Front. Struct. Civ. Eng. 2024, 18(8): 1296–1310
https://doi.org/10.1007/s11709-024-1011-4

RESEARCH ARTICLE

An artificial neural network based deep collocation method for the solution of transient linear and nonlinear partial differential equations

Abhishek MISHRA[a], Cosmin ANITESCU[b], Pattabhi Ramaiah BUDARAPU[a*], Sundararajan NATARAJAN[c], Pandu Ranga VUNDAVILLI[a], Timon RABCZUK[b]

[a] *School of Mechanical Sciences, Indian Institute of Technology, Bhubaneswar 752050, India*
[b] *Institute of Structural Mechanics, Bauhaus University of Weimar, Weimar 99423, Germany*
[c] *Department of Mechanical Engineering, Indian Institute of Technology Madras, Chennai 600036, India*
[*] *Corresponding author. E-mail: pattabhi@iitbbs.ac.in*

ABSTRACT A combined deep machine learning (DML) and collocation based approach to solve the partial differential equations using artificial neural networks is proposed. The developed method is applied to solve problems governed by the Sine–Gordon equation (SGE), the scalar wave equation and elasto-dynamics. Two methods are studied: one is a space-time formulation and the other is a semi-discrete method based on an implicit Runge–Kutta (RK) time integration. The methodology is implemented using the Tensorflow framework and it is tested on several numerical examples. Based on the results, the relative normalized error was observed to be less than 5% in all cases.

KEYWORDS collocation method, artificial neural networks, deep machine learning, Sine–Gordon equation, transient wave equation, dynamic scalar and elasto-dynamic equation, Runge–Kutta method

1 Introduction

The biology of human brain (made-up of neurons) can be mimicked by artificial neural networks (ANNs). Neural networks mainly operate on the minimization of cost function. The major advantage of ANNs is their capability to adopt high-level open source libraries like Keras, Tensorflow, Pytorch, to perform parallel operations while solving mathematical problems. Therefore, this technique has been employed to solve several ordinary and partial differential equations (PDEs) [1–3]. Later on, ANNs is also applied to solve nonlinear PDEs of higher order, thereby addressing the problems associated with finite element and finite difference methods [4–6], as well as boundary value problems with irregular boundaries [7].

Machine learning involves training computers to recognize patterns in data and apply these algorithms to make informed decisions. In deep machine learning (DML), ANNs with extra layers (hidden layers) are used to significantly enhance their capability to deal with complex patterns. Therefore, ANNs in conjunction with classical methods can result in efficient solution techniques for various physical problems involving PDEs. A physics-based approach to the solution of a dynamic fluid-flow problem governed by a nonlinear PDE, using neural network−based machine learning scheme is discussed in Ref. [8].

With the advent of rapid developments in computational science and applications, data generation is exponentially increasing [9,10]. Therefore, machine learning is becoming mandatory to handle such big data in several applications [11–13]. Machine learning techniques to solve the Schrodinger equation using density functional theory are addressed in Ref. [14]. A first physical principles is proposed in Ref. [15], where the authors used unfiltered and scattered clinical data for training, to arrive at physically consistent predictions without the requirement of conventional simulators. Assessment of the uncertainties in soil and water is a challenging task

Article history: Received Sep 8, 2022; Accepted Apr 9, 2023

due to the significant number of parameters involved. Ghaith and Li [16] have investigated the uncertainty propagation in soil and water assessment considering the parameter sensitivity, by integrating the polynomial chaos expansion and a machine learning technique.

One main application of machine learning is to solve the governing differential equations, which are usually ordinary/PDEs. Two popular approaches to solving the governing PDEs are: 1) collocation method [17] and 2) energy minimization method [18]. In collocation method, the idea is to choose a finite-dimensional domain with a specified number of points [19] in the selected domain, known as collocation points, and train the network such that the governing equations and the boundary and initial conditions are satisfied with minimal error at the selected points. The energy minimization approach involves the formulation of governing equations in terms of energy and solve for the response by minimizing the energy. An energy−based deep neural networks (DNN) method to solve PDEs in science and engineering, in particular for mechanical applications based on the natural loss function is proposed in Ref. [18]. A physics informed neural network (PINN) model with supervised learning for solving nonlinear PDEs is introduced in Ref. [20]. A PINN model for minimizing the system energy in fracture problems by satisfying the boundary conditions exactly through appropriate modification of the network output is discussed in Ref. [21].

A combined ANN and an adaptive collocation based hybrid model for solving PDEs is introduced in Ref. [22], by employing residual based variable number of grid points. A feed-forward DNN based deep collocation method for thin plate bending problems is developed in Ref. [23]. A DNN based energy method for finite deformation hyperelasticitiy by minimizing potential energy is developed in Ref. [24]. A deep collocation method based on a feed-forward DNN to solve the heat transfer problems in porous media is introduced in Ref. [25]. Machine learning in materials science, along with its latest applications for characterization, property estimation and materials design for energy related applications, such as: solar cells, batteries, and gas capture are reviewed in Ref. [26].

ANNs can efficiently approximate the continuous functions, where Hornik et al. [27] have demonstrated that a given function $f(x)$ can be approximated within a specified tolerance using a network with a finite number of neurons and a single hidden layer using an appropriate activation function. Furthermore, Lu et al. [28] have proposed that by forming deeper networks using a nonlinear activation function, the number of neurons can be greatly reduced. Sirignano and Dgm [29] have derived different properties of neural networks for approximating PDEs. In this study, we propose a deep collocation method for the solution of transient linear and nonlinear PDEs using ANNs. The main novelties of the developed method are: 1) an ANN-based method for solving transient linear and nonlinear PDEs; 2) use of forward-mode differentiation to efficiently evaluate the derivatives of a neural network with many outputs; 3) testing the developed methodology by comparing the estimated results with the results from the implicit Runge–Kutta (RK) method.

The arrangement of the manuscript is as follows. Details of the developed methodology including the ANNs and the collocation method are explained in Section 2. Section 3 is dedicated for application of the developed methodology to solve four numerical problems, where two one dimensional (1D) problems include: Sine–Gordon equation (SGE) and wave equation and two two dimensional (2D) problems on dynamic scalar wave equation and elasto-dynamic equation. The manuscript is concluded by highlighting the main results in Section 4. Details of the explicit and implicit RK method are presented in Electronic Supplementary Material.

2 Methodology

Finite element method (FEM) has been a popular and established technique to solve PDEs in various engineering applications. Recently, due to the presence of free nodes and the absence of their connectivity, meshfree methods are particularly advantageous in modeling complex problems and geometries, such as: large deformation problems, problems involving discontinuities like cracks and interfaces. Analysis of problems involving large deformation using FEM can lead to large distortion of the elements, hence re-meshing is necessary. Re-meshing increases the computational times, apart from introducing the errors during the transfer of information from the old elements to the new elements. In this study, a combined collocation and DML approach is proposed to solve transient linear and nonlinear PDEs using ANNs (Fig. 1).

