$\begin{cases} q_t + 6|q|^2 q_x + q_{xxx} = 0, (x,t) \in \Omega \times T, \\ q(x,-0.5) = ce^{ia[x-0.4(a^2-6c^2)]} \frac{B(x,-0.4)}{C(x,-0.4)}, x \in [-5,5], \\ q(-5,t) = ce^{ia[-3+t(a^2-6c^2)]} \frac{B(-3,t)}{C(-3,t)}, t \in [-0.5,0.5], \\ q(5,t) = ce^{ia[3+t(a^2-6c^2)]} \frac{B(3,t)}{C(3,t)}, t \in [-0.5,0.5]. \end{cases} \quad (9)$$

We carried out the same experimental method as above and also found that PINNs could not simulate the second-order rogue waves, and the accuracy of MTPINNs was still excellent. The experimental results of the prediction performance of MTPINNs for this second-order rogue waves under all neural networks conditions are shown in Table 3.

4.2 Third-order rogue waves

In this section, we will simulate the third-order rogue waves of the cmKdV equation. The form of the third-order rogue waves of the cmKdV equation [39] is as follows:

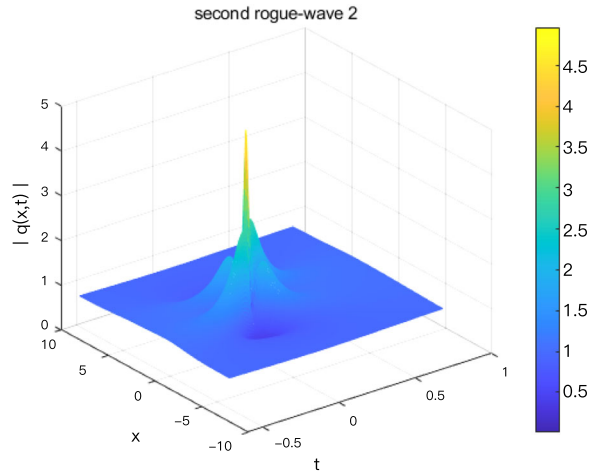

Fig. 5 Three-dimensional diagram of Eq. (9)

Table 3 The relative errors of the two models with respect to the second kind of second-order rogue waves

Numerical experimental model		
Neural network information	PINNs	MTPINNs
40 Neuron|6 hidden layer	1.43e+00	3.885e−02
60 Neuron|6 hidden layer	1.05e+00	1.8e−02
80 Neuron|6 hidden layer	1.43e+00	1.2e−02
40 Neuron|8 hidden layer	1.67e+00	1.743e−02
60 Neuron|8 hidden layer	1.09e+00	1.078e−02
80 Neuron|8 hidden layer	1.43e+00	8.542e−03

$$q^{[3]} = \frac{L_1(x,t)}{L_2(x,t)} e^{i[\frac{3}{2}x - \frac{45}{8}t]}, \tag{10}$$

where $L_1(x,t)$ and $L_2(x,t)$ are polynomials containing terms a, c, x, t, s_0, s_1, and the specific values are given in Appendix B.

Due to the limited data, we only implement numerical experiments on third-order rogue waves in this case. When $a = 1.5, c = 1, s_0 = 0, s_1 = 0, s_2 = 0$, $T \in [-1.0, 1.5]$, $X \in [-10, 10]$, the corresponding cmKdV equation with Dirichlet boundary conditions and the three-dimensional diagram of equations drawn (Fig. 6) are as follows:

$$\begin{cases} q_t + 6|q|^2 q_x + q_{xxx} = 0, (x,t) \in \Omega \times T, \\ q(x,-1.0) = \frac{L_1(x,-0.4)}{L_2(x,-0.4)} e^{i[\frac{3}{2}x + \frac{45}{8} \times \frac{2}{5}]}, x \in [-10, 10], \\ q(-10,t) = \frac{L_1(-6,t)}{L_2(-6,t)} e^{i[-9 - \frac{45}{8}t]}, t \in [-1, 1.5], \\ q(10,t) = \frac{L_1(6,t)}{L_2(6,t)} e^{i[9 - \frac{45}{8}t]}, t \in [-1, 1.5] \end{cases} \tag{11}$$

We still use the three models in Sect. 2, combined with the training conditions and optimization methods described in Sect. 3, to implement numerical experiments. The relative error of PINNs is still very large, and the third-order rogue waves of the equation cannot be simulated. Then we proceeded numerical experiments on the MTPINNs, we found that compared with the PINNs, MTPINNs still has a good accuracy. Table 4 provides a more intuitive and detailed evaluation of the prediction performance of the MTPINNs.

