Hindawi
Advances in Mathematical Physics
Volume 2021, Article ID 5569645, 11 pages
https://doi.org/10.1155/2021/5569645

Research Article

Physics Informed by Deep Learning: Numerical Solutions of Modified Korteweg-de Vries Equation

Yuexing Bai,[1] **Temuer Chaolu**,[2] **and Sudao Bilige**[3]

[1]*School of Information Engineering, Shanghai Maritime University, Shanghai 200135, China*
[2]*College of Arts and Sciences, Shanghai Maritime University, Shanghai 200135, China*
[3]*Department of Mathematics, Inner Mongolia University of Technology, Hohhot 010051, China*

Correspondence should be addressed to Temuer Chaolu; tmchaolu@shmtu.edu.cn

Received 27 February 2021; Revised 23 April 2021; Accepted 13 May 2021; Published 29 May 2021

Academic Editor: Emilio Turco

In this paper, with the aid of symbolic computation system Python and based on the deep neural network (DNN), automatic differentiation (AD), and limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization algorithms, we discussed the modified Korteweg-de Vries (mkdv) equation to obtain numerical solutions. From the predicted solution and the expected solution, the resulting prediction error reaches 10^{-6}. The method that we used in this paper had demonstrated the powerful mathematical and physical ability of deep learning to flexibly simulate the physical dynamic state represented by differential equations and also opens the way for us to understand more physical phenomena later.

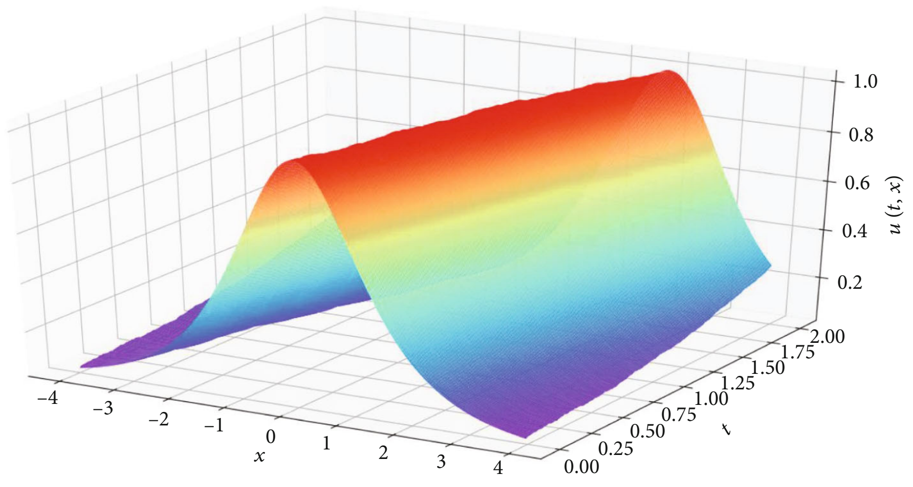
1. Introduction

In recent years, nonlinear phenomena have been widely used in fields such as mathematics, physics, chemistry, biology, finance, and engineering technology. Because a large number of mathematical models of scientific and engineering problems are reduced to the problem for determining solutions of ordinary differential equations (ODEs) and partial differential equations (PDEs) and the problems are complex and the amount of calculation is huge, except for a few special types of differential equations that can be solved by analytical methods, the analytical expressions to be obtained are extremely difficult in most cases. Therefore, the research on the numerical methods for PDE has become a popular mainstream direction. Numerical solutions have attracted the attention of scientific researchers, and it is also a large-scale scientific and engineering calculation.

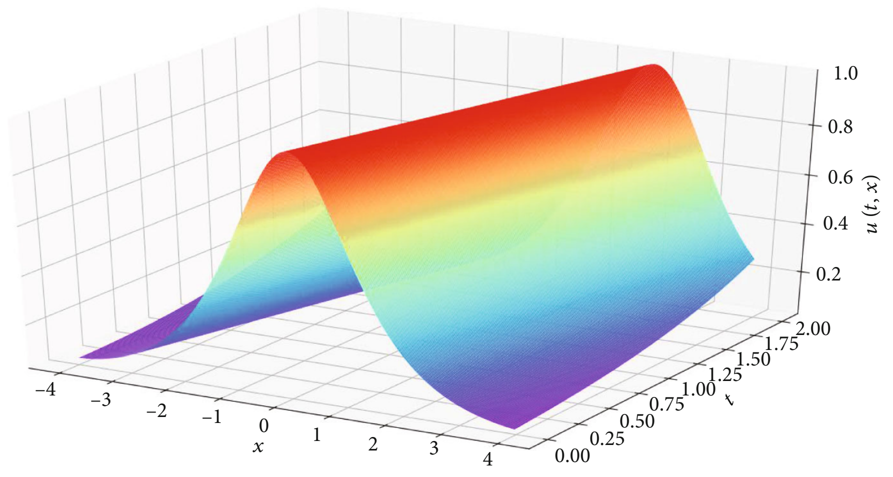
The numerical method of PDE is based on whether the regular grid method and the gridless method are used when discretizing. Due to the difficulties in the structure of the numerical format and meshing, it is subject to many restrictions in practice. In obtaining high-precision and high-resolution solutions, not experienced computational mathematicians will have difficulty for the reason that the structure of the numerical format is very complicated.

Artificial neural networks (ANN) which are simplified models of the biological nervous system represent a technology that has various applications in the area of mathematical modeling, text recognition, voice recognition, learning and memory, pattern recognition, signal processing, automatic control, signal processing, decision-making assistance and time-series analysis, etc. [1]. ANN has been applied to solve ordinary differential equations and partial differential equations as early as more than 20 years ago. As we all know, solving differential equations by neural networks can be regarded as a mesh-free numerical method. Due to the importance of differential equations, many methods have been developed in the literature for solving them [2]. Rosenblatt introduced the first model of supervised learning based on a single-layer neural network with a single neuron [3]. Mcfall studied boundary value problems with arbitrary irregular boundaries by an artificial neural network method in 2006 [4]. Mall and Chakraverty solved ordinary differential equations with the application of the Legendre neural network in 2016 [5].

