Nonlinear Dyn (2023) 111:15279–15291
https://doi.org/10.1007/s11071-023-08614-4

ORIGINAL PAPER

Boussinesq equation solved by the physics-informed neural networks

Ruozhou Gao · Wei Hu · Jinxi Fei · Hongyu Wu

Received: 13 April 2023 / Accepted: 17 May 2023 / Published online: 8 June 2023

Abstract Physics-informed neural networks (PINNs) model is utilized to achieve the first- and second-order rogue wave solvers of the Boussinesq equation with different initial and boundary conditions. A new generalized loss term handling the initial first-order derivate is included in the simulations to guarantee the high prediction accuracies in the adaptive PINNs (APINNs) and the gradient-optimized PINNs (GPINNs) models, with a new regularization parameter being considered in the latter case. Learned results with high precision are fulfilled in the large domain simulations by applying more collocation points and more weight parameters in the neural network architecture. The APINNs model currently can be made use of in more situations with high prediction accuracies, while the GPINNs model is more robust in the current research where the initial condition is distributed in the localized sharp areas. Parallel computing is carried out to get the mean relative L_2-norm errors efficiently in the GPINNs model due to the random choosing of the simulation points during the training iterations.

Keywords Modified physics-informed neural networks · Boussinesq equation · Rogue wave · Parallel computing

R. Gao · W. Hu (✉) · J. Fei · H. Wu
College of Engineering, Lishui University, Lishui 323000, China
e-mail: wickhu@lsu.edu.cn; wickhw@foxmail.com

1 Introduction

Deep learning [1] with the neural network model has achieved great success in artificial intelligence including image recognition [2], computer vision [3], and natural language translation [4]. Scientific computation also benefits due to the ability of deep learning on solving the high-dimensional nonlinear partial differential equations (NPDEs) [5–8]. Physics-informed neural networks (PINNs) invented by Rassi et al. [9] stand out to be a powerful deep learning tool for manipulating a variety of NPDEs. Researchers from the same department have made great improvements to the original software such as the adaptive activation function PINNs method (APINNs) to enhance the robustness and convergence speedup [10–12] and the domain decomposition approaches (CPINNs and XPINNs) employing the parallel GPU computing to reduce the training cost more effectively [13–15]. By monitoring the distribution of the back-propagated gradients of loss with respect to the neural network parameters during training motivated by Glorot and Bengio [16], gradient-optimized PINNs (GPINNs) model is put forward to balance the interplay between different terms in composite loss functions to increase the predictive accuracy of at least one order [17,18]. A novel gradient descent algorithm proposed to utilize the eigenvalues of the neural tangent kernel to adaptively calibrate the convergence rate of the total training error has been verified with a series of numerical experiments [19]. Gradient-

enhanced PINNs algorithms leverage gradient information of the partial differential equation residual and embed the gradient into the loss function to improve the accuracy and training efficiency [20]. A lot of works have been carried out based on the above PINNs software, such as the solver of the local wave solutions of the second- and third-order NPDEs and the nonlinear Schrödinger equations [21–23]. Bilinear neural network method (BNNM) [24] and bilinear residual network method (BRNM) [25] introduced by Zhang et al. are outstanding universal methods for seeking the exact analytical solution of NPDEs based on the deep neural network model, which is different from the numerical solver by the PINNs model. BNNM model has been carried out to solve the rogue waves of the generalized breaking soliton equation [26], the lump solutions, and rogue waves of the Caudrey–Dodd–Gibbon–Kotera–Sawada-like equation [27], the interference wave and the bright and dark soliton of the two integro-differential equation [28], the rogue wave solutions and the bright and dark solitons of the Jimbo–Miwa equation [29], and the bright–dark solitons and interaction phenomenon for p-gBKP equation [30].

Rogue wave [31–33], Painlevé integrability [34,35], and solitary wave [36,37] solvers of the NPDEs have been paid great attention to in the nonlinear scientific fields. Here, we are concerned with the rogue wave solutions of the Boussinesq equation, which was introduced by Joseph Boussinesq in 1871 to describe the propagation of long waves in shallow water [38–41]. The Boussinesq equation is also a soliton equation solvable by inverse scattering [42–44] which arises in several physical applications including one-dimensional nonlinear lattice waves [45,46]; vibrations in a nonlinear string [47]; and ion sound waves in a plasma [48,49]. Especially, the Boussinesq equations have been applied to describe the nonlinear phenomena in ocean, coastal, harbor, and water engineering [50–53].

In this paper, the modified APINNs and GPINNs models are utilized to solve the first- and second-order rogue wave solvers of the Boussinesq equation with the form of $w_{tt}+\mathcal{N}_x[w]=0$, in which $\mathcal{N}_x[w]$ is a nonlinear operator acting on w with respect to x. This is different from the common form $w_t+\mathcal{N}_x[w]=0$ used in many PINNs simulations as the second-order derivate of w about t would require a new loss term to the total loss function in the PINNs model. The Klein–Gordon equation [10,17,54] and wave equation [54,55] in the same form of $w_{tt}+\mathcal{N}_x[w]=0$ as above have been solved by the PINNs method, but the loss term in their work can only handle the case that the initial one-order derivate term $w_t[t=0]$ is 0 at $t=0$. We extend the new loss term about the initial condition about the first-order time derivate term to the generalized situations, which is key to the final simulation results with high precision. The large simulation domain researches on the Boussinesq equations are also carried out in some cases to describe the ability and robustness of the modified APINNs and GPINNs methods. Due to the larger simulation domain and higher-order differential form, much more weight parameters are utilized in the modified GPINNs model compared with the former work [17,18]. The APINNs model can handle more simulations with different initial and boundary (IB) conditions in high accuracies than the GPINNs model on solving the Boussinesq equations, while the GPINNs model is more robust where the initial condition is distributed in the localized sharp areas. Parallel computing is fulfilled and optimized on the supercomputing platform to accelerate the simulations for the GPINNs model.

The rest of the paper is organized as follows. In Sect. 2, we propose the total loss function with a new loss term to be minimized in the current simulations for the modified APINNs and GPINNs models. Each loss term is specified by combining the IB conditions or the simulation domains. In Sect. 3, the training data, the optimization method, and the computing platform are described in detail. The first- and second-order rogue wave solvers of the Boussinesq equation are carried out for the different IB conditions with the two models, and good agreement is achieved between the theoretical and numerical results. Finally, brief conclusions and discussion are given in Sect. 4.