From Table 4, when the number of neurons in the hidden layer and each layer is the same, we can better confirm that MTPINNs is superior to PINNs. MTPINNs not only complete the tasks that PINNs have not completed, but also have high accuracy. From Table 4, we can obtain the experimental results of the prediction performance of MTPINNs on the third-order rogue waves under all neural networks conditions. In order to better and more clearly observe the performance of the model, we still present the exact solution, predicted solution and absolute error of the equation in Fig. 7.

In addition, in order to test PMTPINNs model, we still select the neural networks with 6 hidden layers and 80 neurons in each layer, and add some training points (Fig. 8) near the sharp area of the rogue waves in training area. Through numerical experiments, we find that PMTPINNs have higher accuracy than MTPINNs. Through Table 5, it is found that the optimization rate of the model relative to MTPINNs is still good, which is very meaningful.

In conclusion, MTPINNs are better than PINNs in the simulation of second-order and third-order rogue waves of cmKdV equation, and PMTPINNs again improve the accuracy based on MTPINNs.

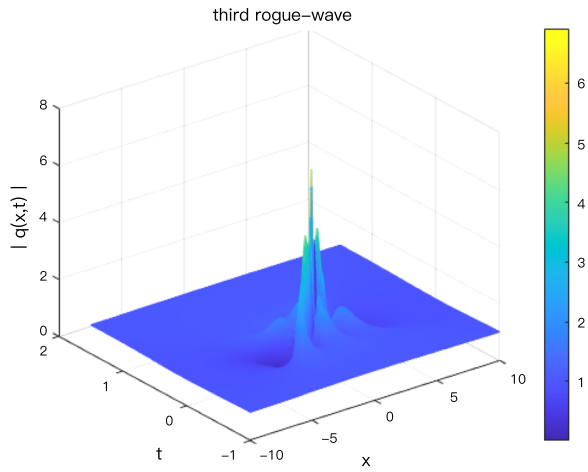

Fig. 6 Three-dimensional diagram of Eq. (11)

Table 4 Relative errors of the two models

Numerical experimental model		
Neural network information	PINNs	MTPINNs
40 Neuron|6 hidden layer	1.672e+00	1.317e−03
60 Neuron|6 hidden layer	1.45e+00	1.15e−03
80 Neuron|6 hidden layer	1.09e+00	9.802e−04
40 Neuron|8 hidden layer	2.31e+00	2.221e−03
60 Neuron|8 hidden layer	1.672e+00	7.234e−04
80 Neuron|8 hidden layer	1.843e+00	6.407e−04

4.3 The inverse problem of cmKdV Equation

In order to explore the unknown parameters $[\lambda_1, \lambda_2]$ of cmKdV equation, we will use data-driven discovery method to discover NPDEs from MTPINNs deep learning framework. The specific form of the equation [39] is as follows:

$$q_t + \lambda_1 |q|^2 q_x + \lambda_2 q_{xxx} = 0, \quad (x,t) \in \Omega \times T. \tag{12}$$

The potential solution $\hat{q}(x,t) = \hat{u}(x,t) + \mathrm{i}\hat{v}(x,t)$, $[\hat{u}(x,t), \hat{v}(x,t)]$ and are the real part and imaginary part, respectively. Then, we make $F(x,t) = \mathrm{i}F_u(x,t) + F_v(x,t)$, we approximate the potential solution by using the reduced value of $F(x,t)$ method. Here, $F(x,t)$, $F_u(x,t)$ and $F_v(x,t)$ satisfy

$$\begin{aligned} F(x,t) &= q_t + \lambda_1 |q|^2 q_x + \lambda_2 q_{xxx}, \\ F_u(x,t) &= v_t + \lambda_1 u^2 v_x + \lambda_1 v^2 v_x + \lambda_2 v_{xxx}, \\ F_v(x,t) &= u_t + \lambda_1 u^2 u_x + \lambda_1 v^2 u_x + \lambda_2 u_{xxx}. \end{aligned} \tag{13}$$