In the reproducing kernel particle method/differential reproducing kernel particle method (RKPM/DRKPM) the response of a domain discretised with meshfree nodes is estimated by solving $[\boldsymbol{K}]\{\boldsymbol{u}\} = \{\boldsymbol{f}\}$ equation, where $\boldsymbol{K}$, $\boldsymbol{f}$ and $\boldsymbol{u}$ are the stiffness, force, and displacement matrices, respectively (Fig. 1). The response is estimated by interpolating the nodal displacements with shape functions. Therefore, displacements at each node are required to be estimated, which is computationally expensive process. On the other hand, in the proposed machine learning approach, the machine is initially trained using ANNs to estimate the response, so that the process can be repeated to estimate the response at various time/load steps (Fig. 1). As a result, in the

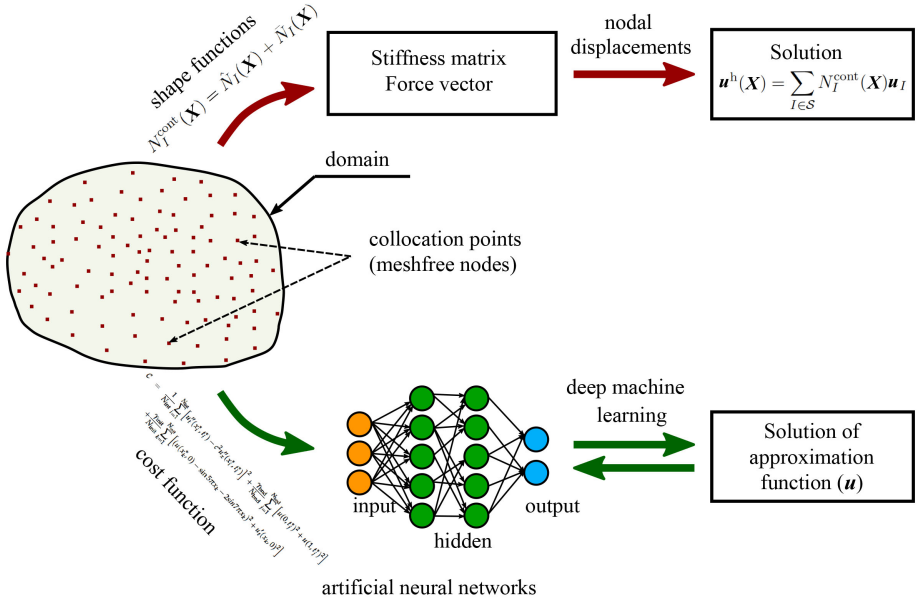


Fig. 1 Conventional and DML approaches to estimate the response of a domain discretised with meshfree/collocation points.

machine learning approach the individual nodal displacements are not required to be estimated, which significantly saves the computational costs. The notation adopted here is asfollows: $u'_x = \dfrac{\partial u}{\partial x}$, $u''_{xx} = \dfrac{\partial^2 u}{\partial x^2}$, $u'_{\text{t}} = \dfrac{\partial u}{\partial t}$ and $u''_{\text{tt}} = \dfrac{\partial^2 u}{\partial t^2}$.

2.1 Artificial neural networks

In this study, feed-forward neural network is employed by connecting the network of neurons in several adjacent column of layers made of an input layer, an arbitrary number of intermediary hidden layers, and an output layer (Fig. 2). The network is used to represent a function u: $R_n \rightarrow R_m$, using 'n' number of neurons in the input layer and 'm' number of neurons in the output layer, as shown in Fig. 2. The layers are indexed starting from the input layer at 0, and the output layer at L. The number of neurons in each layer are denoted by $k_0 = n, k_1, k_2, \ldots, k_{\text{L}} = m$. For each connection between the ith neuron in layer $(l-1)$ and the jth neuron in layer l, where $0 < l \leqslant L$, a weight w_{il}^j is associated and to each neuron in the layers $0 < l \leqslant L$, a bias b_{i} is also associated, where $i = 1,\ldots,k_l$. Moreover, an activation function is defined as σ_l: $R\text{n} \rightarrow R\text{m}$ between the layers l and $(l-1)$. Therefore, the approximation function at each neuron in the current step is expressed in terms of the activation function expressed as a linear combination of the values at neurons in the previous layer, i.e.:

$$u_k^l = \sigma^l\left(\sum_{j=1}^{k_{l-1}} w_{kj}^l u_j^{l-1} + b_k^l\right). \tag{1}$$

Equation (1) can be written in a compact matrix form as:

$$u^l = \sigma^l\left(W^l u^{l-1} + b^l\right), \text{ for } l = 1,2,\ldots,L, \tag{2}$$

where W^l indicates the weights of the connections between the layers $(l-1)$ and l, and u^l represents the vector of input parameters, b^l indicates the bias in the network and the activation function is evaluated elementwise. The choice of activation function plays a crucial role in the performance of a neural network. Since the derivatives of the outputs with respect to the inputs are required to be estimated, a smooth activation function like tanh is used for the first $(L-1)$ layers, and a linear activation is used in the last layer to ensure that the output can take arbitrary values.

Existing computational libraries like Tensorflow/ Pytorch or computational graphs available in graphical processing units [27] can perform parallel execution. To optimize the computation of output and distribute across the available computational resources, the input variables are defined as multi-dimensional arrays. However, the

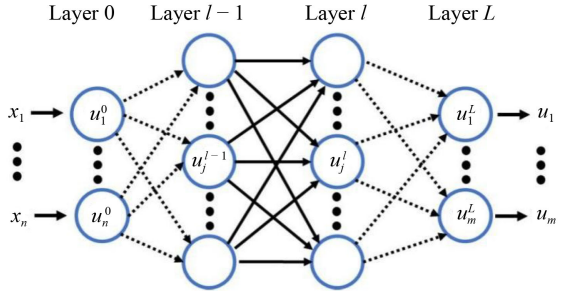


Fig. 2 A feed-forward neural network made of $L-1$ number of hidden layers.

partial derivatives of the output variables of the network with respect to the biases and weights can be efficiently estimated through the back-propagation algorithm. This is achieved by estimating the derivatives using chain rule, starting from the last layer and simultaneously storing the intermediate values in a reverse order of the layers. To minimize the loss function, the gradient-based minimization algorithms are triggered by the back-propagation algorithm. In deep learning applications, a neural network is trained on a set of known outputs with an objective to minimize the loss. This requires large data sets, which are subjected to errors. Such errors are avoided here through the minimization of the loss function. The training points are randomly generated in the domain [30]. A methodology for selecting the training data for efficient control on the performance of the optimization algorithms and the solver is explained in the following section.

The decision on neural networks architecture is based on informed trial-error considering the computational time and error. The number of layers and neurons per layer are decided based on the computational time and memory. With increase in number of layers and neurons, the error is observed to be decreasing in this study. However, 1) the decrease in error is not always guaranteed and 2) the computational time increases with increase in number of layers and neurons. In the similar lines, the activation functions and optimizer are also selected, such that the error is minimal.

2.2 Collocation method

Collocation methods are based on random nodes and the nodal data can be processed although it is unstructured. The governing PDEs can be solved by feeding the collocation points to an ANN. The ANNs are trained with appropriate network architecture to solve the associated PDEs. In this study, computer implementation of the developed algorithms to solve the numerical problems is carried out in the Python language using Tensorflow, an open-source library. The problem domain is divided uniformly to identify the locations of the collocation points (Fig. 1). The neural network is trained based on the mean square error (*MSE*). Xavier initialization method is employed to initialise the neural network. Furthermore, penalty parameters are used in some cases to reduce any specific errors, like boundary error. Relative normalized error is calculated to estimate the deviation of output from the network with respect to the target output. The major time consuming activity of this approach is finding the correct type of network architecture.