2 Model

Firstly, the Boussinesq equation with IB conditions is introduced in the following:

$$\begin{cases} w_{tt}+\mathcal{N}_x[w]=0, & x\in[-X,X], t\in[T_1,T_2], \\ w[T_1,x]=I(x), & x\in[-X,X], \\ w_t[T_1,x]=Ip(x), & x\in[-X,X], \\ w[t,x]=B(t,x), & t\in[T_1,T_2], x=\pm X, \end{cases} \tag{1}$$

where x_1, x_2 and T_1, T_2 are spatial and temporal domain boundaries, respectively. $I(x)$, $Ip(x)$, $B(t, -X)$, and $B(t, X)$ represent the IB conditions. $\mathcal{N}_x[w]$ is a nonlinear operator acting on w with respect to x. Now, the key to computing data-driven solutions to the Boussinesq equation will focus on the neural network setup based on the APINNs and GPINNs models. The two models utilize the automatic differentiation method and backpropagation algorithm to train the deep neural network by calculating the derivative of the loss function with respect to Θ denoting the weights and biases. For the current Boussinesq equation, the loss function is defined as follows utilizing the mean squared error method.

$$\begin{aligned} J(\Theta) &= J_w(\Theta) + J_I(\Theta) + J_{Ip}(\Theta) + J_B(\Theta) \\ &= \frac{1}{N_w}\sum_{i=1}^{N_w} |f_\Theta(t_w^i, x_w^i)|^2 \\ &\quad + \frac{\lambda_I}{N_I}\sum_{i=1}^{N_I} |w_\Theta(t_I^i, x_I^i) - w^i|^2 \\ &\quad + \frac{\lambda_{Ip}}{N_{Ip}}\sum_{i=1}^{N_{Ip}} |w_\Theta(t_{Ip}^i, x_{Ip}^i) - w^i|^2 \\ &\quad + \frac{\lambda_B}{N_B}\sum_{i=1}^{N_B} |w_\Theta(t_B^i, x_B^i) - w^i|^2. \end{aligned} \tag{2}$$

Here, (t_I^i, x_I^i, w^i) and $(t_{Ip}^i, x_{Ip}^i, w^i)$ represent the initial training data on $w(t, x)$, with (t_B^i, x_B^i, w^i) referring to the boundary training data on $w(t, x)$. (t_w^i, x_w^i) specify the collocations points for $w(t, x)$ in the spatial and temporal domain ($x \in [x_1, x_2], t \in [T_1, T_2]$). A new generalized loss term $J_{Ip}(\Theta)$ added in the initial condition is due to the second-order derivate of $w(t, x)$ with respect to t compared with the first-order derivate in the common PINNs model, i.e., $w_t + \mathcal{N}_x[w] = 0$. This loss term to be minimized plays an important role in the following simulations, which is newly added in the APINNs model and modified in the GPINNs model. The loss term $J_w(\Theta)$ to be minimized can guarantee that the equation $w_{tt} + \mathcal{N}_x[w] = 0$ will be learned precisely at a finite set of collocation points. The other three loss terms $J_I(\Theta)$, $J_{Ip}(\Theta)$, and $J_B(\Theta)$ are minimized to fulfill the training of the IB points. Regularization parameters λ_I, λ_{Ip}, and λ_B are set as 1 for APINNs model but will be iterated for the GPINNs model based on the adaptive learning rate annealing algorithm that aims to balance the interplay between the different loss terms [17]. A new parameter λ_{Ip} is introduced with respect to the new loss term, which is updated in the iterations in GPINNs model.

3 Simulation of Boussinesq equation

The current nonlinear (1+1)-dimensional fourth-order Boussinesq equation is carried out, which is in the following form:

$$w_{tt} + w_{xx} - 2w_x^2 - 2ww_{xx} - \frac{1}{3}w_{xxxx} = 0. \tag{3}$$

The trained neural networks are initialized with Xavier methods, and the hyperbolic tangent activation functions are set in the forward propagation process. Dirichlet boundary conditions are chosen in the spatial domain. For the APINNs model, the Adam optimizer method with the adaptive activation function is employed first, which can enhance the robustness and accuracy of the neural network approximation of the nonlinear function as well as the solutions of the equation. The L-BFGS algorithm is then utilized to continue the iteration of the neural network parameters. The default maximum optimizer iteration steps for the Adam and L-BFGS algorithms are chosen to be 20, 000. For the GPINNs model, only the Adam optimizer is made use of and the default number of iterations is selected as 80,000. The number of collocation points N_w is set as 10,000 by default which is generated by the Latin hypercube sampling method which are randomly parsed from the full high-resolution data set in the APINNs model. The default value of N_w is 256 in the GPINNs model which is much smaller than that in the APINNs model due to the different learning algorithms. The positions of the N_w points will be randomly changed in the iterations to cover all the simulation domains in the GPINNs model. The default IB training data $N_I = 200$, $N_{Ip} = 200$, and $N_B = 400$ being equally shared by the lower and upper boundary conditions. For some simulations, the default simulation parameters are changed and will be specifically described in the two models. The weights and biases parameters in the neural network are finally learned using the above two algorithms along with the IB training data and collocation points to minimize the total loss function. The simulations are based on Python 3.7 and TensorFlow_GPU−1.14 with the computing plat-

forms containing the K80 GPU with Linux system and the RTX 2070 GPU with Windows system. The analytical first- and second-order rogue wave solvers of Eq. (3), which will be compared with the prediction results by the modified APINNs and GPINNs models, are given in Eqs. (4) and (5), respectively:

$$w(t,x) = \frac{4(1+t^2-x^2)}{(1+t^2+x^2)^2}, \tag{4}$$

$$w(t,x) = \frac{4A(t,x)}{B(t,x)}, \tag{5}$$

where

$$\begin{aligned} A(t,x) =& -78125 - 2268t^6x^2 - 2430t^4x^4 \\ &+ 1620t^2x^6 + 138750t^4 - 15750x^6 \\ &+ 25470t^6 + 109375t^2 + 265625x^2 \\ &+ 33750x^2t^4 + 47250x^4t^2 + 382500t^2x^2 \\ &+ 93750x^4 + 3807t^8 - 2025x^8 + 243t^{10} \\ &- 243x^{10} + 729t^8x^2 + 486t^6x^4 \\ &- 486t^4x^6 - 729t^2x^8, \end{aligned} \tag{6}$$

$$\begin{aligned} B(t,x) =& (625 + 475t^2 - 125x^2 + 51t^4 + 270t^2x^2 \\ &+ 75x^4 + 9t^6 + 27x^2t^4 + 27x^4t^2 + 9x^6)^2. \end{aligned} \tag{7}$$

3.1 First-order rogue wave

The first-order rogue wave solver of the Boussinesq equation is first selected to provide the IB training data for the modified APINNs model. Three cases are run with different simulation domains, which are $t \in [0,5]$, $x \in [-10,10]$, $t \in [0,5]$, $x \in [-40,40]$, and $t \in [-3,3]$, $x \in [-10,10]$. Thus, three different IB training data are imported in the modified APINNs model. The IB conditions are given in Eqs. (8), (9), and (10), respectively.

$$\begin{cases} I(x)=\frac{4(1-x^2)}{(1+x^2)^2}, & x \in [-10,10], \\ Ip(x)=0, & x \in [-10,10], \\ w[t,-10]=w[t,10]=\frac{4(1+t^2-100)}{(1+t^2+100)^2}, & t \in [0,5]. \end{cases} \tag{8}$$

$$\begin{cases} I(x)=\frac{4(1-x^2)}{(1+x^2)^2}, & x \in [-40,40], \\ Ip(x)=0, & x \in [-40,40], \\ w[t,-10]=w[t,10]=\frac{4(1+t^2-1600)}{(1+t^2+1600)^2}, & t \in [0,5]. \end{cases} \tag{9}$$

$$\begin{cases} I(x)=\frac{4(10-x^2)}{(10+x^2)^2}, & x \in [-10,10], \\ Ip(x)=\frac{24(10-3x^2)}{(10+x^2)^3}, & x \in [-10,10], \\ w[t,-10]=w[t,10]=\frac{4(1+t^2-100)}{(1+t^2+100)^2}, & t \in [-3,3]. \end{cases} \tag{10}$$