The unknown parameters $[\lambda_1, \lambda_2]$ are trained by using input data sets $[u(x,t), v(x,t)]$ from initial values $\mathcal{I}(x,t) = u_0(x,t_0) + v_0(x,t_0)$ and boundary conditions $\mathcal{B}(x,t) = u_{\mathcal{B}}(x,t) + v_{\mathcal{B}}(x,t)$, and the undetected solution $\hat{q}(x,t) = \hat{u}(x,t) + \mathrm{i}\hat{v}(x,t)$ is approximated by minimizing the loss function MSE:

$$\begin{aligned} MSE = &\frac{1}{N_R} \sum_{i=1}^{N_R} \left(\left| f_u(x_R^i, t_R^i) \right|^2 + \left| f_v(x_R^i, t_R^i) \right|^2 \right) \\ &+ \frac{1}{N_S} \sum_{i=1}^{N_S} \left(\left| \hat{u}(x_S^j, t_S^j) - u(x_S^j, t_S^j) \right|^2 \right. \\ &\left. + \left| \hat{v}(x_S^j, t_S^j) - v(x_S^j, t_S^j) \right|^2 \right), \end{aligned} \tag{14}$$

here $N_{\mathcal{R}}$ has been mentioned above, $N_S = N_{\mathcal{I}} + N_{\mathcal{B}} = N_{\mathcal{IB}} = 100$. For simplicity, we will directly use the neural network parameters used in Sect. 2 for training. It is worth mentioning that in addition to the clean data, we also add 2 and 5% noise data to simulate the second-order and third-order rogue waves of the cmKdV equation, the final numerical results shown in Tables 6 and 7. As can be seen from the numerical experimental results in Tables 6 and 7, MTPINNs not only has a very superior ability in solving inverse problems with clean data; when adding noise data, MTPINNs still has a superior simulation ability in solving inverse problems, which indicates that our model has good robustness and undoubtedly confirms the great advantages of our model again.

5 Conclusion

In this work, based on PINNs, we combine initial condition region $\mathcal{I}$ and boundary condition region $\mathcal{B}$ into a

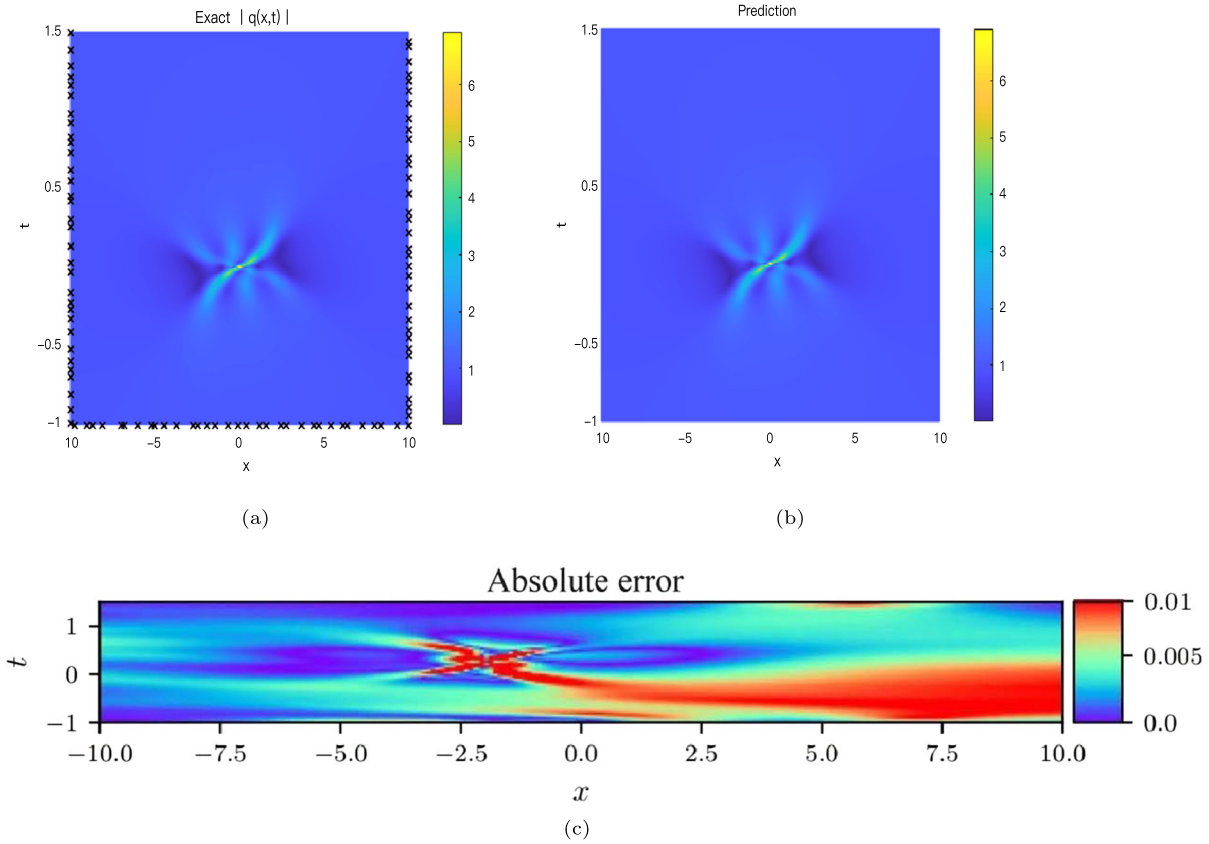

Fig. 7 **a** Exact solution of equation (11), **b** predicted solution of equation (11), **c** absolute error of the third-order rogue waves (11) simulated by MTPINNs (relative error is 9.802e−04)