The main objective in the collocation method is to define a loss function using the governing equations and boundary conditions and estimate at various point sets within a selected domain of influence as well as on the boundary. Consider a time-dependent PDE that can be expressed as:

$$\mathcal{L}(u(x,t)) = f(x,t), \text{ for } x \in \Omega, \quad 0 \leqslant t \leqslant T, \tag{3}$$

together with boundary conditions

$$\mathcal{G}(u(x,t)) = g(x,t), \text{ for } x \in \partial\Omega, \quad 0 \leqslant t \leqslant T, \tag{4}$$

and initial conditions

$$\mathcal{I}(u(x,0)) = h(x), \text{ for } x \in \Omega, \tag{5}$$

where $\mathcal{L}$, $\mathcal{G}$, and $\mathcal{I}$ represent the interior, boundary, and initial differential operators, respectively, f, g, and h are the prescribed functions (e.g., loading data), and $\Omega \in R_{\mathrm{d}}$ indicates the domain of the problem whose boundary is denoted by $\partial\Omega$ and $[0,T]$ is the time domain. The solution u_{NN} which depends on the weights and biases as defined in Eq. (2) is obtained by defining a cost function C, as the sum of the mean squared error of losses on the boundary, interior and initial condition points:

$$\begin{aligned} C = & \frac{1}{N_{\mathrm{int}}} \sum_{i=1}^{N_{\mathrm{int}}} \left[\mathcal{L}\left(u_{\mathrm{NN}}\left(\bar{x}_i, \bar{t}_i\right)\right) - f\left(\bar{x}_i, \bar{t}_i\right) \right]^2 \\ & + \frac{\gamma_{\mathrm{bnd}}}{N_{\mathrm{bnd}}} \sum_{j=1}^{N_{\mathrm{bnd}}} \left[\mathcal{G}\left(u_{\mathrm{NN}}\left(\bar{x}_j^{\mathrm{bnd}}\right) - g\left(\bar{x}_j^{\mathrm{bnd}}, \bar{t}_j^{\mathrm{bnd}}\right)\right)\right]^2 \\ & + \frac{\gamma_{\mathrm{init}}}{N_{\mathrm{init}}} \sum_{k=1}^{N_{\mathrm{init}}} \left[\mathcal{I}\left(u_{\mathrm{NN}}\left(\bar{x}_k^{\mathrm{init}}, 0\right)\right) - h\left(\bar{x}_k^{\mathrm{init}}\right)\right]^2, \end{aligned} \tag{6}$$

where u_{NN} is the approximation function, N_{int}, N_{init}, N_{bnd} are the number of interior, initial, and boundary points, respectively, γ_{init}, γ_{bnd} are penalty parameters for initial and boundary conditions, respectively, and the quantities denoted with '¯' represent the normalized quantities, for instance, $\bar{x}_i = \dfrac{x_i}{L}$ and $\bar{t}_i = \dfrac{t_i}{T}$. The costfunction in Eq. (6) must be minimized at the selected collocation points, which is the process of "training" the network to find the solution of the given PDE.

3 Numerical examples

The developed DML approach is extended to solve the problems considering: 1) continuous and 2) discrete times. Continuous time problems are solved using the DML based collocation method and hence, we call this method as collocation method. Whereas, the discrete time problems are solved using the DML based RK method, which we denote by DML based RK method. The collocation points are created by dividing the domain uniformly in both space and time. The methodology is implemented using Tensorflow library in Python language. The *MSE* at each collocation point is calculated and

minimized to train the network. Furthermore, a novel scheme is developed here to solve the governing PDEs involving double derivative of an approximation function u with respect to time, using the DML−based RK method. Therefore, results from the DML based RK method or the exact solution are compared to the estimates from the collocation method. In this section, the developed collocation based DML approach is applied to solve the 1D and 2D problems governed by transient linear and nonlinear PDEs. We solved four problems in total: 1D wave equation, SGE and 2D dynamic scale wave equation and elastic wave equation.

3.1 Example 1: One dimensional wave equation

The wave equation in one dimension can be written as [31,32]:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \quad 0 \leqslant x \leqslant 1 \quad \text{and} \quad 0 \leqslant t \leqslant 1, \tag{7}$$

where u is the wave amplitude at position x and time t, c is a fixed nonnegative real coefficient, $c = \frac{1}{v}$ and v is the wave velocity. In the collocation method, in order to solve PDEs, the initial:

$$u(x,0) = \sin 5\pi x + 2\sin 7\pi x,\ u_t'(x,0) = 0,\ \text{for } x \in [0,1], \tag{8}$$

and boundary conditions

$$u(0,t) = u(1,t) = 0,\ \text{for } t \geqslant 0, \tag{9}$$

are specified as constraints in the PINNs. The selected domain both in time and space is given by [0,1]. The obtained solution is verified by comparing it with the exact solution of Eq. (7):

$$u_{\text{exact}}(x,y) = \sin(5\pi x)\cos(5\pi t) + 2\sin(7\pi x)\cos(7\pi t). \tag{10}$$

The cost function in Eq. (6) is minimised at the chosen collocation points, while training the network to find the solution as its output. This can be verified by comparing with it exact solution in Eq. (10).

The 1D wave equation is solved using the developed DML approach, considering a layer architecture in the network as [12,60,60], indicating an input layer with two neurons, two hidden layers containing 60 nerons each and an output layer with 1 neuron. The training time for considering the L^2 normalized error in approximation function u as 2.031188×10^{-2} is observed to be 1052.1711 s. The domain is discretized with a total number of 201 space and time collocation points. The hyperbolic tangent function is used as the activation function and the Adam optimiser combined with L-BFGS-B are employed for the optimization.

Comparison of the estimated solution using the present method and the exact solution (Eq. (10)) is shown in Fig. 3(a), where the estimated solution is found to be in perfect agreement with the exact solution.

The 1D wave equation is also solved using the RK method. However, wave equation in Eq. (7) contains a second order derivative of displacement in time. In order solve this problem using the RK method, explained in Electronic Supplementary Material, a variable v is defined as: $v = u_t' = \frac{\partial u}{\partial t}$, such that Eq. (7) can be rewritten as:

$$\frac{\partial u}{\partial t} = v, \tag{11a}$$

$$\frac{\partial v}{\partial t} = \frac{\partial^2 u}{\partial x^2},\ 0 \leqslant x \leqslant 1 \text{ and } 0 \leqslant t \leqslant 1. \tag{11b}$$

An observation of Eqs. (11) indicates the presence of first derivative with respect to time on the left hand side, which can be solved by defining U_t and F as:

$$U_t = [u_t', v_t']^{\mathrm{T}}, \tag{12a}$$

$$F = [v, u_{xx}'']^{\mathrm{T}}. \tag{12b}$$

Therefore, after substituting Eq. (12) in Eq. (11):

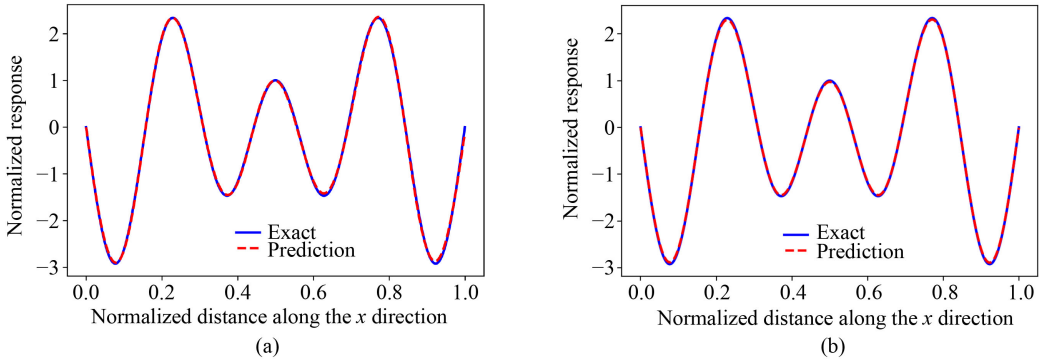


Fig. 3 Comparison of the estimated solution of 1D wave equation from: (a) the developed DML approach; (b) the RK method, with the exact solution in Eq. (10).

$$U_t = F. \tag{13}$$

Key computer implementation steps in the solution of 1D wave equation are listed in Listing 1 in Electromic Supplementary Materials. The neural network function net_f_u() accepts the input variables x and t and returns function f_u which is equal to u_xx-u_tt. The variable u_tt represents the second derivative of u with respect to time. The function f_u is later minimized during the training process.