For the first-order rogue wave, the $x_{\max} = 10$ and $x_{\max} = 40$ cases are carried out first with the initial conditions at $t = 0$. In the $x_{\max} = 10$ case, the neural network contains 4 hidden layers and 40 neurons are included in each layer. The simulation results are verified to be convergent when adding more hidden layers and more neurons in an appropriate way. For the $x_{\max} = 40$ case, a new setup of 6 hidden layers with 40 neurons in each layer is needed to fulfill the numerical convergence. The number of the collocation points is doubled as 20,000, which is randomly chosen from the original mesh setup of 600 and 100 in $x = \pm 10$ and $t = 0$ positions. From the left and middle figures of the upper and middle Fig. 1, it can be known that the numerical results are consistent with the analytical results very well. The $\times$ symbols in the left figures represent the IB training data in the two simulations that will be imported to the modified APINNs model. The relative L_2-norm errors are about 0.63% and 0.74% in the upper and middle pointwise error contour plots, which are achieved by the exact values subtracting the learned ones. Convergence with the number of collocation points is also validated, in which a maximum $x_{\max} = 40$ case with 80,000 points are loaded that consumes about 5.2GB VRAM. The $x_{\max} = 10$ simulation consumes about 2132 seconds and the $x_{\max} = 40$ case costs about 3548 seconds. The sharp decrease in x-direction is more remarkable in the $x_{\max} = 40$ case, which can lead to two orders of more simulation time in the usual finite difference method due to the fourth-order differential difference method. However, the modified APINNs model only consumes about 1.7 times more time in the current research due to the use of automatic differentiation. In the left and middle figures of Fig. 2, the comparisons of the theoretical and predicted results are analyzed in the x-direction with three diagnosis time points at $t = 0.20$, $t = 1.5$, and $t = 2.5$. The first-order solver simulations with the green dot lines are accordant with the theoretical results with the red lines, even in the situation that the values drop rapidly locally in the $x_{\max} = 40$ case.

Similar results by the modified GPINNs model can also be achieved, except that the regularization param-

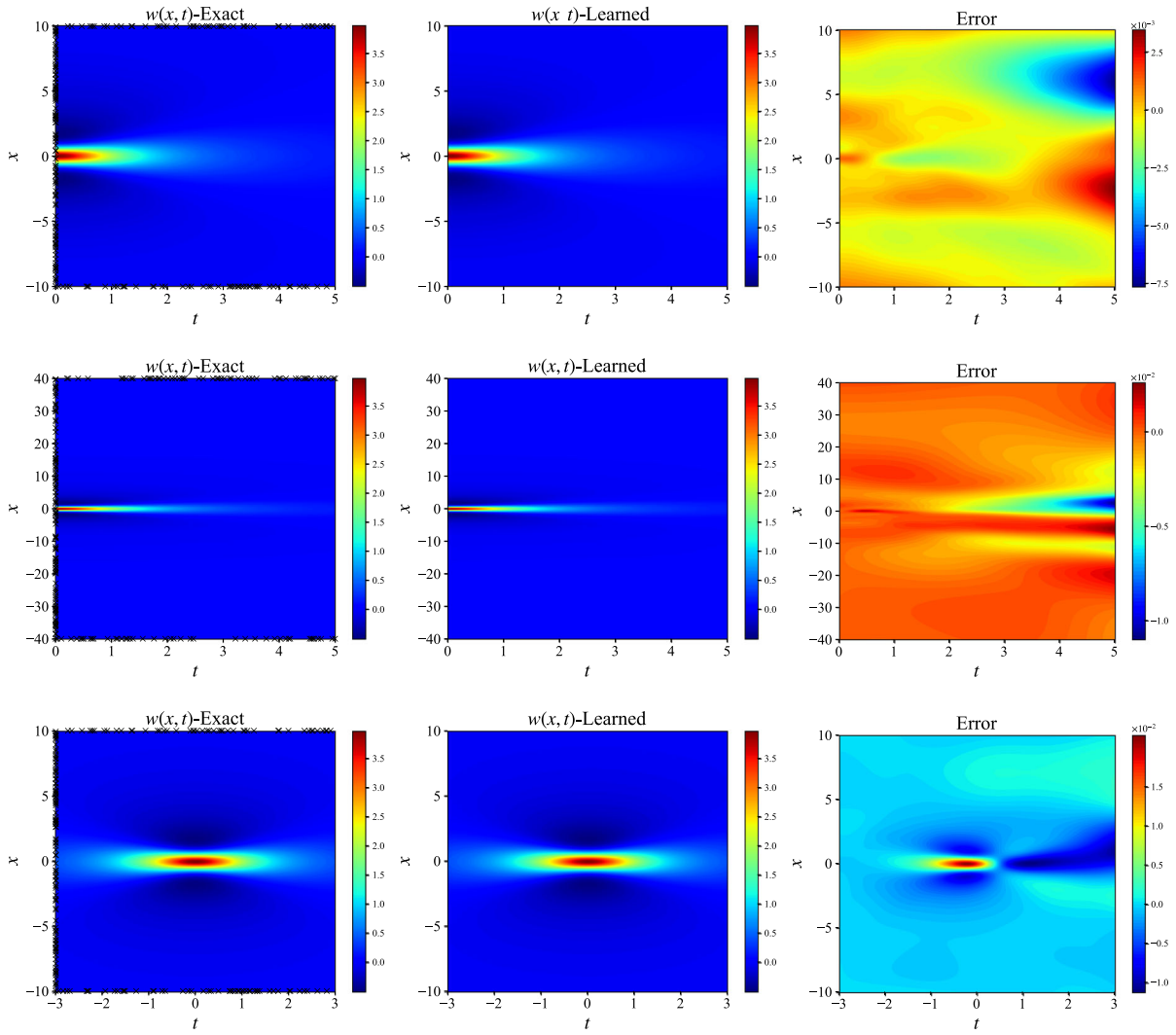

Fig. 1 The exact solutions, predicted results, and pointwise errors are presented from left to right with $t = 0$ initial conditions for $x_{\max} = 10$ (upper) and $x_{\max} = 40$ (middle) cases and $t = -3$ initial conditions for $x_{\max} = 10$ (below) case with relative L_2-norm errors of 0.63%, 0.74%, and 0.42%

eters λ_I, λ_{Ip}, and λ_B will be changed by the adaptive learning rate annealing algorithm in the learning process. The relative L_2-norm errors are about 0.56% and 0.87% for the $x_{\max} = 10$ and $x_{\max} = 40$ cases, which are on a similar level as above. Figure 3 exhibits the evolutions of the three regularization parameters in the training process and all three parameters are much greater than the unchanged values that are set up as 1 in the APINNs model as shown in Fig. 3. The regularization parameters make difference to balance the different loss terms by utilizing the back-propagated gradient statistics during model training. Parameter λ_{Ip} is larger than λ_I from Fig. 3, revealing the imbalance of the initial first-order derivative loss term is more pronounced than the initial loss term at $t = 0$. The deep neural networks are configured as 5 hidden layers with 100 and 300 neurons per hidden layer for the $x_{\max} = 10$ and $x_{\max} = 40$ cases, respectively. In these two cases, the GPU-based TensorFlow deep learning plus the CPU calculation constitute heterogeneous computing, which can give one-order speedup compared with single CPU computing. The computing