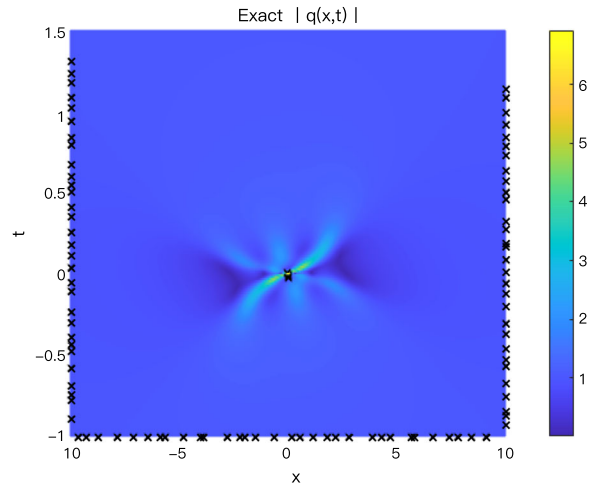

Fig. 8 New sampling area of third-order rogue waves by PMTPINNs

Table 5 Compared with MTPINNs, the relative error and optimization rate of PMTPINNs

Numerical experimental model	MTPINNs	PMTPINNs
Neural network information	80 Neuron|6 hidden layer	80 Neuron|6 hidden layer
Relative error	9.802e−04	4.146e−04
Model optimization rate	–	57.7%

Table 6 Second-order rogue waves: the correct cmKdV equation and the identified one obtained by learning λ_1 and λ_2 and their relative errors

NPDEs	Item	
	The cmKdV equation	Relative error $[\lambda_1, \lambda_2]$ (%)
Correct	$q_t + 6.0\|q\|^2 q_x + 1.0 q_{xxx} = 0$	[0, 0]
MTPINNs: identified (clean data)	$q_t + 6.00248\|q\|^2 q_x + 1.00229 q_{xxx} = 0$	[0.041318, 0.228977]
MTPINNs: identified (2% noise)	$q_t + 6.03350\|q\|^2 q_x + 0.95910 q_{xxx} = 0$	[0.558376, 4.090452]
MTPINNs: identified (5% noise)	$q_t + 6.03552\|q\|^2 q_x + 0.95584 q_{xxx} = 0$	[0.591969, 4.416352]

Table 7 Third-order rogue waves: the correct cmKdV equation and the identified one obtained by learning λ_1 and λ_2, and their relative errors

NPDEs	Item	
	The cmKdV equation	Relative error $[\lambda_1, \lambda_2]$ (%)
Correct	$q_t + 6.0\|q\|^2 q_x + 1.0 q_{xxx} = 0$	[0, 0]
MTPINNs: identified (clean data)	$q_t + 6.00915\|q\|^2 q_x + 1.00172 q_{xxx} = 0$	[0.152508, 0.172484]
MTPINNs: identified (2% noise)	$q_t + 5.90552\|q\|^2 q_x + 1.02934 q_{xxx} = 0$	[1.574747, 2.934206]
MTPINNs: identified (5% noise)	$q_t + 5.88343\|q\|^2 q_x + 1.10192 q_{xxx} = 0$	[1.942833, 10.191547]

new training region $\mathcal{IB}$ and propose MTPINNs. Then, we use an adaptive search algorithm to find sample points in the sharp region of rogue waves in the cmKdV equation. These new sample points are added into the training area $\mathcal{IB}$ as priors to form a new training area $\mathcal{IBP}$, and PMTPINNs is proposed.

Taking the second-order and third-order rogue waves of cmKdV equation as examples, the numerical experiments show that the original PINNs cannot simulate the high-order rogue waves of cmKdV equation at all. However, our proposed models not only accomplish the task that PINNs cannot, but also significantly improve the simulation capability. And the prediction accuracy improved by three or four orders of magnitude. We verify the inverse problem of cmKdV equation with the proposed model; the numerical results in Tables 6 and 7 show that proposed models have good performance. In addition, we found that after adding noise data, our proposed models still have good simulation ability, which also proves that these models have good robustness.