On the other hand, key computer implementation steps in the solution of 1D wave equation using the RK method are listed in Listing 2. The neural network function net_U0() accepts the input variable x containing the locations of the internal points and returns the variables $u0$ and $v0$ estimated at the internal points, which are later used in the minimization during the training process. Whereas, the main purpose of the neural network function net_U1() is to repeat the activities of the neural network function net_U0() for the boundary values of the variable x. Therefore, the function net_U1() considers the boundary values of x as input and returns the variables $u1$ and $v1$ estimated along the boundary, which are later used during the minimization while training the network. The variable q indicates the total number of stages and IRK weights denote the weights used in the implicit RK method.

Equation (13) can be solved in the similar lines of solving the Burgers' equation, considering a layer architecture in the network as [1,50,50,50,202], indicating an input layer with 1 neurons, three hidden layers containing 50 nerons each and an output layer with 202 neurons. The training time for considering the L^2 normalised error in approximation function u as 3.216116×10^{-2} is observed to be 1207.10 s. The domain is discretised with a total number of 250 points in space and 100 collocation points in time. A comparison of the solution from the RK method with the exact solution is provided in Fig. 3(b), where the estimated solution is found to closely agree with the exact solution, with an L^2 error 3.216116×10^{-2} This error shows the efficiency of the RK method while solving the problems involving double time derivatives. A comparison of various parameters used in the DML and RK methods is provided in Table 1.

A close observation of Table 1 indicates that the number of spatial points are higher in case of the RK method. Whereas, the number of points in the time domain are slightly more than double in the collocation method as compared to the RK method. However, a comparison of the network architecture indicates that the RK method involves the usage of 3 hidden layers with 50 neurons each, whereas there are only 2 hidden layers with 60 neurons each in the collocation method (Table 1).

3.2 Example 2: Sine–Gordon equation

The SGE is a nonlinear hyperbolic PDE involving the sine of an unknown function and the d'Alembert operator. Due to the presence of soliton (self-reinforcing wave packet which maintains its shape while propagating at a constant velocity) solutions of SGE attracted a lot of attention. The SGE can be found in a number of applications [33]. The SGE in space-time coordinates (x, t) reads [34]:

$$u_{tt} - u_{xx} + \sin u = 0. \tag{14}$$

In this study, Eq. (14) is solved in the spacial domain $x \in [-30, 30]$ within a time interval of 10 s, considering the initial conditions $u(x,0) = f(x)$ and $u_t(x,0) = g(x)$. The domain is discretized with 101 collocation points along the spacial (x) direction, at 201 discrete time instances. Figure 4 shows the discretizations along the spatial (collocation points) and temporal axes. Therefore, simulations were carried out with space and time discretization steps of 0.3 and 0.05, respectively.

3.2.1 Breather solution

The breather soliton solution of SGE, also known as breather mode is a time periodic oscillatory and spatially localized solution, representing a field periodically oscillating in time with exponential decay in space with increase in the distance from the center. The breather solution of SGE is characterized by a phase which depends on the breather's evolution history. The breather solution of SGE is given by Ref. [35]:

Table 1 Comparison of parameters specified while solving the wave equation using DML approach and RK Method

No.	Description	Collocation method	RK method
1	number of space and time collocation points	201	–
2	number of point at initial and final stages	–	250
3	number of stages (q)	1	100
4	layers architecture	[2,60,60,1]	$[1,50,50,50,2 \times (q+1)]$
5	training time (seconds)	1052.1711	1207.10
6	L^2 error in u	2.031188×10^{-2}	3.216116×10^{-2}

$$u(x,t) = 4\arctan\left(\frac{\sqrt{1-\omega^2}\sin(\omega t)}{\omega\cosh\left(\sqrt{1-\omega^2}x\right)}\right). \quad (15)$$

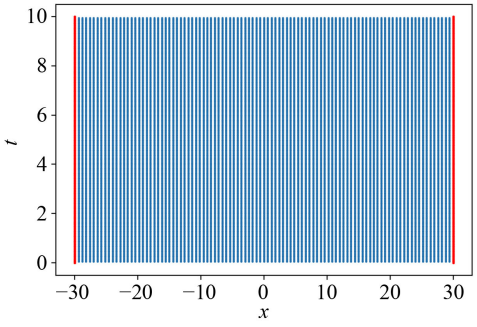


Fig. 4 Discretized spatial and temporal domains along the x and the y axes, respectively, to solve the SGE.

Key computer implementation steps to estimate the exact the breather solution of SGE are listed in Listing 3 in Electromic Supplementary Materials. The neural network function compExactDispVel (XT) estimates the response as a function of space (x) at different instances of time (t) through the variables u and v estimated using the exact solution.

On the other hand, solution steps to estimate the breather solution of SGE using the proposed collocation method are listed in Listing 4. A total domain size of 60 units was considered to estimate the breather response over 10 s. Uniform edge boundary conditions are applied at either edges. The breather response of SGE estimated as a function of time using the developed methodology is plotted in Fig. 5. The computed solution is compared to the exact solution, where they are observed to be in close agreement.