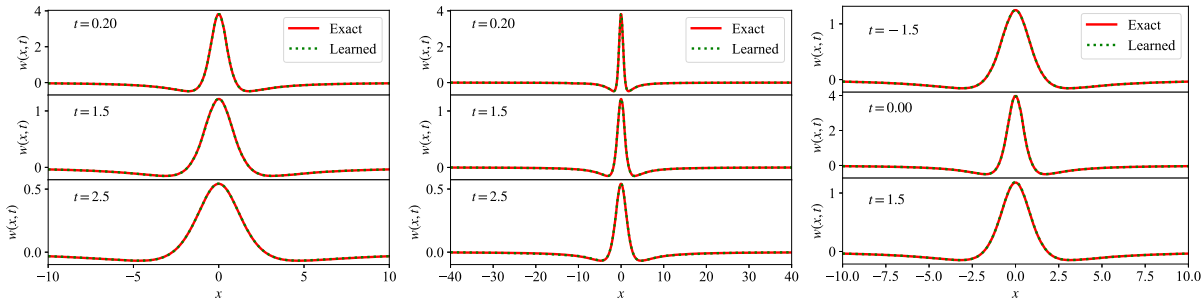

Fig. 2 Comparisons of the theoretical and the predicted results at $t = 0.2$, $t = 1.5$, and $t = 2.5$ diagnosis points for $x_{\max} = 10$ (left) and $x_{\max} = 40$ (middle) cases with $t = 0$ initial conditions and at $t = -1.5$, $t = 0.00$, and $t = 1.5$ diagnosis points with $t = -3$ initial condition (below)

time is about 4076 and 4861 seconds for the $x_{\max} = 10$ and $x_{\max} = 40$ cases with 80, 000 iterations. The values of N_I, N_{Ip}, N_B, and N_w are all increased to 512 in the $x_{\max} = 40$ case. The largest neural network architecture tested includes 7 hidden layers with 600 neurons per hidden layer containing about 2.16 million weight parameters, which gives a relative L_2-norm error of 0.91% with $N_w = 1024$. It should be noted that more neurons in the neural network architecture need more collocation points in the simulation domain to keep the similar relative L_2-norm error in the GPINNs model. The large simulative parameter range exhibits the robustness of the GPINNs model.

For the $t = -3$ initial value simulation, the neural network architecture includes 4 hidden layers with 40 neurons in each one in sequence. The APINNs model gives correct prediction with relatively small L_2-norm error, while the GPINNs cannot work currently. The left and middle figures in Fig. 1 give the exact and learned results, which are in good agreement with the two contour plots. The symmetries of the first-order solver about $x = 0.0$ and $t = 0.0$ can also be clearly shown from the predicted results. The L_2-norm error is about 0.42% calculated by the pointwise error values in the below right one of Fig. 1. The results are obtained by the learning iterations of the Adam optimizer followed by the L-BFGS algorithm. Again the $\times$ symbols in the left figure represent the IB training data to be used by the modified APINN model to constitute the loss terms. The learned green dot results are consistent with the theoretical red line results at $t = -1.5$, $t = 0.00$, and $t = 1.5$ diagnosis time points in specific values from the below right one of Fig. 2.

Then, we give a scan of the relative L_2-norm errors with respect to the number of hidden layers and the neurons per hidden layer. The numerical comparisons of APINNs and GPINNs models are carried out in the domains of $x \in [-5, 5]$ and $t \in [0, 3]$ as the above first-order case. The number of IB points is the same in the two models, while the number of the collocation points and the learning rates are different due to the different learning algorithms. The number of iterations is also different due to the aforementioned L-BFGS algorithm being terminated by its tolerance setting. Table 1 gives the comparisons of the relative L_2-norm errors with regard to the same neural network configurations in the two models. The GPINNs model utilizes the non-fixed random seed to select the IB and collocation points during the learning process, while the APINNs model takes advantage of the fixed one. So, the mean relative L_2-norm errors and their standard deviations are calculated for the GPINNs model by varying the number of hidden layers and different numbers of neurons per layer over 8 independent trials. For the same computing platform, the relative L_2-norm errors are the same when all the parameters in the APINNs model are chosen. For the GPINNs model, the prediction accuracy improves as the increase of neurons or the hidden layers. Increasing the neurons of the hidden layer can enhance the accuracy of the predictions more effectively than increasing the depth of the network. For the APINNs model, the relation between the relative L_2-norm errors and the model parameters is not obvious. Generally speaking, both models can obtain a high accuracy for the first order of Boussinesq with the above simulation domain. The GPINNs model is more robust than the APINNs model in these simulations due to the large simulative

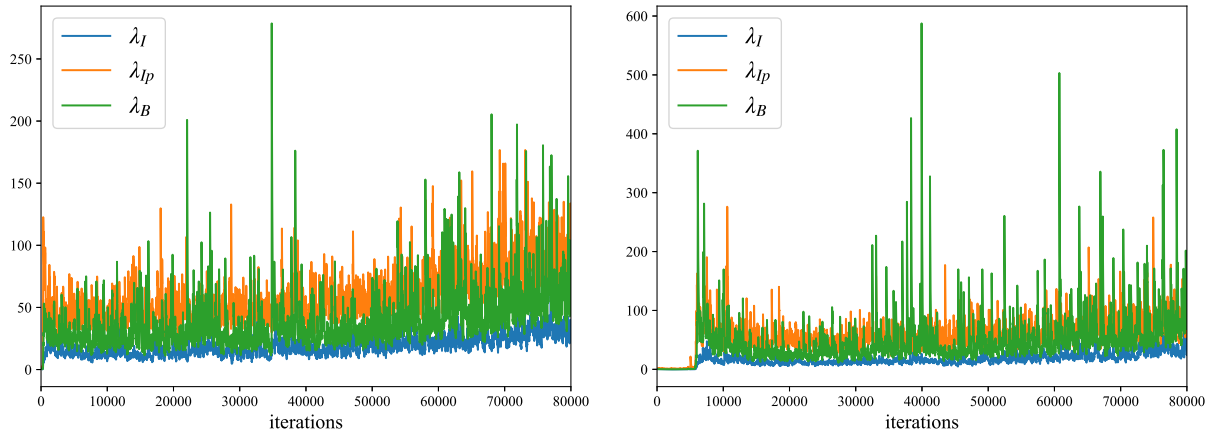

Fig. 3 Evolution of three regularization parameters λ_I, λ_{Ip}, and λ_B in the training process to balance the different loss terms for $x_{\max} = 10$ (left) and $x_{\max} = 40$ (right) cases with $t = 0$ initial conditions

Table 1 The comparison of the relative L_2-norm errors for the GPINNs and APINNs models for the first-order solver

Architecture	GPINNs	APINNs
30 units/3 hidden layers	2.03E−02 ± 3.51E−03	1.27E−02
50 units/3 hidden layers	8.80E−03 ± 6.99E−04	5.62E−03
100 units/3 hidden layers	4.42E−03 ± 6.37E−04	3.79E−03
30 units/5 hidden layers	1.43E−02 ± 2.18E−03	2.80E−03
50 units/5 hidden layers	6.01E−03 ± 8.49E−04	1.06E−02
100 units/5 hidden layers	2.56E−03 ± 4.43E−04	3.04E−03
30 units/7 hidden layers	1.36E−02 ± 2.23E−03	2.19E−03
50 units/7 hidden layers	3.81E−03 ± 8.77E−04	2.95E−03
100 units/7 hidden layers	2.45E−03 ± 4.49E−04	3.17E−03

parameter range capability similarly validated as the above $x_{\max} = 40$ case.