However, due to the limitation of computing methods and computing resources at that time, this technology has



(a)

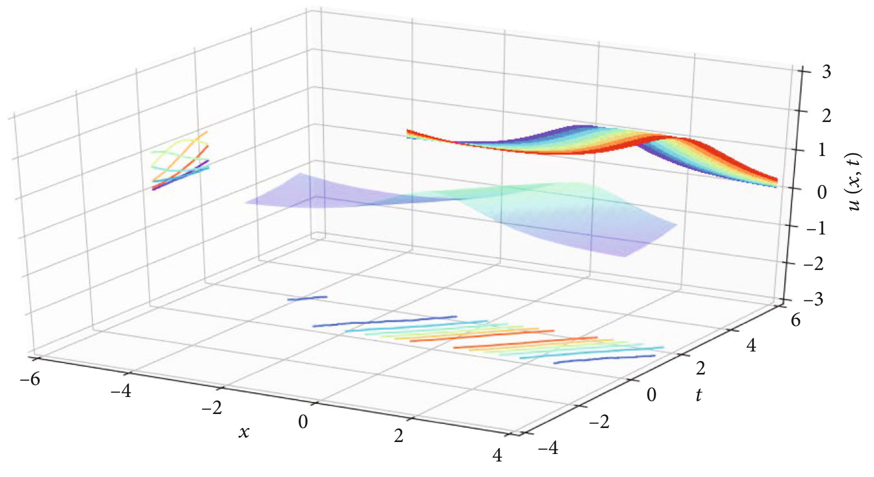


(b)

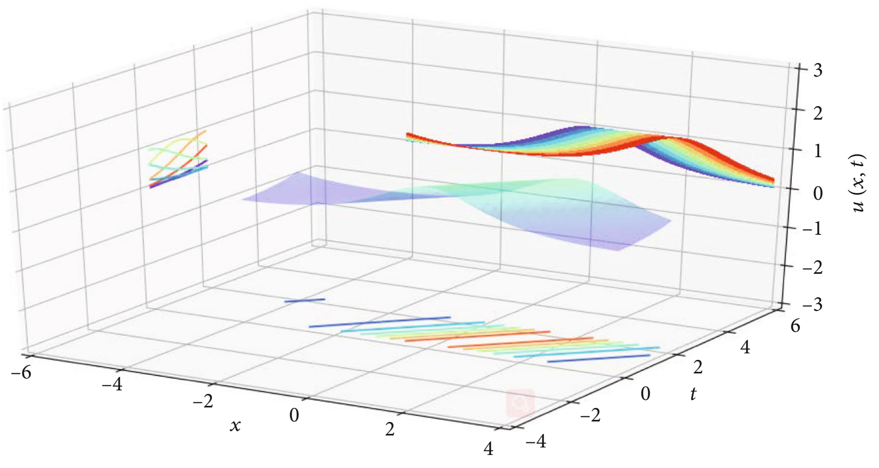
Figure 1: The three-dimensional diagram of the exact solutions (a) and predicted solutions (b) of the mkdv equation with initial state $u(x, 0) = 2\exp(-x)/(\exp(-2x) + 1)$.

not received enough attention. With the development of deep learning in recent years, Professor Karniadakis from the Department of Applied Mathematics at Brown University and his collaborators reexamined the technology and developed a set of deep learning algorithm frameworks based on the original. It was named "physics-informed neural networks (PINN)" and was first used to solve forward and inverse problems of partial differential equations. This has also triggered a lot of follow-up research work and has gradually become a research hotspot in the emerging interdisciplinary field of Scientific Machine Learning (SCIML). From the point of view of function approximation theory in mathematics, the neural network can be regarded as a general nonlinear function approximator, and the modeling process of partial differential equations is also looking for nonlinear functions satisfying constraint conditions, which have something in common. Thanks to the AD technology widely used in the deep learning neural network, the differential form constraint conditions in the differential equation are integrated into the loss function design of the neural network, so as to obtain the neural network constrained by the physical model—this is the most basic design idea of PINN.

Both PINN's network structure and loss function need to be tailored to the form of differential equations, which is different from current work in computational physics that directly utilizes machine learning algorithms. Different from the classical supervised learning task, PINN has the regularization factor of differential equation and initial boundary value condition in addition to the supervised data part in the design of loss function. These regularization factors are different and need to be tailored to achieve the optimal design according to the problem. The traditional computational differential equation numerical solution is obtained by finite difference, finite element, and other numerical methods, but the disadvantage is that it needs to give clear initial value conditions, and the numerical solution algorithm is sensitive to the boundary region; the condition slightly changed must be recalculated, which is difficult to be used in real-time calculation and prediction. PINN overcomes the problem that the traditional numerical simulation method is sensitive to



(a)

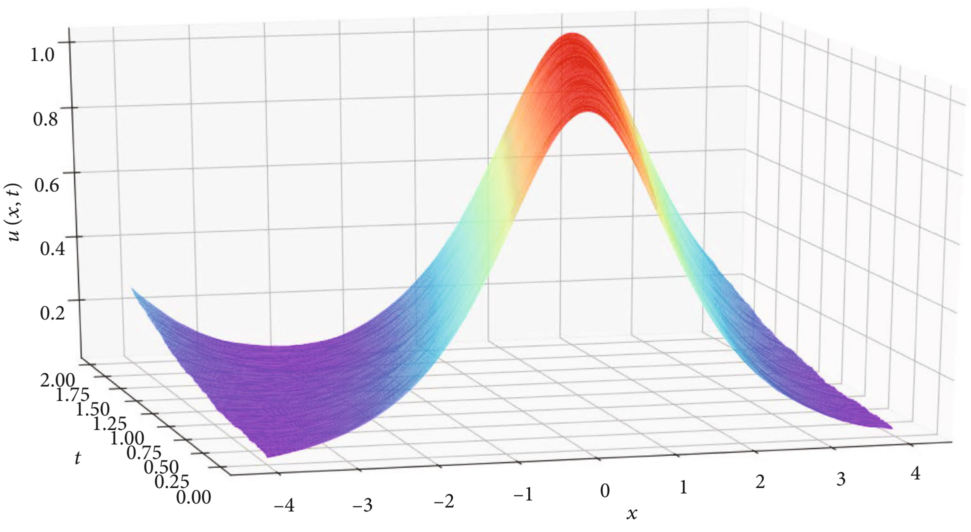


(b)