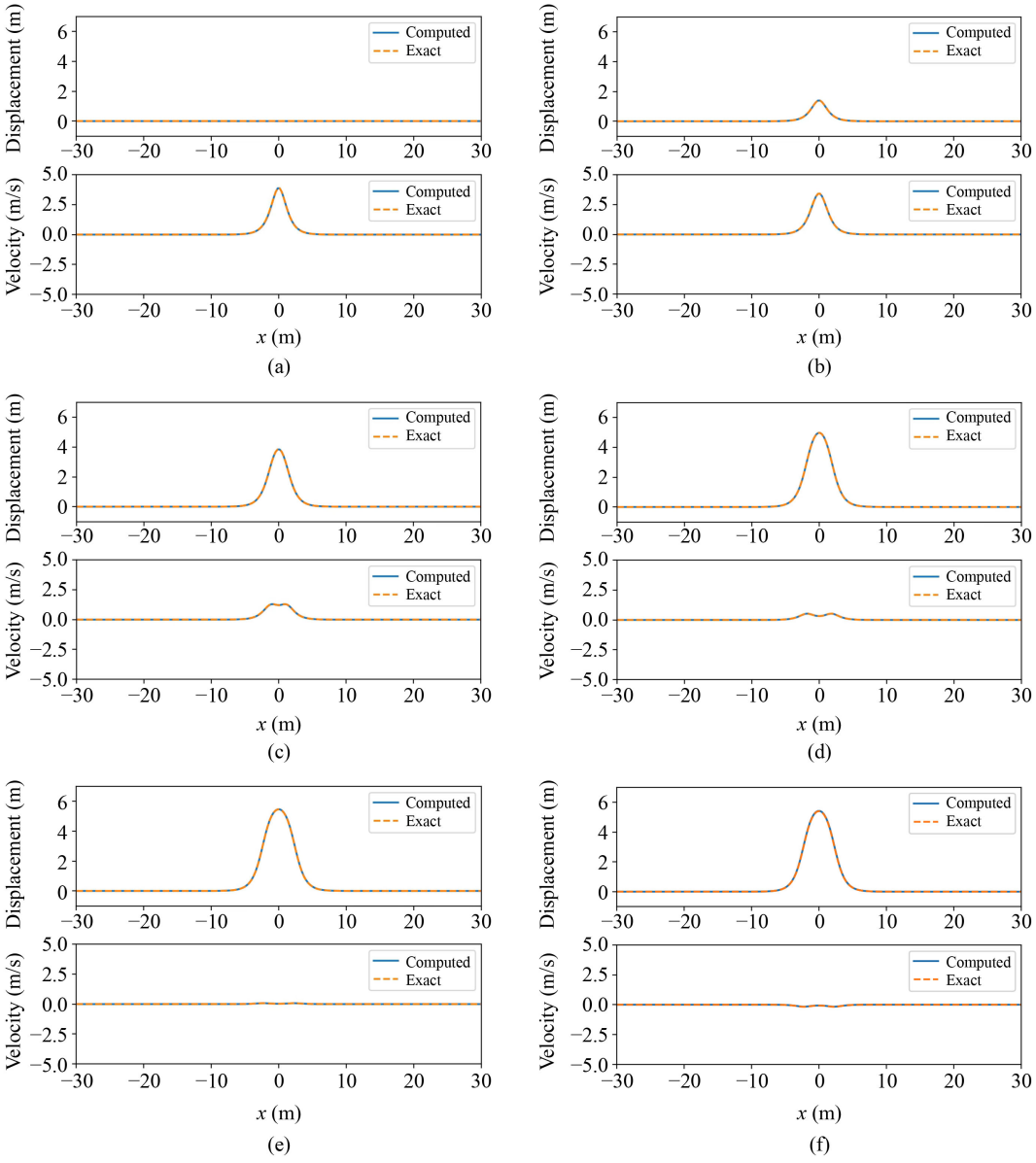


Fig. 5 Breathing solution of SGE at various time instances: (a) 0.0; (b) 0.374; (c) 1.496; (d) 3.241; (e) 6.982; (f) 9.975.

A comparison of the exact and computed solutions of breather response of SGE is provided in Fig. 6. Displacement, velocity and error in displacement estimated using the exact and computed solutions are shown in the first and second columns of Fig. 6, respectively. The maximum errors in displacement and velocity are observed to be 0.69% and 0.344%, respectively, indicating a very close agreement of the computed solution with the exact solution.

3.2.2 Kink-kink solution

The kink-kink solution of SGE is given by [35]:

$$u(x,t) = 4\arctan\left[\frac{c\sinh\left(\dfrac{x}{\sqrt{1-c^2}}\right)}{\cosh\left(\dfrac{ct}{\sqrt{1-c^2}}\right)}\right]. \tag{16}$$

The kink-kink collision response indicates that the two kinks approaches the origin from $-\infty$, collide with velocities of c and $-c$, respectively, and continue to travel away from the origin with velocities $\pm c$ as time tends to ∞. The kink-kink collision response of SGE estimated as a function of time using the developed methodology is plotted in Fig. 7. The solution is observed to start it's

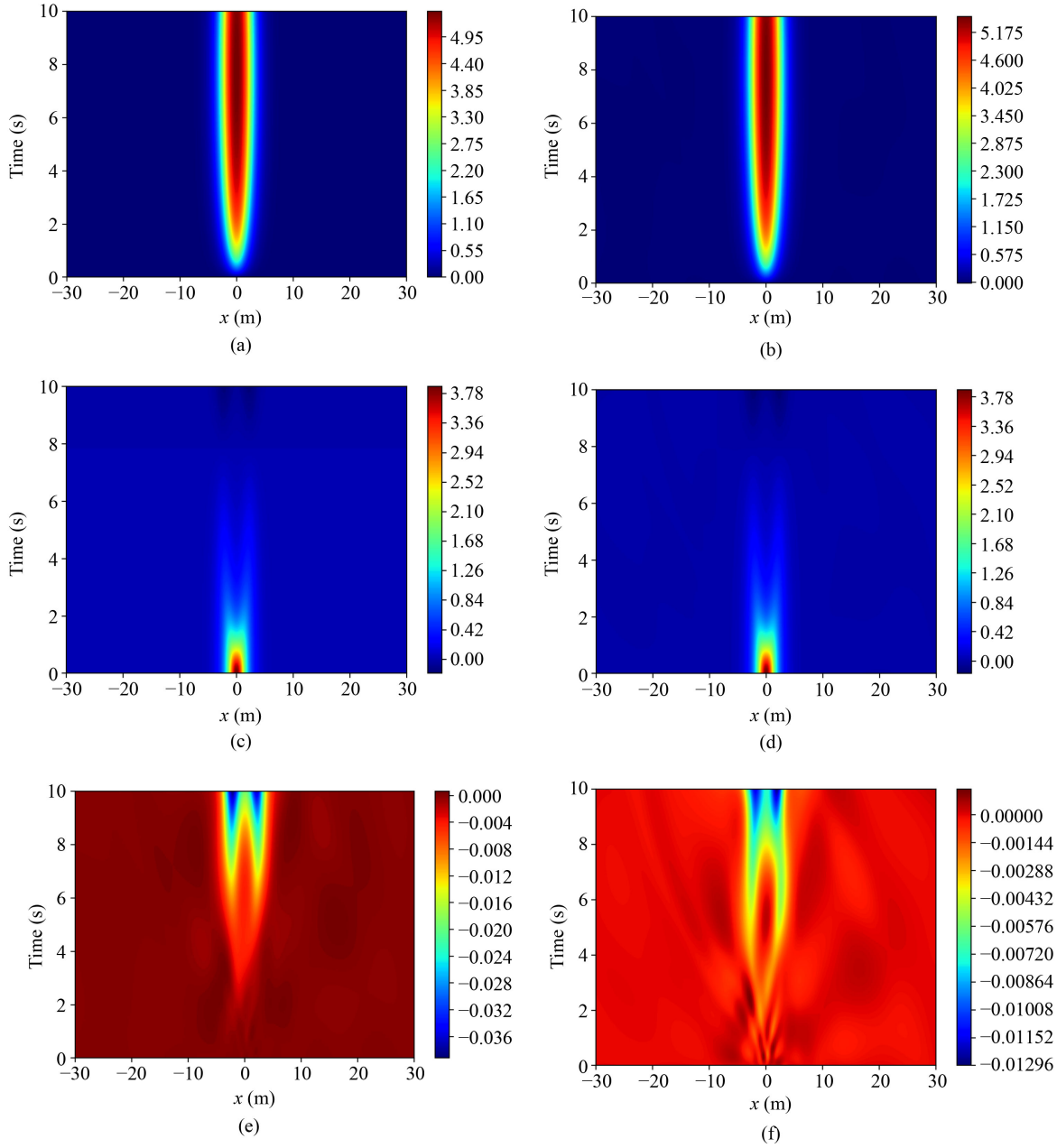


Fig. 6 A comparison of the exact and computed displacement and velocity, along with a plot of the error distribution: (a) exact displacement; (b) computed displacement; (c) exact velocity; (d) computed velocity; (e) error in displacement; (f) error in velocity.