3.2 Second-order rogue wave

The research of the second-order rogue wave solver of the Boussinesq equation is then carried out to verify the versatile application of the modified APINNs model. The first simulation is conducted with a spatial and temporal domain of $t \in [0, 5]$ and $x \in [-5, 5]$. The IB conditions are given in Eq. (11), which provides the IB training data.

$$\begin{cases} I(x) = \frac{4A(0,x)}{B(0,x)}, & x \in [-5, 5], \\ Ip(x) = 0, & x \in [-5, 5], \\ w[t, -10] = w[t, 10] = \frac{4A(t,10)}{B(t,10)}, & t \in [0, 5]. \end{cases} \tag{11}$$

For the second-order rogue wave, the neural network contains 5 hidden layers and 100 neurons are included in each hidden layer. A maximum neuron number setup with 7 hidden layers with 100 neurons in each layer is computed to check the simulation convergence. From the left and middle figures of upper Fig. 4, it can be known that the simulations are in consistency with the analytical results very well. The $\times$ symbols in the left figure represent the IB training data in the simulation that will be imported to the modified APINNs model to minimize the penalty loss function terms. The right figure of upper Fig. 4 gives the contour plot of point-wise error by the way of the exact function subtracting the learned function, which gives an L_2-norm error of about 0.33%. Comparison of the exact and learned results is exhibited in the x-direction with three diagnosis time points at $t = 0.20$, $t = 1.5$, and $t = 2.5$ in Fig. 5. The fitting of the second-order solver is also

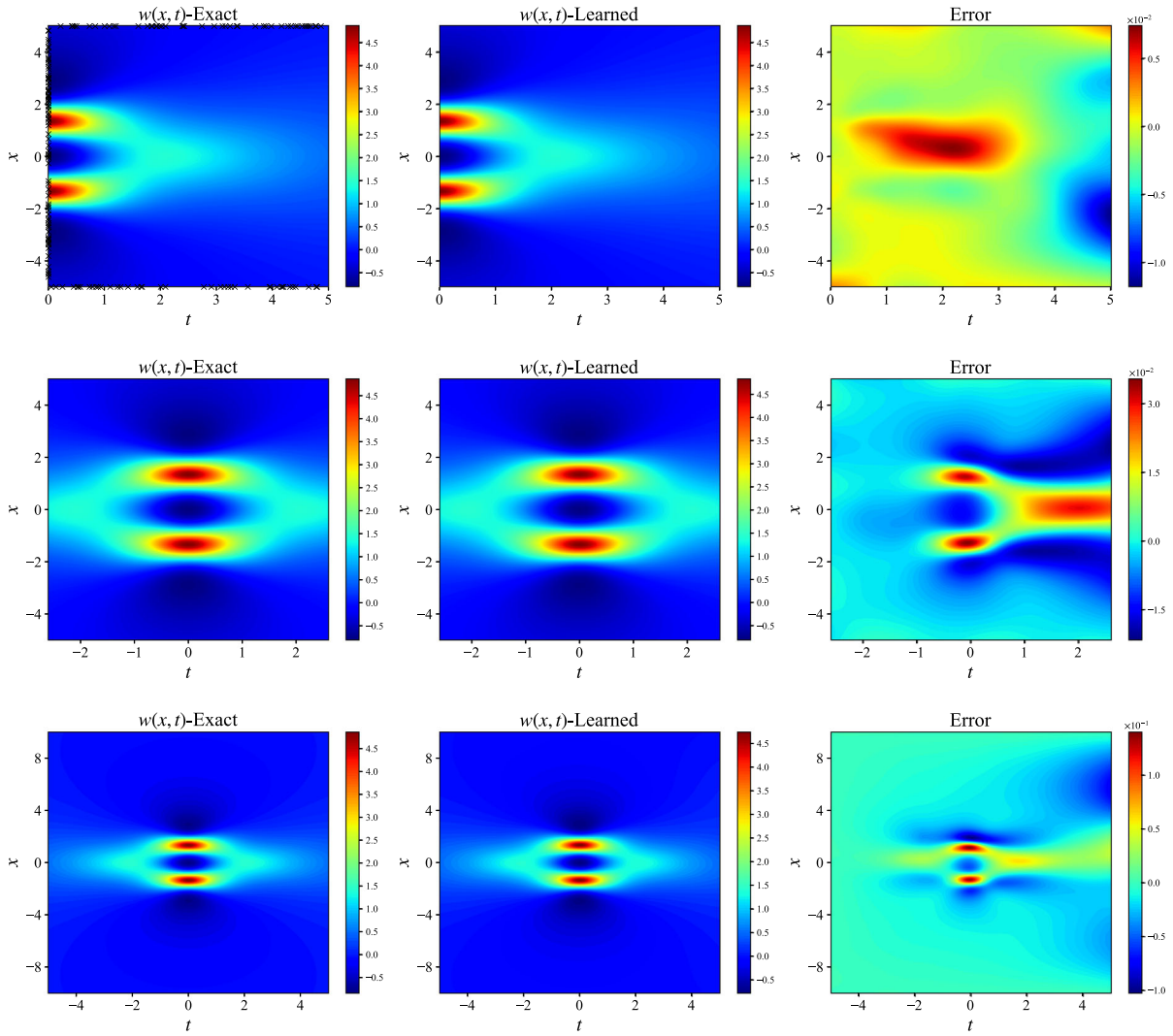

Fig. 4 The exact solutions, predicted results, and pointwise errors are presented from left to right with $t = 0$ initial conditions (upper) and $t = -1.7$ initial conditions for $x_{\max} = 5$ (middle) and $x_{\max} = 10$ (below) cases with relative L_2-norm errors of 0.33%, 0.75%, and 3.14%