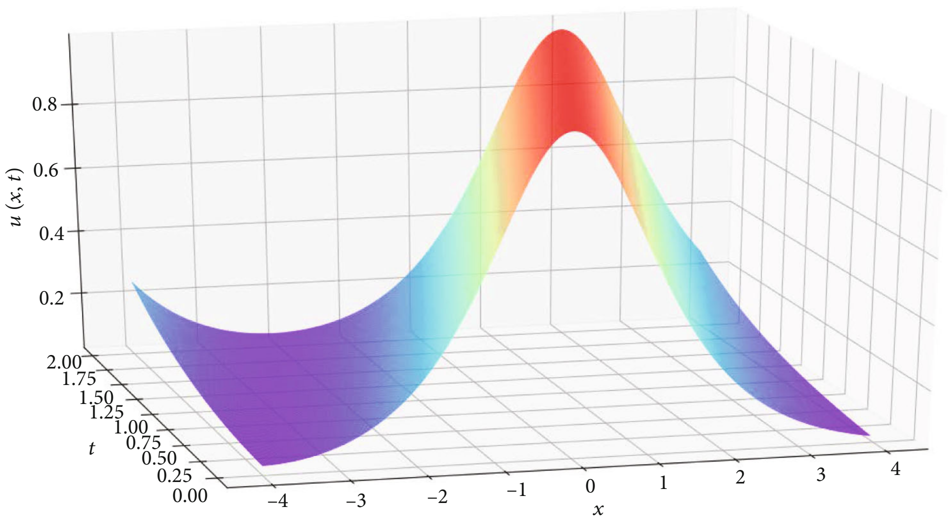
Figure 2: The projection of the exact solution (a) in the $x-t$, $x-u$, $t-u$ direction and the projection of the predicted solution (b) in the $x-t$, $x-u$, $t-u$ direction of the mkdv equation with initial state $u(x,0) = 2\exp(-x)/(\exp(-2x)+1)$.

the region and the initial and boundary conditions. Raissi et al. introduced physics-informed neural network data-driven solution, and they presented their developments in the context of solving two main classes of problems: data-driven solution and data-driven discovery of partial differential equations in 2017 [6]. Raissi et al. used multistep neural networks to study the dynamical systems of nonlinear dynamical systems in 2018 [7]. Raissi and Karniadakis study the Navier-Stokes, Schrödinger, Kuramoto-Sivashinsky, and time-dependent linear fractional equations by machine learning [8]. Liu et al. solved differential equations with neural networks in 2019 [9]. Han et al. solved high-dimensional partial differential equations by using deep learning in 2018 [10].

In the present study, we take advantage of the fast developing machine learning and use the method of PINN that was proposed by Raissi et al. [11] to study the mkdv equation. AD and L-BFGS [12] optimization algorithms had been used to train loss function. First, we introduced the main ideas of the algorithm. Second, we use the method to study two kinds of initial solutions of the mkdv equation, and the predicted solitary wave is first shown in this paper. We also show the relative $\mathscr{L}_2$-norm error between the predicted and the exact solution $u(t,x)$ for the different number of initial and boundary training data N_u and different number of collocation points N_f. The three-dimensional diagram and projected image of the exact solutions and predicted solutions of the mkdv equation with different initial solutions are shown in Figures 1–4. Finally, we conclude the paper. From the results obtained in the experiment, some novel and important developments for searching for analytical solitary wave solutions for PDE were investigated. The results of this manuscript may well complement the existing literature as the following: extended and modified direct algebraic method, extended mapping method, and Seadawy techniques to find solutions for some nonlinear partial differential equations such as dispersive solitary wave solutions of Kadomtsev-Petviashvili-Burgers dynamical equations [13]; the elliptic function, bright and dark solitons, and solitary wave solutions of higher-order NLSE [14]; abundant lump solution and interaction phenomenon of $(3+1)$-dimensional generalized Kadomtsev-Petviashvili equation [15]; describing the bidirectional propagation of small amplitude long capillary



(a)



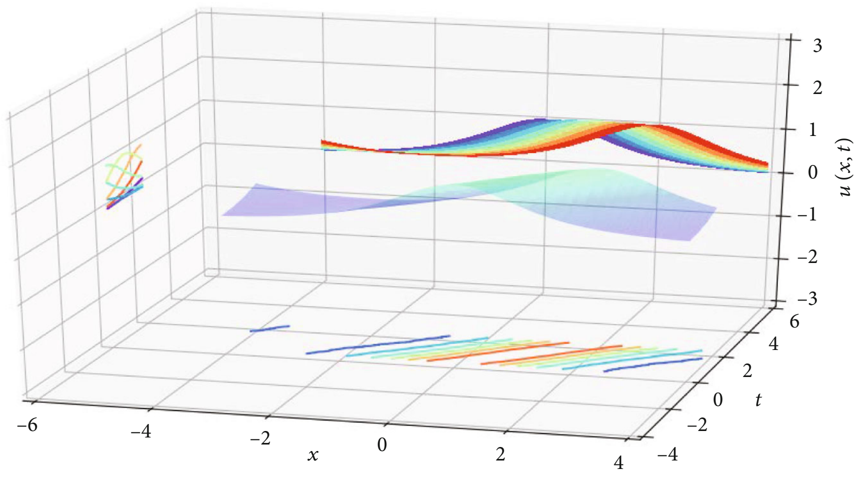
(b)

FIGURE 3: The three-dimensional diagram of the exact solutions (a) and predicted solutions (b) of the mkdv equation with initial state $u(x, 0) = 2\ \text{sech}\ (2x)$.

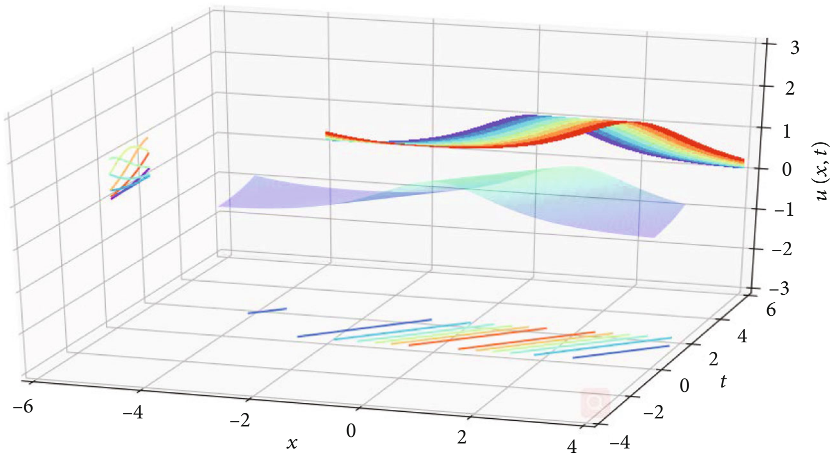
gravity waves on the surface of shallow water [16]; dispersive traveling wave solutions of the equal-width and modified equal-width equations [17]; periodic solitary wave solutions of the $(2+1)$-dimensional variable-coefficient Caudrey-Dodd-Gibbon-Kotera-Sawada equation [18]; rational solutions and lump solutions to the generalized $(3+1)$-dimensional shallow water-like equation [19]; new solitary wave solutions to coupled Maccari's system [20]; and lump solutions to a $(2+1)$-dimensional fourth-order nonlinear PDE possessing a Hirota bilinear form [21]. Therefore, this study is of significance for the later study of soliton solutions.