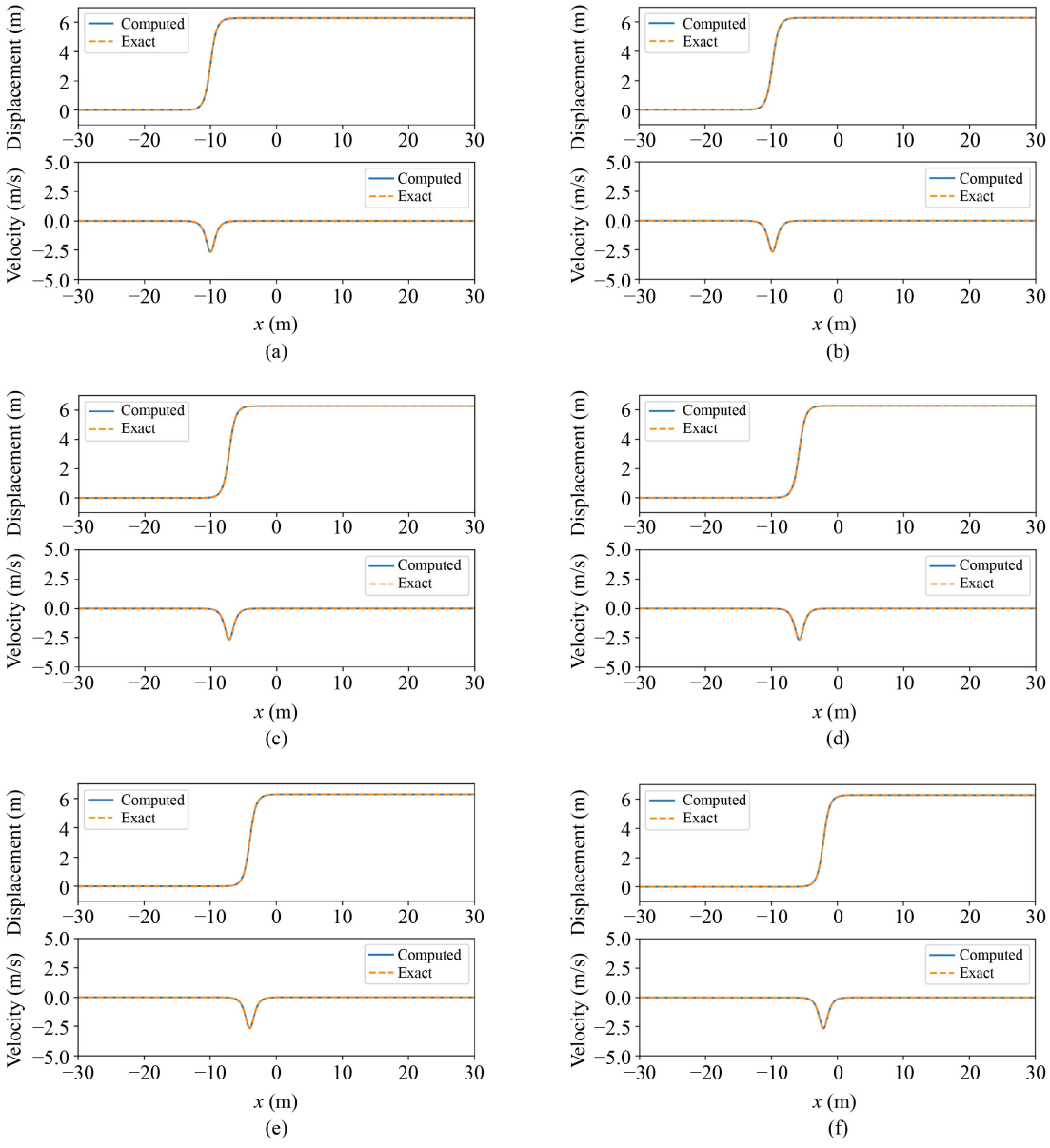


Fig. 7 Kink solution of SGE at various time instances: (a) 0.0; (b) 0.249; (c) 3.491; (d) 5.236; (e) 7.481; (f) 9.850.

travel from −10 along the spatial domain and approaches 0 toward the end of the simulation. The computed solution is compared to the exact solution, where they are observed to be in close agreement.

3.2.3 Kink-antikink solution

The kink-antikink solution of SGE is given by [35]:

$$u(x,t) = 4\arctan\left[\frac{\sinh\left(\frac{ct}{\sqrt{1-c^2}}\right)}{c\cosh\left(\frac{x}{\sqrt{1-c^2}}\right)}\right]. \tag{17}$$

The kink-antikink collision response indicates that the two kinks approach the origin separation of two kinks with velocities of c and $-c$, respectively, from the origin toward $-\infty$ and ∞ as time tends to ∞. The kink-antikink collision response of SGE estimated as a function of time using the developed methodology is plotted in Fig. 8. The solution is observed to start it's travel from the origin along the spatial domain and approaches −10 and 10 toward the end of the simulation. The computed solution is compared to the exact solution, where they are observed to be in close agreement.

3.3 Example 3: Two dimensional dynamic scalar wave equation

In the third example, the 2D dynamic scalar wave equation is solved considering the annulus of a quarter