Acknowledgements This work is supported by National Natural Science Foundation of China under Grant Nos. 12175111 and 12235007 and K.C. Wong Magna Fund in Ningbo University.

Funding The funding was provided by National Natural Science Foundation of China under Grant Nos. 12175111 and 12235007 and K.C. Wong Magna Fund in Ningbo University.

Data Availability All data generated or analyzed during this study are included in this article.

Declarations

Conflict of interest The authors declare that they have no conflict of interests.

Appendix A: ($B(x,t)$ and $C(x,t)$ of second-order rogue waves)

$$B(x,t) = 2239488a^4c^{14}t^6 + 46656a^{12}c^6t^6$$

$$
\begin{aligned}
&+2985984c^{18}t^6\\
&+559872a^8c^{10}t^6+746496a^6c^{10}t^5x\\
&+93312a^{10}c^6t^5x-1492992a^4c^{12}t^5x\\
&-186624a^8c^8t^5x+1492992a^2c^{14}t^5x\\
&-2985984c^{16}t^5x+1244160c^{14}t^4x^2\\
&-248832a^6c^8t^4x^2-995328a^2c^{12}t^4x^2\\
&+622080a^4c^{10}t^4x^2+77760a^8c^6t^4x^2\\
&-276480c^{12}t^3x^3+34560a^6c^6t^3x^3\\
&-124416a^4c^8t^3x^3+248832a^2c^{10}t^3x^3\\
&+34560c^{10}t^2x^4+8640a^4c^6t^2x^4\\
&-27648a^2c^8t^2x^4+1152a^2c^6tx^5\\
&-2304c^8tx^5+64c^6x^6-331776c^6a^6t^4\\
&-518400c^{12}t^4+155520c^8a^4t^4\\
&-11664c^4a^8t^4\\
&+290304c^{10}t^3x-15552c^4a^6t^3x\\
&+103680c^8a^2t^3x-176256c^6a^4t^3x\\
&-58752c^8t^2x^2\\
&-6912c^6a^2t^2x^2-7776c^4a^4t^2x^2\\
&+4992c^6tx^3-1728c^4a^2tx^3-144c^4x^4\\
&+124416ac^{10}s_1t^3+31104a^5c^6s_1t^3\\
&-165888a^3c^8s_1t^3+20736a^3c^6s_1t^2x\\
&-41472ac^8s_1t^2x\\
&+3456ac^6s_1tx^2-18000c^6t^2\\
&-1620c^2a^4t^2-1080c^2a^2tx\\
&+5616c^4tx-180c^2x^2\\
&+864c^4as_1t+144c^4s_1^2\\
&+45+i(2985984ac^{14}t^5+1492992a^5c^{10}t^5\\
&+186624a^9c^6t^5-497664a^5c^8t^4x\\
&+995328a^3c^{10}t^4x-1990656ac^{12}t^4x\\
&+248832a^7c^6t^4x-331776a^3c^8t^3x^2\\
&+124416a^5c^6t^3x^2+497664ac^{10}t^3x^2\\
&-55296ac^8t^2x^3+27648a^3c^6t^2x^3\\
&+2304ac^6tx^4+207360c^8at^331104c^4a^5t^3\\
&-13824c^6at^2x-20736c^4a^3t^2x\\
&-3456c^4atx^2+20736c^8s_1t^2\\
&-41472a^2c^6s_1t^2+5184a^4c^4s_1t^2\\
&+3456a^2c^4s_1tx-6912c^6s_1tx+576c^4s_1x^2\\
&-2160ac^2t+144s_1c^2)
\end{aligned}
$$