2. Main Ideas of the Algorithm

2.1. Illustration of the Algorithm. Deep learning is a new field in machine learning research. Its motivation lies in establishing and simulating a neural network for analysis and learning of the human brain. It mimics the mechanism of the human brain to interpret data. The concept of deep learning comes from the research of artificial neural networks. The multilayer perceptron with multiple hidden layers is a kind of deep learning structure. We give the structure of a simple neural network and deep neural network in Figure 5. In this paper, the network was used as a supervised network that means multilayer perceptron needs a teacher to tell the neural network what the desired output should be. Deep learning forms a more abstract high-level representation attribute category or feature by combining low-level features to discover distributed feature representations of data. Deep learning uses a hierarchical structure similar to neural networks. The system consists of a multilayer network consisting of an input layer, a hidden layer (multilayer), and an output layer. Only nodes in adjacent layers are connected. There is no connection between each other, and each layer can be regarded as



(a)



(b)

Figure 4: The projection of the exact solution (a) in the $x-t$, $x-u$, $t-u$ direction and the projection of the predicted solution (b) in the $x-t$, $x-u$, $t-u$ direction of the mkdv equation with initial state $u(x,0)=2\ \mathrm{sech}\ (2x)$.

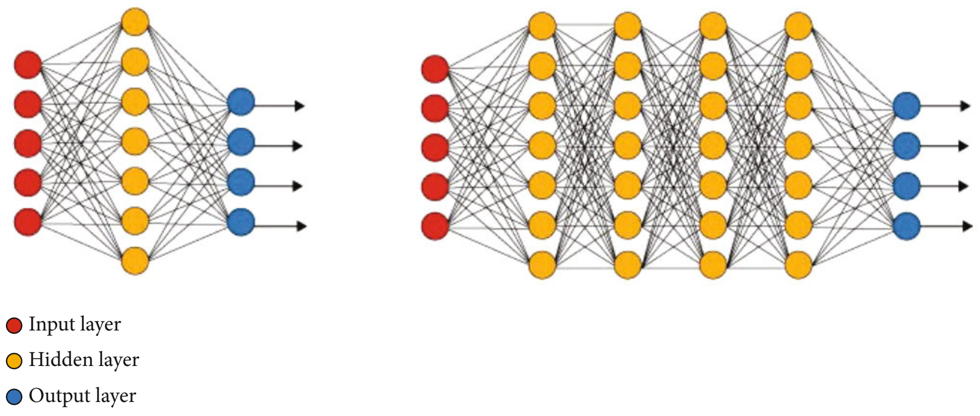


Figure 5: Basic structure of the simple neural network and deep neural network (perceptron).

a logistic regression model. Deep learning allows computers to construct complex concepts through simpler concepts with powerful capabilities and flexibility. DNN have shown great potential in approximating high-dimensional functions compared with traditional approximations based on Lagrangian interpolation or spectral methods [22].

Typical examples of deep learning models are feedforward deep networks or multilayer perceptrons. Multilayer

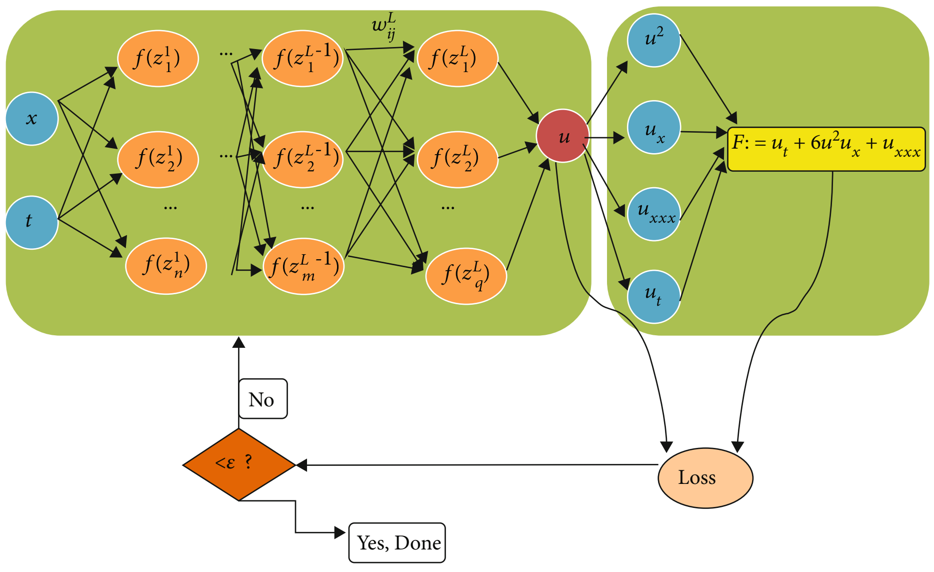


FIGURE 6: Schematic of physics-informed neural network for the mkdv equation.