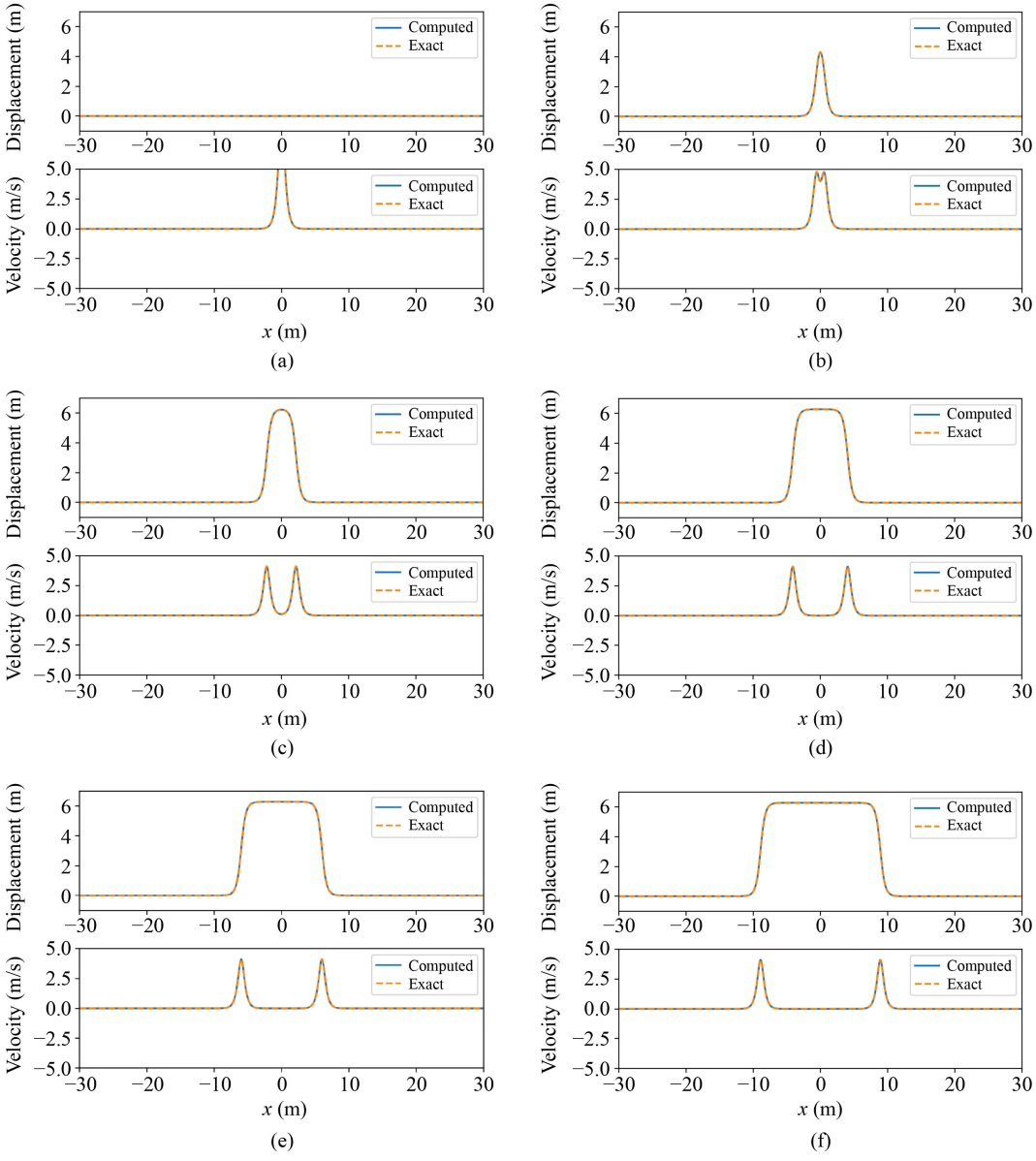


Fig. 8 Kink-antikink solution of SGE: (a) 0.0; (b) 0.623; (c) 2.369; (d) 4.488; (e) 6.608; (f) 9.975.

circle. The governing differential equation annulus in two dimensions is given by:

$$\frac{\partial^2 u}{\partial t^2} - \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = r(x,y,t), \tag{18}$$

where

$$\begin{aligned} r(x,y,t) = &-4\pi^2 \sin(2\pi t)(x^2+y^2-16)(x^2+y^2-1)\sin x \sin y \\ &-[-2\sin(2\pi t)(\sin y((x^4+(x^2)(2y^2-25)+y^4 \\ &-25y^2+50)\sin x - 2x(2x^2+2y^2-17)\cos x) \\ &-2y(2x^2+2y^2-17)\sin x \cos y)], \end{aligned} \tag{19}$$

subjected to the boundary conditions:

$$\begin{aligned} &u(x,0,t) = 0, \\ &u(0,y,t) = 0, \\ &u(x^2+y^2=1,0) = 0, \\ &u(x^2+y^2=4,0) = 0. \end{aligned} \tag{20}$$

and initial conditions:

$$\begin{aligned} &u(x,y,0) = 0 \\ &u_t'(x,y,0) = 2\pi(x^2+y^2-1)(x^2+y^2-4)\sin x \sin y. \end{aligned} \tag{21}$$

The exact solution of Eq. (18) can be written as:

$$u_{\text{exact}} = (x^2+y^2-1)(x^2+y^2-4)\sin x \sin y \sin(2\pi t). \tag{22}$$

The key computer implementation steps in the solution of 2D dynamic wave equation are mentioned in Listing 5.

The neural network function net_f_u() accepts the input variables x, y and t and returns the estimated value of the function f_u which is equal to u_tt-(u_xx + u_yy). The function f_u is later minimized during the training process.

The key computer implementation steps for solving the 2D dynamic wave equation using RK method are highlighted in Listing 6. The neural network function net_u0() accepts the locations of the internal points from the variables x and y as input and returns the variables u0 and v0 calculated at internal points. The neural network function net_u1() does the similar tasks of net_u0() along the boundary of the domain and outputs the boundary values u1 and v1, which are later used in the minimization during the training.

The solution of Eq. (18) is obtained using the collocation method, considering two hidden layers with 50 neurons each. The training time for the network is found to be 1343.0051 s. The total number of collocation points in the space and time used in the analysis are 31. The Hyberbolic tangent activation function with Adam and L-BFGS-B optimisers are employed for the analysis. The L^2 normalized error in approximation function u is observed to be 3.862141×10^{-3}. The solution along with normalized error are plotted in Fig. 9.

3.4 Example 4: Two dimensional elasto-dynamic equation

In the third example, the 2D elasto-dynamic equation is solved considering the annulus of a quarter circle. The governing differential equation annulus in two dimensions is given by:

$$\rho\frac{\partial^2 u}{\partial t^2} = f + (\lambda + 2\mu)\nabla(\nabla \cdot u) - \mu\nabla \times (\nabla \times u), \tag{23}$$

where $\rho = \mu = \lambda$, is set to be equal to 1 for selected problem. Furthermore, Eq. (23) can be explicitly written along the x direction:

$$\frac{\partial^2 u}{\partial t^2} = f_u + 3\frac{\partial^2 u}{\partial x^2} + 2\frac{\partial^2 v}{\partial x \partial y} + \frac{\partial^2 u}{\partial y^2}, \tag{24}$$

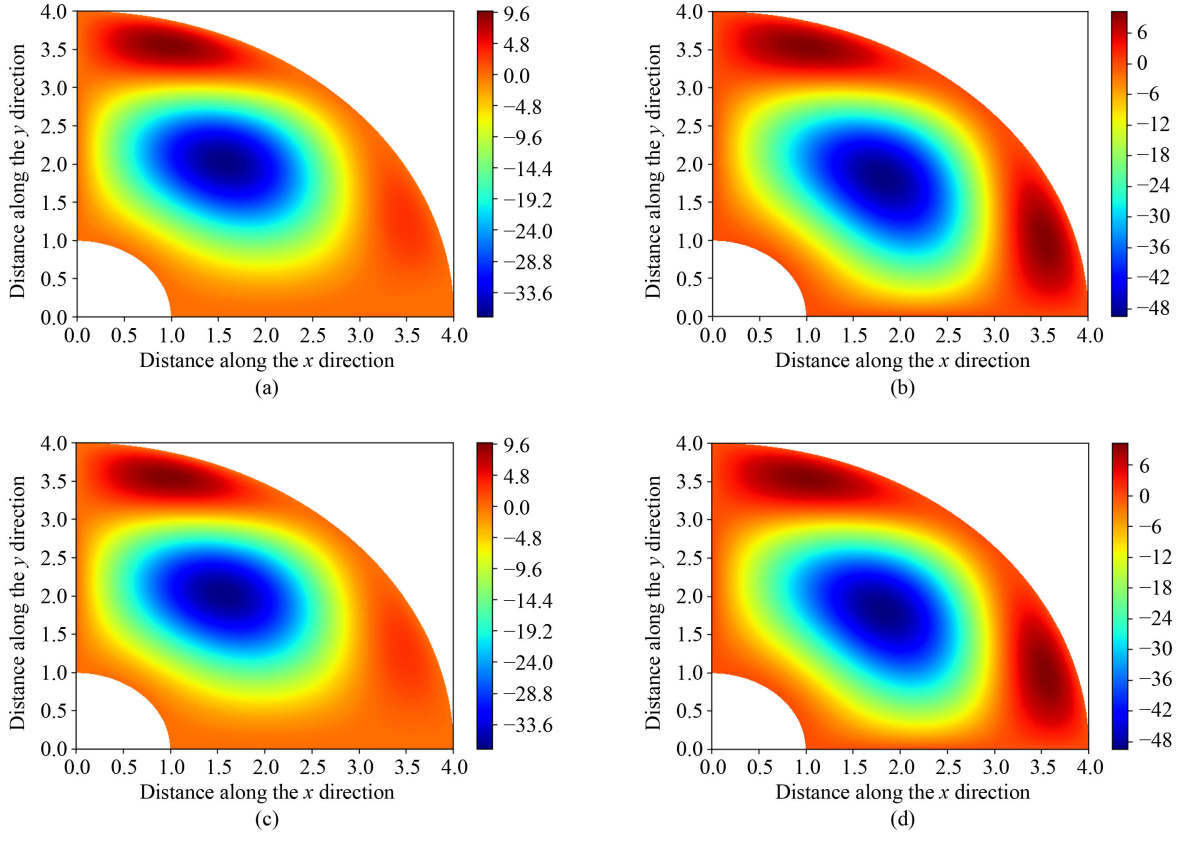
and y directions:

$$\frac{\partial^2 v}{\partial t^2} = f_u + 3\frac{\partial^2 v}{\partial x^2} + 2\frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 v}{\partial y^2}. \tag{25}$$