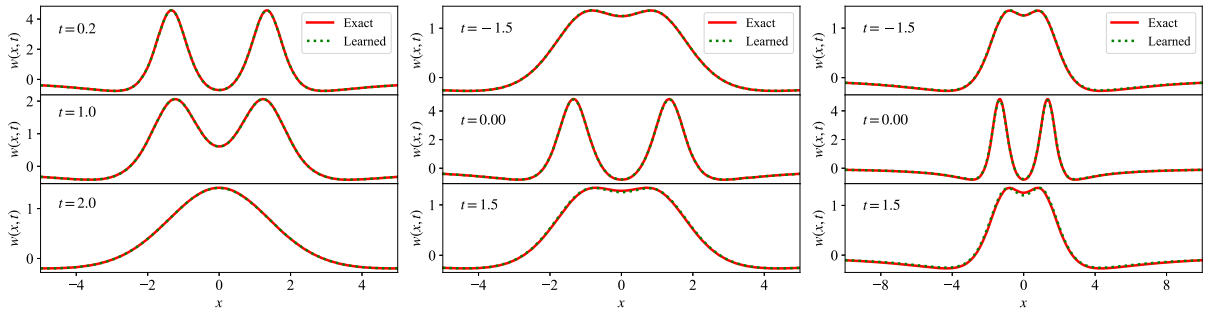

Fig. 5 Comparison of the theoretical and the predicted results at $t = 0.2$, $t = 1.0$, and $t = 2.0$ diagnosis points and at $t = -1.5$, $t = 0.00$, and $t = 1.5$ diagnosis points for $x_{\max} = 5$ (upper) and $x_{\max} = 10$ (below) cases

excellent from the exact red line result and the green dot prediction result in specific values. The transition from double crests to one crest is acquired by the simulation, which is in agreement with the theory.

For the modified GPINNs model, reliable results can also be implemented with the regularization parameters λ_I, λ_{Ip}, and λ_B varying during the training iterations. The computing time is about 6135 seconds with 80, 000 iterations with the relative L_2-norm error of about 0.31% similar to the above APINNs result. From Fig. 6, the three regularization parameters in the learning process are also much larger than 1 that is fixed in the APINNs model. The fluctuations of the three parameters in the training iterations are weaker than those in the $x_{\max} = 40$ case in Fig. 3 due to the smaller simulation domains. Parameter λ_{Ip} is larger than λ_I, revealing the imbalance of the loss term about the first-order derivative about initial t is more pronounced than the loss term about initial t as the above first-order case. The deep neural network contains 7 hidden layers with 300 neurons per hidden layer. The deep learning by the GPU-based TensorFlow can give at least one-order speedup compared with the CPU-based TensorFlow for the model with a large number of weight parameters.

Initial training data with a minus time position are also carried out for the second-order rogue wave solver analysis with two different IB training data in the modified APINNs model. The selection of the initial time position is key to the numerical convergence of the total loss function due to the contribution of the IB training data-related loss terms to the total loss terms. The former research has shown that training data in the localized sharp areas are easier to fulfill the prediction accuracy [33]. The current studies find that numerical solvers could be convergent by the Adam optimizer plus L-BFGS algorithm method for the initial point in the section of $-1.7 \leq t < 0$. The $t = -1.7$ related initial value simulations are executed with the neural network including 4 hidden layers and 70 neurons in each layer for both $t \in [-2.5, 2.5]$, $x \in [-5, 5]$(2S) and $t \in [-5, 5]$, $x \in [-10, 10]$(2B) cases. The IB conditions are given in Eqs. (12) and (13), respectively.

$$\begin{cases} I(x) = \frac{4A(t_0,x)}{B(t_0,x)}, & x \in [-5, 5], \\ Ip(x) = \frac{4[A'(t_0,x)B(t_0,x) - A(t_0,x)B'(t_0,x)]}{B^2(t_0,x)}, & x \in [-5, 5], \\ w[t, -5] = w[t, 5] = \frac{4A(t,5)}{B(t,5)}, & t \in [-2.5, 2.5], \end{cases} \tag{12}$$

$$\begin{cases} I(x) = \frac{4A(t_0,x)}{B(t_0,x)}, & x \in [-10, 10], \\ Ip(x) = \frac{4[A'(t_0,x)B(t_0,x) - A(t_0,x)B'(t_0,x)]}{B^2(t_0,x)}, & x \in [-10, 10], \\ w[t, -10] = w[t, 10] = \frac{4A(t,10)}{B(t,10)}, & t \in [-5, 5], \end{cases} \tag{13}$$

where $t_0 = -1.7$ and A' and B' represent derivate with respect to t which will be solved by the symbolic computation method in the modified APINNs model. The current 2S and 2B simulations take about 2335 and 2634 seconds, respectively. Comparing the two simulations, only a little more time is required when the simulation areas enlarge both two times in space and time domains. For the common finite difference method, it would increase the computing time by an order of magnitude at least as the aforementioned first-order solver description. The learned results are also in agreement with the exact values from the two contour plots in middle and below Fig. 4 in the two simulations. It should be noted that the training process is bidirectional in the time domain to show the symmetries. In the 2S simulation, the symmetries are excellent in both space and time domains. The symmetry is destroyed only a little in the time domain in the 2L case due to the enlarged simulation domains. Larger simulation areas may lead to a litter greater L_2-norm errors in the APINNs model as shown in another paper [56]. The L_2-norm errors are about 0.75% and 3.14% in the right middle and below pointwise error contour plots from Fig. 4. The largest error comparisons of the learned red line results and the theoretical green dot results at $t = -1.5$, $t = 0.00$, and $t = 1.5$ diagnosis time points are shown in Fig. 5 in specific values for the second-order Boussinesq solver in the two simulations. Both simulations give good consistency, with the 2L case only showing a very little discrepancy at $t = 1.5$.

A scan of the relative L_2-norm errors is also carried out with respect to the neural network architecture in the APINNs and GPINNs models. The neural networks contain more neurons in the training process due to the larger maximum results to be learned compared with the first-order case, which can affect the learning values around 0. The simulations are conducted in the domain of $x \in [-5, 5]$ and $t \in [0, 3]$ that the nonzero information is mostly distributed. The number of IB points is the same, while the number of collocation points, iterations, and learning rates are different which is the same principle as the aforementioned first-order case. Table 2 gives the comparisons of the relative L_2-norm errors with regard to the same neu-

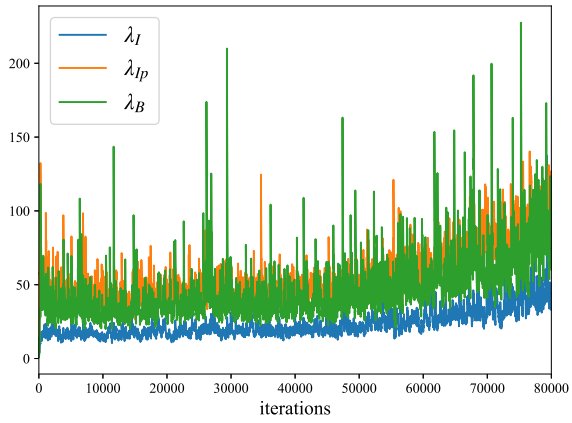

Fig. 6 Evolution of three regularization parameters λ_I, λ_{Ip}, and λ_B in the training process to balance the different loss terms for $x_{\max} = 10$ with $t = 0$ initial conditions

ral network configurations. The mean relative L_2-norm errors and their standard deviations are calculated for the GPINNs model by varying the number of hidden layers and neurons per layer over 8 independent trials due to the non-fixed random seed. The relative L_2-norm errors are the same when all the parameters in the APINNs model are chosen for the same computing platform as the above first-order simulations. For the GPINNs model, the prediction accuracy improves as the increase in the number of weight parameters that is decided by the neural network. For the APINNs model, the relation between the relative L_2-norm errors and the prediction accuracy is not obvious. The GPINNs model is more robust than the APINNs algorithm in the situations where the IB conditions are distributed in the localized sharp areas due to the large simulative parameter range capability, which is identically depicted as the first-order $x_{\max} = 40$ case. Heterogeneous parallel computing is implemented for the 8 independent trials in each neural network architecture to fully make use of the computing resources. The supercomputing system contains 4 nodes, with each containing two K80 GPUs and two Xeon CPUs. MPI-based distributed computing utilizing the MPI_Split method is made the best use of to split the original communication domains containing 8 ranks into two new communication domains with each consisting of 4 ranks to utilize all the computing resources. For the small-scale simulations with weight parameters smaller than 300, 000 and collocation points smaller 512, two jobs can be simultaneously assigned in each new rank to fully take advantage of the GPU resources.