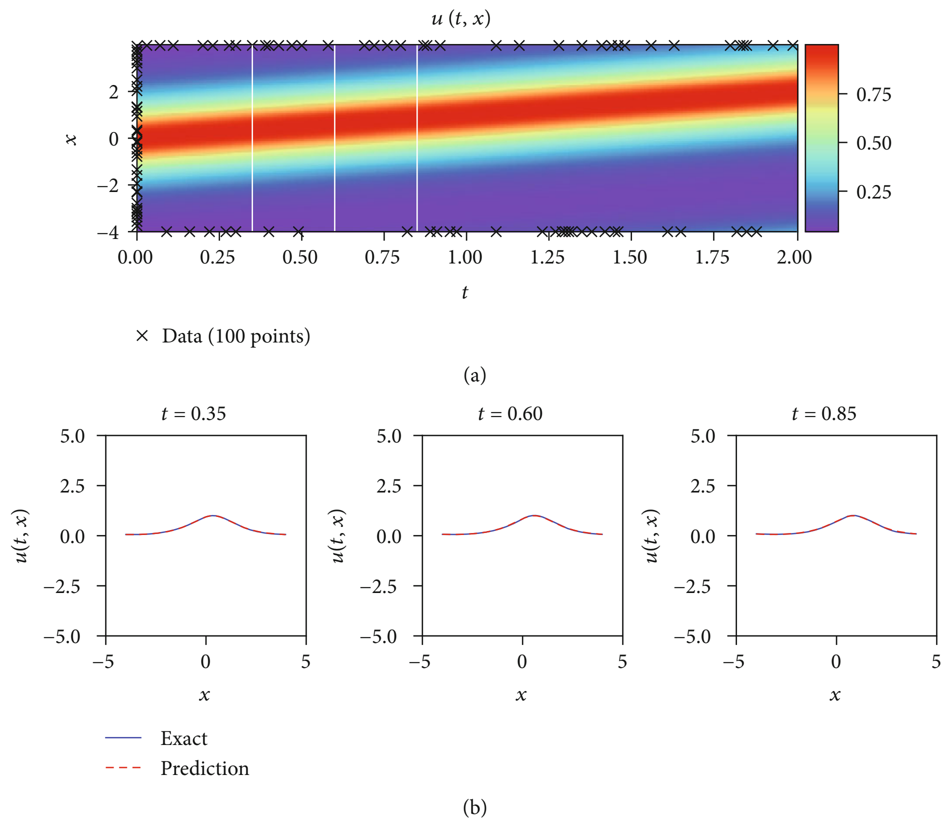


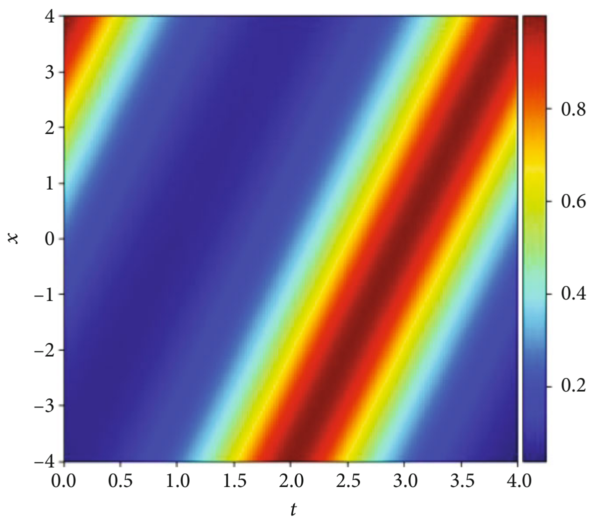
FIGURE 7: mkdv equation. (a) Predicted solution $u(t, x)$. (b) Predicted solution and exact solution at $t = 0.35, 0.60, 0.85$ were depicted by the white vertical lines in (a). The red dashed line is predicted solution, and the blue line is exact solution of $u(x, t)$.

perceptron is a mathematical function that maps input to output value. This function is composed of many simpler functions. Each layer of a fully connected DNN can be expressed as follows:

$$z_l = f\left(W^{(l)} z_{l-1} + b^{(l)}\right), \quad l = 1, \cdots, L-1, \tag{1}$$

$$\begin{aligned} z_L &= N(z_0\,;\vartheta) \\ &= f\left(W^{(L)} f\left(\cdots f\left(W^{(2)} f\left(W^{(1)} z_0 + b^{(1)}\right) + b^{(2)}\right)\right) + b^{(L)}\right), \end{aligned} \tag{2}$$

$$\vartheta = \left\{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, \cdots, W^{(L)}, b^{(L)}\right\}, \tag{3}$$



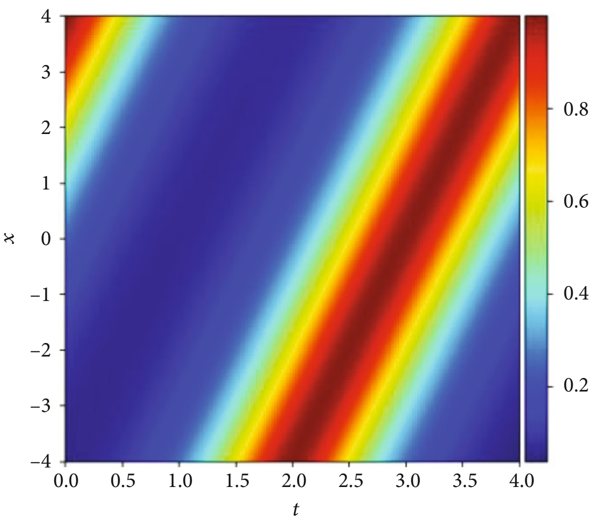


FIGURE 8: Exact dynamics and learned dynamics of $u(x, t)$.

where z_0 means input vector and f is activation function (we choose hyperbolic tangent function as the activation function where $\tanh = (e^x - e^{(-x)})/(e^x + e^{(-x)})$):

$$W^{(k)} = \begin{pmatrix} w_{11}^k & \cdots & w_{1n_{k-1}}^k \\ \vdots & \ddots & \vdots \\ w_{n_k 1}^k & \cdots & w_{n_k n_{k-1}}^k \end{pmatrix},$$
$$b^{(k)} = \begin{pmatrix} b_1^k \\ b_2^k \\ \vdots \\ b_{n_k}^k \end{pmatrix}, \tag{4}$$

$k = 1, 2, \cdots, L$, $W^{(k)}$ is the weights from layer $k-1$ to layer k, and w_{ij} means the weight between the j-input and i-neuron of the hidden layer. $b^{(k)}$ is bias vector. $N(z_0\,;\vartheta)$ can be considered an approximate solution for $u(x, t)$ of a PDE. The final approximate solution is solved by adjusting the parameters ϑ to minimize the error of the approximate solution and the exact solution.

A fully connected neural network was previously proven [23, 24] by Jones and Carroll and Dickinson that any continuous function defined in a finite domain can be approximated. In this paper, we introduced the form and construction of the solution of PDE using the physics-informed neural network method. The schematic of the physics-informed neural network for the mkdv equation is shown in Figure 6. Consider the general form of the PDE as follows:

$$u_t + \mathcal{N}(x, u, u_x, u_{xx}, \cdots) = 0, \quad x \in \Omega, t \in [0, T], \tag{5}$$

where $\mathcal{N}$ is a nonlinear function of time t, space x, solution u, and its derivatives and the subscripts denote partial differentiation in either time t or space x. For example, u_{xx} is the second derivative of u with respect to x.

TABLE 1: mkdv equation: $\mathcal{L}_2$-norm error between the predicted and exact solutions of $u(t, x)$ for different numbers of hidden layers and different numbers of neurons per layer. The number of training points is $N_u = 100$, and collocation points is $N_f = 20000$.