and

$$
\begin{aligned}
C(x,t)=\;&2239488a^4c^{14}t^6+2985984c^{18}t^6\\
&+46656a^{12}c^6t^6\\
&+559872a^8c^{10}t^6+746496a^6c^{10}t^5x\\
&-1492992a^4c^{12}t^5x-186624a^8c^8t^5x\\
&-2985984c^{16}t^5x+93312a^{10}c^6t^5x\\
&+1492992a^2c^{14}t^5x+1244160c^{14}t^4x^2\\
&-995328a^2c^{12}t^4x^2+622080a^4c^{10}t^4x^2\\
&+77760a^8c^6t^4x^2-248832a^6c^8t^4x^2\\
&+248832a^2c^{10}t^3x^3+34560a^6c^6t^3x^3\\
&-124416a^4c^8t^3x^3-276480c^{12}t^3x^3\\
&-27648a^2c^8t^2x^4+8640a^4c^6t^2x^4\\
&+34560c^{10}t^2x^4-2304c^8tx^5\\
&+1152a^2c^6tx^5+64c^6x^6\\
&+995328c^{10}a^2t^4+279936c^8a^4t^4\\
&-82944c^6a^6t^4+3888c^4a^8t^4\\
&-269568c^{12}t^4+124416c^{10}t^3x\\
&+5184c^4a^6t^3x-51840c^6a^4t^3x\\
&-145152c^8a^2t^3x\\
&-17280c^8t^2x^2-6912c^6a^2t^2x^2\\
&+2592c^4a^4t^2x^2+384c^6tx^3+576c^4a^2tx^3\\
&+48c^4x^4-165888a^3c^8s_1t^3\\
&+124416ac^{10}s_1t^3+31104a^5c^6s_1t^3\\
&+20736a^3c^6s_1t^2x\\
&-41472ac^8s_1t^2x\\
&+3456ac^6s_1tx^2+20016c^6t^2\\
&+972c^2a^4t^2+6912c^4a^2t^2\\
&+648c^2a^2tx-2448c^4tx\\
&+108c^2x^2-2592c^4as_1t+144c^4s_1^2+9.
\end{aligned}
$$

Appendix B: ($L1(x,t)$ and $L2(x,t)$ of third-order rogue waves)

$$
\begin{aligned}
L_1(x,t)=\;&-4939273445868140625t^{12}\\
&-545023276785450000t^{11}x\\
&-38840739265170000t^{10}x^2\\
&-34025850934560000t^9x^3\\
&-12374529519456000t^8x^4\\
&-841946352721920t^7x^5\\
&-204871837925376t^6x^6\\
&-10322713903104t^5x^7\\
&-1860148592640t^4x^8\\
&-62710087680t^3x^9-8776581120t^2x^{10}\\
&-150994944tx^{11}-16777216x^{12}\\
&+23002372922807250000t^{10}
\end{aligned}
$$

$$
\begin{aligned}
&+5000802910383600000t^9x\\
&+178207653254544000t^8x^2\\
&+20160748631347200t^7x^3\\
&+4231975119851520t^6x^4\\
&+253682249269248t^5x^5\\
&+40317552230400t^4x^6\\
&+880347709440t^3x^7\\
&+141203865600t^2x^8-1447034880tx^9\\
&+75497472x^{10}+257120426548112400t^8\\
&-88521031030049280t^7x\\
&-1841385225323520t^6x^2\\
&-617799343104000t^5x^3\\
&-88747774156800t^4x^4\\
&-8252622766080t^3x^5\\
&-200693514240t^2x^6\\
&+1415577600tx^7+235929600x^8\\
&+12647412412496640t^6\\
&+42148769126400t^5x\\
&-2996161228800t^4x^2\\
&-15902996889600t^3x^3\\
&+313860096000t^2x^4\\
&-8139571200tx^5+707788800x^6\\
&-149676507590400t^4\\
&+2148738969600t^3x\\
&-1622998425600t^2x^2+31186944000t^3x\\
&-928972800x^4-95215564800t^2\\
&+21598617600tx\\
&-464486400x^2+58060800\\
&+i(-65402793214254000000t^{11}\\
&-601404995073600000t^{10}x\\
&-423057306879360000t^9x^2\\
&-29901007761408000t^8x^3\\
&-10648770814771200t^7x^4\\
&-552411244265472t^6x^5\\
&-130559642173440t^5x^6\\
&-4494741995520t^4x^7\\
&-779700142080t^3x^8-13589544960t^2x^9\\
&-1811939328tx^{10}\\
&-303419904173280000t^9\\
&+263517097430016000t^8x\\
&+29562711599923200t^7x^2\\
&+7820253397647360t^6x^3\\
&+799710056939520t^5x^4\\
&+58500443013120t^4x^5\\
&+5243865661440t^3x^6\\
&+40768634880t^2x^7+6794772480tx^8\\
&+10822374648023040t^7\\
&-158051569987584000t^6x\\
&-36629818195968 0t^5x^2\\
&+15745929707520 0t^4x^3\\
&-6736379904000t^3x^4-249707888640t^2x^5\\
&+16986931200tx^6\\
&+2321279745269760t^5\\
&-164322282700800t^4x\\
&+7849554739200t^3x^2\\
&-700710912000t^2x^3+38220595200tx^4\\
&+4992863846400t^3\\
&-967458816000t^2x\\
&-33443020800tx^2-8360755200t)
\end{aligned}
$$