4 Conclusions and discussion

The modified APINNs and GPINNs models with a new generalized loss term are utilized to obtain the first- and second-order solver of the Boussinesq equation with different IB conditions. For the GPINNs model, a new regularization parameter is taken into account to specify the imbalance of the initial first-order derivate loss term with the other terms. High prediction accuracies are accomplished in the large domain simulations by applying more collocation points and more weight parameters. The number of the weight parameters ranging from about 160, 000 with 5 hidden layers and 200 neurons in each layer to about 2.16 million neurons with 7 hidden layers and 600 neurons in each layer is validated to realize the prediction with small relative L_2-norm errors for the first-order $x_{\max} = 40$ case.

Table 2 Comparison of the relative L_2-norm errors for the GPINNs and APINNs models for the second-order solver

Architecture	GPINNs	APINNs
100 units/3 hidden layers	1.66E−02 ± 2.36E−03	3.82E−03
150 units/3 hidden layers	1.31E−02 ± 2.98E−03	4.38E−03
200 units/3 hidden layers	5.14E−03 ± 7.47E−04	1.72E−03
100 units/5 hidden layers	9.03E−03 ± 6.51E−04	4.73E−03
150 units/5 hidden layers	4.01E−03 ± 4.68E−04	3.84E−03
200 units/5 hidden layers	3.26E−03 ± 2.10E−04	2.04E−03
100 units/7 hidden layers	8.34E−03 ± 9.42E−04	8.69E−03
150 units/7 hidden layers	6.72E−03 ± 8.35E−04	3.59E−02
200 units/7 hidden layers	3.65E−03 ± 3.83E−04	4.20E−03

More simulations in different IB conditions with high prediction can be implemented by the APINNs model, while the GPINNs model is more robust where the initial conditions are distributed in the localized sharp areas due to the capability of a large parameter range. Larger simulation domain and higher-order differential form lead to more weight parameters and more simulation time compared with the former GPINNs models [17,18]. GPU-based TensorFlow is utilized to speed up the simulations, bringing $10-20$ times faster computing speed in the current platform compared with CPU-based TensorFlow for the large model parameters. For the GPINNs model, parallel computing is carried out over 8 independent trials efficiently to obtain the mean relative L_2-norm errors and their standard deviations due to the random choosing of the simulation points during the iterations. MPI-based distributed computing accompanying the MPI_Split method takes full advantage of high-performance computing.

The understanding of the PINNs algorithm for solving the NPDEs still needs further studies despite the great progress in recent years. The general way of loss function construction with different IB conditions should be paid attention to and understood with a solid theoretical foundation. Domain decomposition techniques [13–15] should be an alternative method to carry out large domain simulations with the PINNs model. The loss functions in these researches contain constant regularization parameters in the loss terms, which may better combine the back-propagated gradient statistics method to optimize the parameters in the training process. GPINNs model [17] plus the cPINNs model [14] should constitute a new model to set up domain decomposition training with balanced loss terms utilizing the gradient statistics method in future work. Painlevé integrable fifth-order equation with third-order temporal dispersion [34] in integrable systems could be studied to extend the framework of the current PINNs work. The comparison of the NPDEs solver between the numerical simulation by the PINNs model and the symbolic analytical computation by the BNNM or BRNM models should be interesting to verify the advantages of each other. An accelerated computing platform should be incorporated with software development to improve simulation efficiency. The current GPINNs model simulations have been upgraded and optimized to support the Python 3.9 and Tensorflow_GPU-2.x software on the Linux and Windows systems. Different software configurations with different computing platforms bring about consistent results, which is in favor of the new high-end GPU deployment to greatly accelerate deep learning computing. Heterogeneous parallel computing would make difference in the further development of PINNs model.

Funding This work was supported by the Scientific Research Fund of Zhejiang Provincial Education Department under Grant No. Y202148297 and the National Natural Science Foundation of China under Grant No. 11975008.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

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

Ethical approval This research does not involve human participants and/or animals.

References

1. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017)
3. Voulodimos, A., Doulamis, N., Doulamis, A., Eftychios, P., et al.: Deep learning for computer vision: a brief review. Comput. Intel. Neurosci. (2018)
4. Hirschberg, J., Manning, C.D.: Advances in natural language processing. Science **349**(6245), 261–266 (2015)
5. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neur. Netw. **9**(5), 987–1000 (1998)
6. Han, J., Jentzen, A., et al.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Commun. Math. Stat. **5**(4), 349–380 (2017)
7. Rudy, S.H., Brunton, S.L., et al.: Data-driven discovery of partial differential equations. Sci. Adv. **3**, 4 (2017)
8. Sirignano, J., Spiliopoulos, K.: Dgm: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. **375**, 1339–1364 (2018)
9. Raissi, M., Perdikaris, P., 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. **378**, 686–707 (2019)
10. Jagtap, A.D., Kawaguchi, K., Karniadakis, G.E.: Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. J. Comput. Phys. **404**, 109136 (2020)