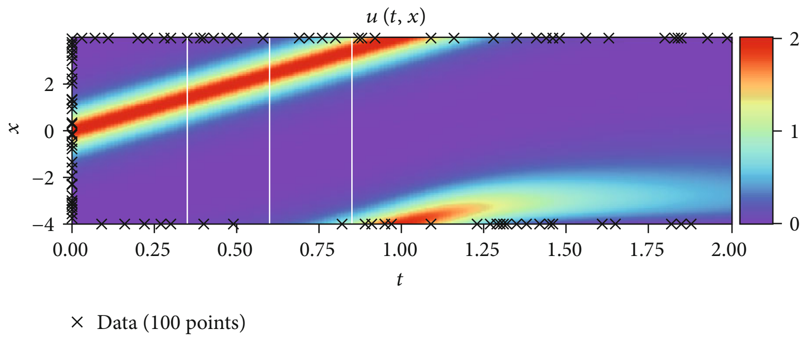
Neurons $\mathcal{L}_2$-norm error Layers	10	15	25	30
2	$2.14e-05$	$1.29e-05$	$1.22e-05$	$1.75e-05$
5	$1.85e-05$	$2.01e-05$	$2.00e-05$	$1.28e-05$
7	$2.01e-05$	$2.32e-05$	$1.20e-05$	$1.08e-05$
9	$1.40e-05$	$1.45e-05$	$1.28e-05$	$1.17e-05$

TABLE 2: mkdv equation: $\mathcal{L}_2$-norm error between the predicted and exact solutions of $u(t, x)$ for DNN architecture which is constructed by 9 hidden layers with 20 neurons per hidden layer with different training points N_u and collocation points N_f.

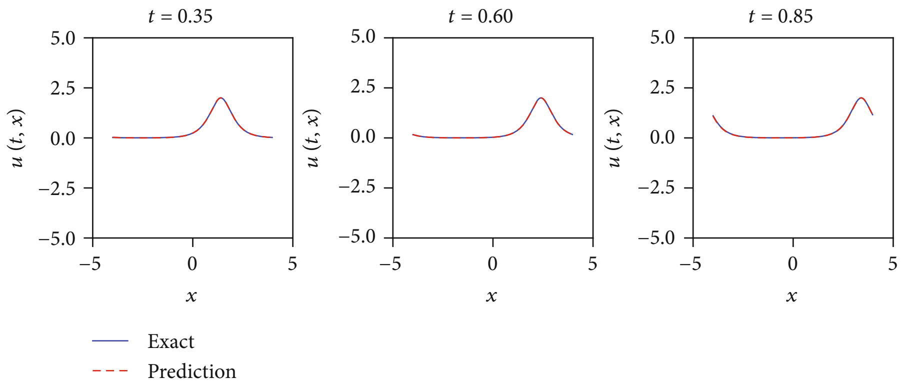
N_f $\mathcal{L}_2$-norm error N_u	1000	5000	9000	10000
30	$8.56e-05$	$1.15e-05$	$7.22e-06$	$1.56e-06$
50	$7.97e-06$	$1.54e-05$	$1.76e-05$	$1.67e-06$
80	$1.23e-05$	$1.42e-05$	$1.10e-05$	$1.58e-06$
90	$1.17e-05$	$1.35e-05$	$1.95e-05$	$1.62e-06$
100	$1.05e-05$	$1.34e-05$	$2.14e-06$	$1.67e-06$

2.2. Details of the Algorithm. According to Equation (1), let us define $f(t, x)$ as follows:

$$f(t, x) := u_t + \mathcal{N}(x, u, u_x, u_{xx}, \cdots). \tag{6}$$



(a)



(b)

FIGURE 9: mkdv equation. (a) Predicted solution $u(t, x)$. (b) Predicted solution and exact solution at $t = 0.35, 0.60, 0.85$ were depicted by the white vertical lines in (a). The red dashed line is predicted solution, and the blue line is exact solution of $u(x, t)$.

Objective function of the trial function can be defined as [25]

$$\mathrm{MSE} = \mathrm{MSE}_u + \mathrm{MSE}_f, \tag{7}$$

where mean square errors are defined, respectively, as MSE_u and MSE_f:

$$\begin{aligned} \mathrm{MSE}_u &= \frac{1}{N_u}\sum_{i=1}^{N_u}\left\|u\left(t_u^i, x_u^i\right) - u^i\right\|^2, \\ \mathrm{MSE}_f &= \frac{1}{N_f}\sum_{i=1}^{N_f}\left\|f\left(t_f^i, x_f^i\right)\right\|^2. \end{aligned} \tag{8}$$

The objective function of DNN training is performed by the mean squared error on the network outputs.

The weight and bias between the neural networks $u(t, x)$ and $f(t, x)$ can be learned by minimizing the mean squared error loss, t_f^i, x_f^i were domain data, N_u is the number of sampling points on the boundary, N_f is the number of sampling points on the region, t_u^i, x_u^i, u^i were initial and boundary training data on $u(t, x)$, and $u(t_u^i, x_u^i)$ is predicted solution.

3. Example for Modified Korteweg-de Vries Equation

The modified Korteweg-de Vries (mkdv) equation may be written as [26]

$$\begin{aligned} u_t + 6u^2 u_x + u_{xxx} = 0, \\ u(x, 0) = \frac{2\exp(-x)}{\exp(-2x) + 1}. \end{aligned} \tag{9}$$

If $-4 \le x \le 4$, $0 \le t \le 2$. We got the training and test data by using conventional spectral methods and using the Chebfun package [27] with a spectral Fourier discretization with 256 modes and a fourth-order explicit Runge-Kutta temporal integrator with time-step size 10^{-4}.