$$
\begin{aligned}
L_2(x,t) = {}& 49392734458681406 25t^{12}\\
&+545023276785450000t^{11}x\\
&+3884073926517000 00t^{10}x^2\\
&+340258509345600 00t^9x^3\\
&+123745295194560 00t^8x^4\\
&+8419463527219 20t^7x^5\\
&+2048718379253 76t^6x^6\\
&+1032271390310 4t^5x^7\\
&+186014859264 0t^4x^8\\
&+62710087680t^3x^9+8776581120t^2x^{10}\\
&+150994944tx^{11}+16777216x^{12}\\
&+1671541529351175000t^{10}\\
&-201221180459640000t^9x\\
&+295833742149600 00t^8x^2\\
&-864620698429440 0t^7x^3\\
&-20587214555136 0t^6x^4\\
&-1021850783907 84t^5x^5\\
&-5361951375360t^4x^6-141416202240t^3x^7\\
&-16349921280t^2x^8+2202009600tx^9\\
&+25165824x^{10}+2141921095479036 00t^8\\
&+434636773579008 0t^7x\\
&+37831543469107 20t^6x^2\\
&-10756355519692 80t^5x^3\\
&+4094217031680 0t^4x^4\\
&+1840480911360t^3x^5+84333035520t^2x^6\\
&+849346560tx^7+141557760x^8
\end{aligned}
$$

$$
\begin{aligned}
&+3537485306138880t^6\\
&+789853361080320t^5x\\
&+130057245388800t^4x^2\\
&-11222478028800t^3x^3\\
&+402245222400t^2x^4\\
&-4034396160tx^5+613416960x^6\\
&+90779142700800t^4-6216180019200t^3x\\
&-269186457600t^2x^2+11988172800tx^3\\
&+221184000x^4+213178521600t^2\\
&-5009817600tx+199065600x^2\\
&+8294400
\end{aligned}
$$

References

1. Gustafsson, T., Rajagopal, K.R., Stenberg, R., Videman, J.: Nonlinear Reynolds equation for hydrodynamic lubrication. Appl. Math. Model. **39**, 5299–5309 (2015)
2. Kaup, D.J., Newell, A.C.: An exact solution for a derivative nonlinear Schrödinger equation. J. Math. Phys. **19**, 798–801 (1978)
3. Polyanin, A.D., Zhurov, A.I.: The functional constraints method: application to non-linear delay reaction?diffusion equations with varying transfer coefficients. Int. J. Nonlinear Mech. **67**, 267–277 (2014)
4. Parkins, A.S., Walls, D.F.: The physics of trapped dilute-gas Bose-Einstein condensates. Phys. Rep. **303**, 1–80 (1998)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
6. Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: a generative model for raw audio. In: 9th ISCA Speech Syn Thesis Workshop, pp. 125–135 (2016)
7. Heaton, J., Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. Genet. Program Evol. Mach. **19**, 305–307 (2018)
8. Alipanahi, B., Delong, A., Weirauch, M.T., Frey, B.J.: Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nat. Biotechnol. **33**, 831–838 (2015)
9. Raissi, M., Karniadakis, G.E.: Physics informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **357**, 125–141 (2018)
10. Weinan, E., Han, J.Q., Jentzen, A.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Commun. Math. Stat. **5**, 349–380 (2017)
11. Sirignano, J., Spiliopoulos, K.: A deep learning algorithm for solving partial differential equations. J. Comput. Phys. **375**, 1339–1364 (2018)
12. Moseley, B., Markham, A., Nissen-Meyer, T.: Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations. arXiv:2107.07871 (2021)
13. Bihlo, A., Popovych, R.O.: Physics-informed neural networks for the shallow-water equations on the sphere. J. Comput. Phys. **456**, 111024 (2022)
14. Luo, H.T., Wang, L., Zhang, Y.B., Lu, G., Su, J.J., Zhao, Y.C.: Data-driven solutions and parameter discovery of the Sasa-Satsuma equation via the physics-informed neural networks method. Physica D **440**, 133489 (2022)
15. Li, J.H., Li, B.: Solving forward and inverse problems of the nonlinear Schrödinger equation with the generalized $\mathcal{PT}$-symmetric Scarf-II potential via PINN deep learning. Commun. Theor. Phys. **73**, 125001 (2021)
16. Pu, J.C., Chen, Y.: Data-driven vector localized waves and parameters discovery for Manakov system using deep learning approach. Chaos, Solitons Fractals **160**, 112182 (2022)
17. Jagtap, A.D., Mao, Z.P., Adams, N., Karniadakis, G.E.: Physics-informed neural networks for inverse problems in supersonic flows. J. Comput. Phys. **466**, 111402 (2022)
18. Matteya, R., Ghosha, S.: A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations. Comput. Methods Appl. Mech. Eng. **390**, 114474 (2022)
19. Rezaei, S., Harandi, A., Moeineddin, A., Xu, B.X., Reese, S.: A mixed formulation for physics-informed neural networks as a potential solver for engineering problems in heterogeneous domains: Comparison with finite element method. Comput. Methods Appl. Mech. Eng. **401**, 115616 (2022)
20. Li, J., Chen, Y.: A physics-constrained deep residual net work for solving the sine-Gordon equation. Commun. Theor. Phys. **73**, 015001 (2021)
21. Pu, J.C., Li, J., Chen, Y.: Soliton, breather and rogue wave solutions for solving the nonlinear Schrödinger equation using a deep learning method with physical constraints. Chin. Phys. B **30**, 060202 (2021)
22. Pu, J.C., Li, J., Chen, Y.: Solving localized wave solutions of the derivative nonlinear Schrödinger equation using an improved PINN method. Nonlinear Dyn. **105**, 1723–1739 (2021)
23. Pu, J.C., Chen, Y.: PINN deep learning for the Chen–Lee–Liu equation: rogue wave on the periodic back ground. Chaos, Solitons Fractals **160**, 112182 (2022)
24. Lin, S.N., Chen, Y.: A two-stage physics-informed neural network method based on conserved quantities and applications in localized wave solutions. J. Comput. Phys. **41**, 898–909 (2022)
25. Ling, L.M., Mo, Y.F., Zeng, D.L.: Data-driven vector soliton solutions of coupled nonlinear Schrödinger equation using a deep learning algorithm. Phys. Lett. A **421**, 127739 (2022)
26. Wang, L., Yan, Z.Y.: Data-driven rogue waves and parameter discovery in the defocusing nonlinear Schrödinger equation with a potential using the PINN deep learning. Phys. Lett. A **404**, 127408 (2021)
27. Wang, L., Yan, Z.Y.: Data-driven peakon and periodic peakon travelling wave solutions of some nonlinear dispersive equations via deep learning. Phys. Lett. A **450**, 128373 (2022)
28. Fang, Y., Wu, G.Z., Wang, Y.Y., et al.: Data-driven femtosecond optical soliton excitations and parameters discovery of the high-order NLSE using the PINN. Nonlinear Dyn. **105**, 603–616 (2021)