11. Jagtap, A.D., Kawaguchi, K., Karniadakis, G.E.: Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. Proc. R. Soc. A **476**(2239), 20200334 (2020)
12. Jagtap, A.D., Shin, Y., Kawaguchi, K., Karniadakis, G.E.: Deep kronecker neural networks: a general framework for neural networks with adaptive activation functions. Neurocomputing **468**, 165–180 (2022)
13. Jagtap, A.D., Karniadakis, G.E.: Extended physics-informed neural networks (xpinns): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. In: AAAI Spring Symposium: MLPS, pp. 2002–2041, (2021)
14. Jagtap, A.D., Kharazmi, E., Karniadakis, G.E.: Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. Comput. Method Appl. Mech. Eng. **365**, 113028 (2020)
15. Shukla, K., Jagtap, A.D., Karniadakis, G.E.: Parallel physics-informed neural networks via domain decomposition. J. Comput. Phys. **447**, 110683 (2021)
16. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256. JMLR Workshop and Conference Proceedings, (2010)
17. Wang, S., Teng, Y., Perdikaris, P.: Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM J. Sci. Comput. **43**(5), A3055–A3081 (2021)
18. Li, J., Chen, J., 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)
19. Wang, S., Xinling, Yu., Perdikaris, P.: When and why pinns fail to train: a neural tangent kernel perspective. J. Comput. Phys. **449**, 110768 (2022)
20. Yu, J., Lu, L., Meng, X., Karniadakis, G.E.: Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. Comput. Methods Appl. Mech. Eng. **393**, 114823 (2022)
21. Li, J., Chen, Y.: Solving second-order nonlinear evolution partial differential equations using deep learning. Commun. Theor. Phys. **72**(10), 105005 (2020)
22. Li, J., Chen, Y.: A deep learning method for solving third-order nonlinear evolution equations. Commun. Theor. Phys. **72**(11), 115003 (2020)
23. Juncai, P., 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)
24. Zhang, R.-F., Bilige, S.: Bilinear neural network method to obtain the exact analytical solutions of nonlinear partial differential equations and its application to p-gbkp equation. Nonlinear Dyn. **95**, 3041–3048 (2019)
25. Zhang, R.-F., Li, M.-C.: Bilinear residual network method for solving the exactly explicit solutions of nonlinear evolution equations. Nonlinear Dyn. **108**(1), 521–531 (2022)
26. Zhang, R.-F., Li, M.-C., Gan, J.-Y., Li, Q., Lan, Z.-Z.: Novel trial functions and rogue waves of generalized breaking soliton equation via bilinear neural network method. Chaos Solit. Fract. **154**, 111692 (2022)
27. Zhang, R.-F., Li, M.-C., Albishari, M., Zheng, F.-C., Lan, Z.-Z.: Generalized lump solutions, classical lump solutions and rogue waves of the (2+ 1)-dimensional caudrey-dodd-gibbon-kotera-sawada-like equation. Appl. Math. Comput. **403**, 126201 (2021)
28. Zhang, R.-F., Li, M.-C., Cherraf, A., Vadyala, S.R.: The interference wave and the bright and dark soliton for two integro-differential equation by using bnnm. Nonlinear Dyn. **111**(9), 8637–8646 (2023)
29. Zhang, R.-F., Li, M.-C., Yin, H.-M.: Rogue wave solutions and the bright and dark solitons of the (3+ 1)-dimensional jimbo-miwa equation. Nonlinear Dyn. **103**, 1071–1079 (2021)
30. Zhang, R.-F., Bilige, S., Liu, J.-G., Li, M.: Bright-dark solitons and interaction phenomenon for p-gbkp equation by using bilinear neural network method. Phys. Scrip. **96**(2), 025224 (2020)
31. Ankiewicz, A., Akhmediev, N.: Rogue wave-type solutions of the mkdv equation and their relation to known nlse rogue wave solutions. Nonlinear Dyn. **91**, 1931–1938 (2018)
32. El-Tantawy, S.A., Alharbey, R.A., Salas, A.H.: Novel approximate analytical and numerical cylindrical rogue wave and breathers solutions: an application to electronegative plasma. Chaos Solit. Fract. **155**, 111776 (2022)
33. Li, J., Li, B.: Mix-training physics-informed neural networks for the rogue waves of nonlinear schrödinger equation. Chaos Solit. Fract. **164**, 112712 (2022)
34. Wazwaz, A.-M.: New (3+ 1)-dimensional painlevé integrable fifth-order equation with third-order temporal dispersion. Nonlinear Dyn. **106**(1), 891–897 (2021)
35. Wazwaz, A.-M.: Painlevé integrability and lump solutions for two extended (3+ 1)-and (2+ 1)-dimensional kadomtsev-petviashvili equations. Nonlinear Dyn. **111**(4), 3623–3632 (2023)
36. Wazwaz, A.-M., Albalawi, W., El-Tantawy, S.A.: Optical envelope soliton solutions for coupled nonlinear schrödinger equations applicable to high birefringence fibers. Optik **255**, 168673 (2022)
37. Kaur, L., Wazwaz, A.-M.: Optical soliton solutions of variable coefficient biswas-milovic (bm) model comprising kerr law and damping effect. Optik **266**, 169617 (2022)
38. Boussinesq, J.: Théorie de l'intumescence liquide appelée onde solitaire ou de translation se propageant dans un canal rectangulaire. CR Acad. Sci. Paris **72**(755–759), 1871 (1871)
39. Boussinesq, J.: Théorie des ondes et des remous qui se propagent le long d'un canal rectangulaire horizontal, en communiquant au liquide contenu dans ce canal des vitesses sensiblement pareilles de la surface au fond. Journal de mathématiques pures et appliquées **17**, 55–108 (1872)
40. Ursell, F.: The long-wave paradox in the theory of gravity waves. Math. Proc. Camb. Philos. Soc. **49**, 685–694 (1953)
41. Whitham, G.B.: Linear and Nonlinear Waves, p. 651. Wiley-Interscience, New York (1974)
42. Ablowitz, M. J., Ablowitz, M.A., Clarkson, P.A.: Solitons, Nonlinear Evolution Equations and Inverse Scattering, vol. 149. Cambridge University Press (1991)

43. Ablowitz, M.J., Haberman, R.: Resonantly coupled nonlinear evolution equations. J. Math. Phys. **16**(11), 2301–2305 (1975)
44. Ablowitz, M.J., Segur, H.: Solitons and the Inverse Scattering Transform. SIAM, (1981)
45. Toda, M.: Studies of a non-linear lattice. Phys. Rep. **18**(1), 1–123 (1975)
46. Zabusky, N.J.: A synergetic approach to problems of nonlinear dispersive wave propagation and interaction. In: Nonlinear Partial Differential Equations, pp. 223–258. Elsevier (1967)
47. Zakharov, V.E.: On stochastization of one-dimensional chains of nonlinear oscillations. Sov. Phys. JETP **38**, 108–110 (1974)
48. Infeld, E., Rowlands, G.: Nonlinear Waves, Solitons and Chaos. Cambridge University Press (2000)
49. Scott, A.C.: The application of bäcklund transforms to physical problems. In: Bäcklund Transformations, the Inverse Scattering Method, Solitons, and Their Applications: NSF Research Workshop on Contact Transformations, pp. 80–105. Springer (2006)
50. Wazwaz, A.-M.: Solitons and singular solitons for a variety of Boussinesq-like equations. Ocean Eng. **53**, 1–5 (2012)
51. Wazwaz, A.-M.: Gaussian solitary waves for the logarithmic Boussinesq equation and the logarithmic regularized Boussinesq equation. Ocean Eng. **94**, 111–115 (2015)
52. Gao, J., Zhou, X., Zang, J., Chen, Q., Zhou, L.: Influence of offshore fringing reefs on infragravity period oscillations within a harbor. Ocean Eng. **158**, 286–298 (2018)
53. Yan, S., Liu, Z.Y.: Numerical model of sloshing in rectangular tank based on boussinesq-type equations. Ocean Eng. **121**, 166–173 (2016)
54. Chen, M., Niu, R., Zheng, W.: Adaptive multi-scale neural network with resnet blocks for solving partial differential equations. Nonlinear Dyn. **111**, 1–20 (2022)
55. Sana, D.: Approximating the Wave Equation Via Physics Informed Neural Networks: Various Forward and Inverse Problems (2022)
56. Fang, Y., Gang-Zhou, W., Wang, Y.-Y., Dai, C.-Q.: Data-driven femtosecond optical soliton excitations and parameters discovery of the high-order nlse using the pinn. Nonlinear Dynamics **105**(1), 603–616 (2021)