There are two parts of data points to form the collocation points of training $f(x, t)$: one part used the Latin hypercube sampling strategy to generate 10000 data points and the other part uses random sampling to generate 456 data points. Randomly extract $N_u = 100$ data from the initial and boundary data as training points, and we learn the latent solution $u(t, x)$ by using the L-BFGS algorithm to optimize the parameters to minimize the error function Equation (7). We had shown the predicted solution $u(x, t)$ in Figure 7 by an 11-layer deep neural network in which each hidden layer contained 30 neurons. The relative $\mathscr{L}_2$-norm error for this

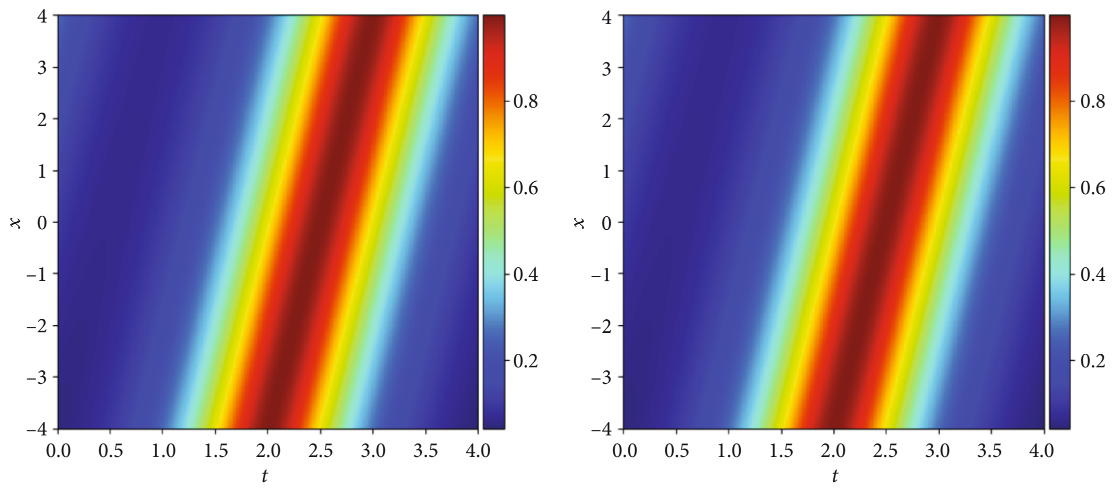


Figure 10: Exact dynamics and learned dynamics of $u(x,t)$.

case is $9.402109 \cdot 10^{-6}$. The code runs on a personal laptop with Intel® Core™ i5, 2.50 GHz, and the running time is 930.9392 seconds. Judging from the physical propagation diagram of the exact solution which is a soliton solution obtained by the Chebfun package and the predicted solution at the bottom of Figure 7, the waveform of the single soliton has not changed over time. The exact dynamics and learned dynamics of $u(x,t)$ are shown in Figure 8. We choose the number of training points as $N_u = 100$ and collocation points as $N_f = 20000$; under this condition, we study the influence of different layers and different neurons on the relative $\mathscr{L}_2$-norm error. The relative $\mathscr{L}_2$-norm error tends to decrease with the increase of layers and neurons, and it is shown in Table 1. We also studied the effect of the DNN architecture which is constructed by 9 layers with 20 neurons per hidden layer with different training points N_u and collocation points N_f on relative $\mathscr{L}_2$-norm error which is shown in Table 2. The three-dimensional diagram and projected image of the exact solutions and predicted solutions of the mkdv equation with initial state $u(x,0) = 2\exp(-x)/(\exp(-2x)+1)$ are shown in Figures 1 and 2.

In order to further study the effectiveness of the performance of the algorithm to approximate the exact solutions of mkdv equations, we change the initial condition as follows [28]:

$$
\begin{gathered}
u_t + 6u^2 u_x + u_{xxx} = 0, \\
u(x,0) = 2\,\mathrm{sech}\,(2x).
\end{gathered} \tag{10}
$$

We got the training and test data by using conventional spectral methods and using the Chebfun package with a spectral Fourier discretization with 256 modes and a fourth-order explicit Runge-Kutta temporal integrator with time-step size 10^{-4}. The data points used to obtain $f(x,t)$ are divided into two parts, one part used the Latin hypercube sampling strategy to generate 10000 data points and the other part uses random sampling to generate 456 data points. Randomly extract $N_u = 100$ data from the initial and boundary data as training points, and

Table 3: mkdv equation: $\mathscr{L}_2$-norm error between the predicted and exact solutions of $u(t,x)$ for different numbers of hidden layers and different numbers of neurons per layer. The number of training points is $N_u = 100$ and collocation points is $N_f = 10000$.

Neurons / $\mathscr{L}_2$-norm error / Layers	15	20	25	30
2	$1.13e-05$	$1.08e-05$	$7.95e-06$	$8.78e-06$
5	$1.25e-05$	$8.55e-06$	$8.10e-06$	$8.64e-06$
7	$9.18e-06$	$9.07e-06$	$8.46e-06$	$8.62e-06$
9	$1.02e-05$	$1.21e-05$	$1.59e-05$	$9.40e-06$

Table 4: mkdv equation: $\mathscr{L}_2$-norm error between the predicted and exact solutions of $u(t,x)$ for DNN architecture which is constructed by 9 hidden layers with 15 neurons per hidden layer with different training points N_u and collocation points N_f.

N_f / $\mathscr{L}_2$-norm error / N_u	3000	5000	8000	10000
50	$1.25e-05$	$2.75e-05$	$1.37e-05$	$1.29e-05$
80	$1.64e-05$	$1.71e-05$	$1.76e-05$	$1.39e-05$
90	$1.19e-05$	$1.08e-05$	$1.57e-05$	$9.07e-06$
100	$1.77e-05$	$1.21e-05$	$1.01e-05$	$1.02e-06$

we learn the latent solution $u(t,x)$ by using the L-BFGS algorithm to optimize the parameters to minimize the error function Equation (7). We had shown the predicted solution $u(x,t)$ in Figure 9 by an 11-layer deep neural network in which each hidden layer contained 15 neurons. Running time of the code is 439.9942 seconds. The relative $\mathscr{L}_2$-norm error for this case is $1.0333972 \cdot 10^{-5}$. Judging from the physical propagation diagram of the exact solution which is a soliton solution obtained by the Chebfun package and the predicted solution in Figure 9, the waveform of the single soliton has not changed over time. The exact dynamics and learned dynamics of $u(x, t)$ are shown in Figure 10. We choose the number of training points as $N_u = 100$ and collocation points as $N_f = 10000$; under this condition, we study the influence of different layers and different neurons on the relative $\mathscr{L}_2$-norm error. The relative $\mathscr{L}_2$-norm error tends to decrease with the increase of layers and neurons, and it is shown in Table 3. We also studied the effect of the DNN architecture constructed by 9 layers with 15 neurons per hidden layer with different training points N_u and collocation points N_f on relative $\mathscr{L}_2$-norm error which is shown in Table 4. The three-dimensional diagram and projected image of the exact solutions and predicted solutions of the mkdv equation with initial state $u(x, 0) = 2\ \mathrm{sech}\ (2x)$ are shown in Figures 3 and 4.

4. Conclusions

With the increase of data volume, the improvement of computing power, and the emergence of new machine learning algorithms (deep learning), artificial intelligence has become a field with many practical applications and active research topics. Deep learning is one of the ways to artificial intelligence. It is a type of machine learning, a technology that enables computer systems to be improved from experience and data.