29. Zhou, Z.J., Yan, Z.Y.: Solving forward and inverse problems of the logarithmic nonlinear Schrödinger equation with $\mathcal{PT}$-symmetric harmonic potential via deep learning. Phys. Lett. A **387**, 127010 (2021)
30. Li, J.H., Chen, J.C., Li, B.: Gradient-optimized physics-informed neural networks (GOPINNs): a deep learning method for solving the complex modified KdV equation. Nonlinear Dyn. **107**, 781–792 (2022)
31. Tian, S.F., Li, B.: Gradient-optimized physics-informed neural networks (GOPINNs): a deep learning method for solving complex nonlinear problems. Acta Phys. Sin. https://doi.org/10.7498/aps.72.20222381
32. Li, J.H., Li, B.: Mix-training physics-informed neural networks for the rogue waves of nonlinear Schrödinger equation. Chaos, Solitons Fractals **164**, 112712 (2022)
33. Wen, X.K., Wu, G.Z., Liu, W., Dai, C.Q.: Dynamics of diverse data-driven solitons for the three component coupled nonlinear Schrödinger model by the MPS-PINN method. Nonlinear Dyn. **109**, 3041–3050 (2022)
34. Fang, Y., Wu, G.Z., Dai, C.Q.: Data-driven soliton solutions and model parameters of nonlinear wave models via the conservation-law constrained neural network method. Chaos, Solitons Fractals **158**, 112118 (2022)
35. Wu, G.Z., Fang, Y., Dai, C.Q.: Predicting the dynamic process and model parameters of the vector optical solitons in birefringent fibers via the modified PINN. Chaos, Solitons Fractals **152**, 111393 (2022)
36. Li, J., Cheng, J.H., Shi, J.Y., Huang, F.: Brief introduction of back propagation (BP) Neural Network algorithm and its improvement. Adv. CSIE **2**, 553–558 (2012)
37. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. arXiv:1412.6980 (2014)
38. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Program. **45**, 503–528 (1989)
39. He, J.S., Wang, L.H., Li, L.J., Porsezian, K., Erdelyi, R.: Few-cycle optical rogue waves: complex modified Korteweg–de Vries equation. Phys. Rev. E **89**, 062917 (2014)