In this paper, we briefly describe details of the algorithm of DNN. Figures 5–10 show the basic structure of the simple neural network and deep neural network, schematic of the physics-informed neural network, and comparison diagram of the precise dynamical system and the predicted dynamical system of the mkdv equation. Tables 1 and 3 show $\mathscr{L}_2$-norm error between the predicted and exact solutions of $u(t, x)$ for different numbers of hidden layers and different numbers of neurons per layer. Tables 2 and 4 show the relative $\mathscr{L}_2$ error between the predicted and exact solutions $u(t, x)$ for different numbers of training points N_u and collocation points N_f. Tables 1–4 illustrate the relative $\mathscr{L}_2$ error that tends to decrease with the increase in layers and neurons. This method demonstrates the strong mathematical and physical ability of deep learning to simulate the physical dynamic state represented by differential equations and also opens the way for us to understand more physical phenomena later.

Data Availability

The data in the manuscript can be generated by MATLAB software. The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflict of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (11571008, 11661060 and 12061054) and Natural Science Foundation of Inner Mongolia Autonomous Region of China (2018LH01013).

References

[1] N. Yadav, A. Yadav, and M. Kumar, "An introduction to neural network methods for differential equations," in *SpringerBriefs in Applied Sciences and Technology*, Springer, London, 2015.

[2] Z. Liu, Y. Yang, and Q. D. Cai, "Solving differential equation with constrained multilayer feedforward network," 2019, https://arxiv.org/abs/1904.06619.

[3] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[4] K. S. Mcfall, *An Artificial Neural Network Method for Solving Boundary Value Problems with Arbitrary Irregular Boundaries*, 2006, Georgia Institute of Technology.

[5] S. Mall and S. Chakraverty, "Application of Legendre neural network for solving ordinary differential equations," *Applied Soft Computing*, vol. 43, pp. 347–356, 2016.

[6] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations," 2017, https://arxiv.org/abs/1711.10561.

[7] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Multistep neural networks for data-driven discovery of nonlinear dynamical systems," 2018, https://arxiv.org/abs/1801.01236.

[8] M. Raissi and G. E. Karniadakis, "Hidden physics models: machine learning of nonlinear partial differential equations," *Journal of Computational Physics*, vol. 357, pp. 125–141, 2018.

[9] Z. Y. Liu, Y. T. Yang, and Q. D. Cai, "Neural network as a function approximator and its application in solving differential equations," *Applied Mathematics and Mechanics*, vol. 40, no. 2, pp. 237–248, 2019.

[10] J. Han, A. Jentzen, and W. E, "Solving high-dimensional partial differential equations using deep learning," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, no. 34, pp. 8505–8510, 2018.

[11] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part II): data-driven solutions of nonlinear partial differential equations," 2017, https://arxiv.org/abs/1711.10561.

[12] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[13] R. A. Seadawy, "Ion acoustic solitary wave solutions of two-dimensional nonlinear Kadomtsev-Petviashvili-Burgers equation in quantum plasma," *Mathematical Methods in the Applied Sciences*, vol. 40, no. 5, pp. 1598–1607, 2017.

[14] M. Arshad, A. R. Seadawy, and D. Lu, "Elliptic function and solitary wave solutions of the higher-order nonlinear Schrödinger dynamical equation with fourth-order dispersion and cubic-quintic nonlinearity and its stability," *European Physical Journal Plus*, vol. 132, no. 8, pp. 371–382, 2017.

[15] J. Lü, S. Bilige, and X. Gao, "Abundant lump solution and interaction phenomenon of (3+1)-dimensional generalized Kadomtsev–Petviashvili equation," *International Journal of Nonlinear Sciences and Numerical Simulation*, vol. 20, no. 1, pp. 33–40, 2019.

[16] M. A. Helal, A. R. Seadawy, and M. H. Zekry, "Stability analysis of solitary wave solutions for the fourth-order nonlinear Boussinesq water wave equation," *Applied Mathematics and Computation*, vol. 232, no. 232, pp. 1094–1103, 2014.

[17] D. Lu, A. R. Seadawy, and A. Ali, “Dispersive traveling wave solutions of the equal-width and modified equal-width equations via mathematical methods and its applications,” *Results in Physics*, vol. 9, pp. 313–320, 2018.

[18] Y. Zhang, “Periodic solitary wave solutions of the (2 + 1)-dimensional variable-coefficient Caudrey-Dodd-Gibbon-Kotera-Sawada equation,” *Open Journal of Applied Sciences*, vol. 10, no. 3, pp. 60–68, 2020.

[19] Y. Zhang, H. Dong, X. Zhang, and H. Yang, “Rational solutions and lump solutions to the generalized (3+1)-dimensional shallow water-like equation,” *Computers and Mathematics with Applications*, vol. 73, no. 2, pp. 246–252, 2017.

[20] J. Vahidi, S. M. Zekavatmand, H. Rezazadeh, M. Inc, M. A. Akinlar, and Y. M. Chu, “New solitary wave solutions to the coupled Maccari's system,” *Results in Physics*, vol. 21, article 103801, 2021.

[21] W. X. Ma, S. Manukure, H. Wang, and S. Batwa, “Lump solutions to a (2+1)-dimensional fourth-order nonlinear PDE possessing a Hirota bilinear form,” *Modern Physics Letters B*, vol. 35, no. 9, article 2150160, 2021.

[22] W. Cai, X. G. Li, and L. Z. Liu, “A phase shift deep neural network for high frequency wave equations in inhomogeneous media,” 2019, https://arxiv.org/abs/1909.11759.

[23] L. K. Jones, “Constructive approximations for neural networks by sigmoidal functions,” *Proceedings of the IEEE*, vol. 78, pp. 1586–1589, 1990.

[24] S. Carroll and B. Dickinson, “Construction of neural networks using the Radon transform,” *IEEE International Conference on Neural Networks*, vol. 1, pp. 607–611, 1989.

[25] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[26] Z. Z. Dong, *Symmetric Reduction and Exact Solutions of Some Nonlinear Problems*, East China Normal University, 2010.

[27] T. A. Driscoll, N. Hale, and L. N. Trefethen, Eds., *Chebfun Guide*, Pafnuty Publications, Oxford, 2014.

[28] Y. G. Zhao and J. R. Yan, “The theory of the perturbation equation of MKdV,” *Acta Physica Sinica*, vol. 48, no. 11, pp. 1976–1982, 1999.