The adopted initial conditions are given by:

$$\begin{aligned} u(x,y,0) = v(x,y,0) = 0 \\ \frac{\partial u(x,y,0)}{\partial t} = \frac{\partial v(x,y,0)}{\partial t} \\ = 2\pi(x^2 + y^2 - 1)(x^2 + y^2 - 16)\sin x \sin y. \end{aligned} \tag{26}$$

Considering all the four ends as clamped, the exact solution is estimated to be:



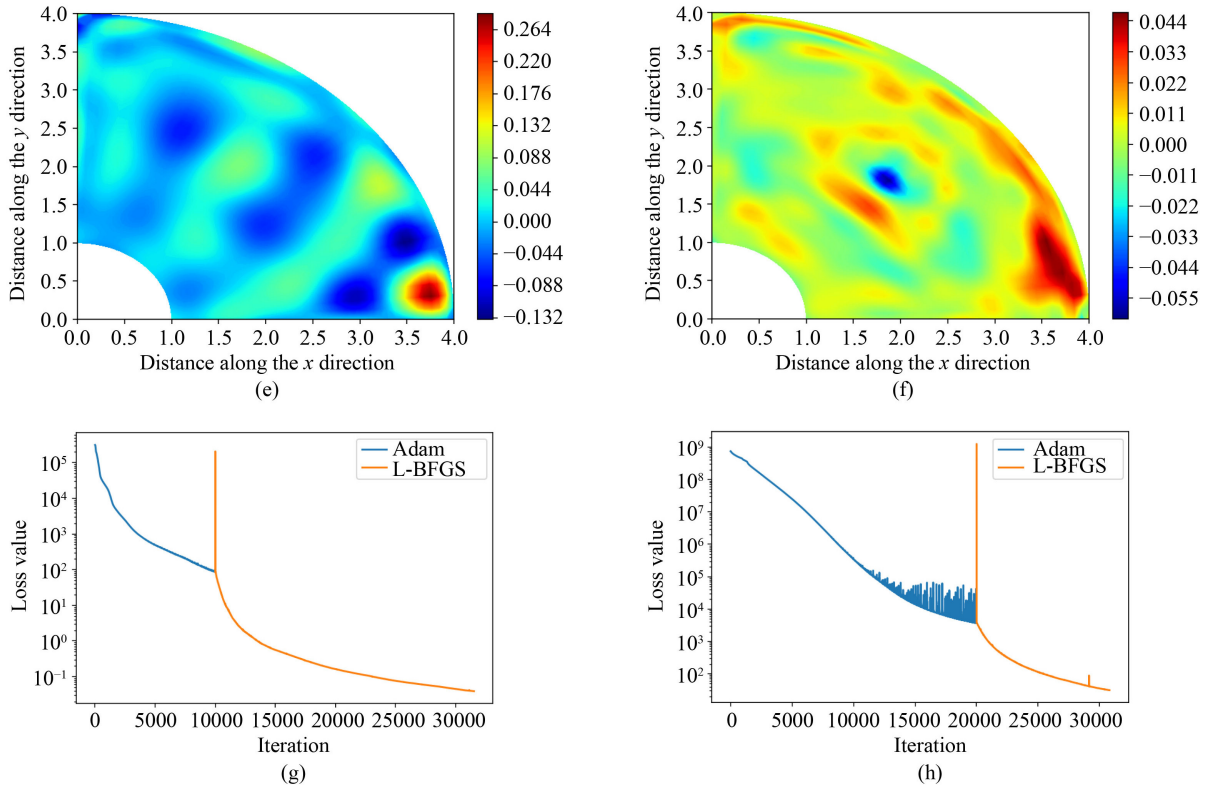
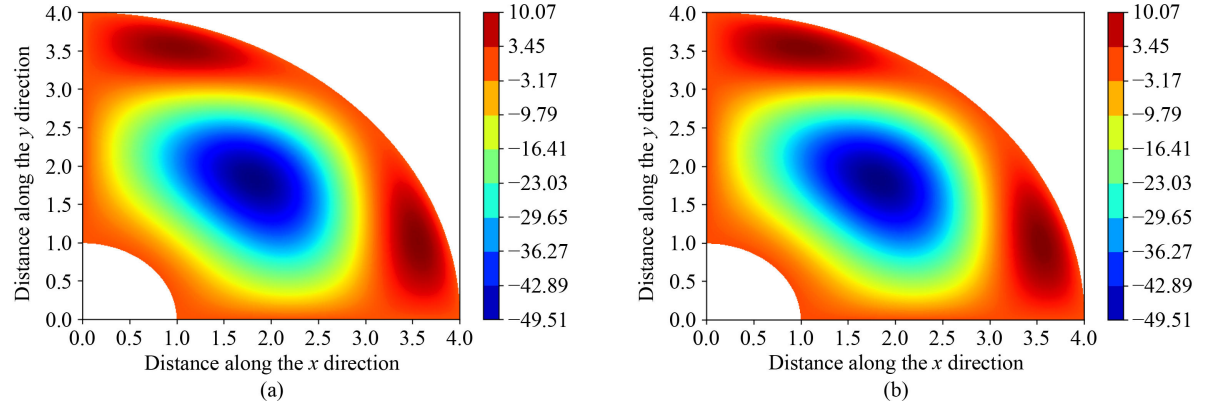


Fig. 9 Solution of 2D dynamic scalar wave equation applied on annulus of a quarter circle: (a) computed distribution of u; (b) computed distribution of u using the RK method; (c) exact distribution of u estimated using Eq. (18); (d) exact distribution of u based on Eq. (18) using the RK method; (e) distribution of error in u estimated as the difference between computed and exact values plotted in (a) and (c), respectively; (f) distribution of error in u estimated as the difference between computed and exact values using the RK method plotted in (b) and (d), respectively; (g) Distribution of the loss function; (h) distribution of the loss function arrived using the RK method.

$$u_{\text{exact}} = v_{\text{exact}} = (x^2 + y^2 - 1)(x^2 + y^2 - 16)\sin x \sin y \sin 2\pi t. \tag{27}$$

The key computer implementation steps in the solution of 2D elasto-dynamic wave equation are highlighted in Listing 7. The neural network function net_f_uv() accepts the locations of the internal points from the variables x and y as input and returns the functions f_u and f_v which are equal to u_tt-3*u xx-2*v_xy-u_yy and v_tt-3*v_yy-2*u_xy-v_xx, respectively. The variable u_xy represent the derivative of variable u with respect to variables x and y. The functions f_u and f_v are later minimized during the training process.

The solution of Eq. (23) is obtained using the collocation method, considering two hidden layers with 100 neurons each. The training time for the network is found to be 2976.9547 s. The total number of collocation points in the space and time used in the analysis are 41. The Hyberbolic tangent activation function with Adam and L-BFGS-B optimisers are employed for the analysis. The L^2 normalized error; in approximation function u is observed to be 2.804726×10^{-2} and in the approximation function v is observed to be 4.333535×10^{-2}. The solution along with normalized error are plotted in Fig. 10.



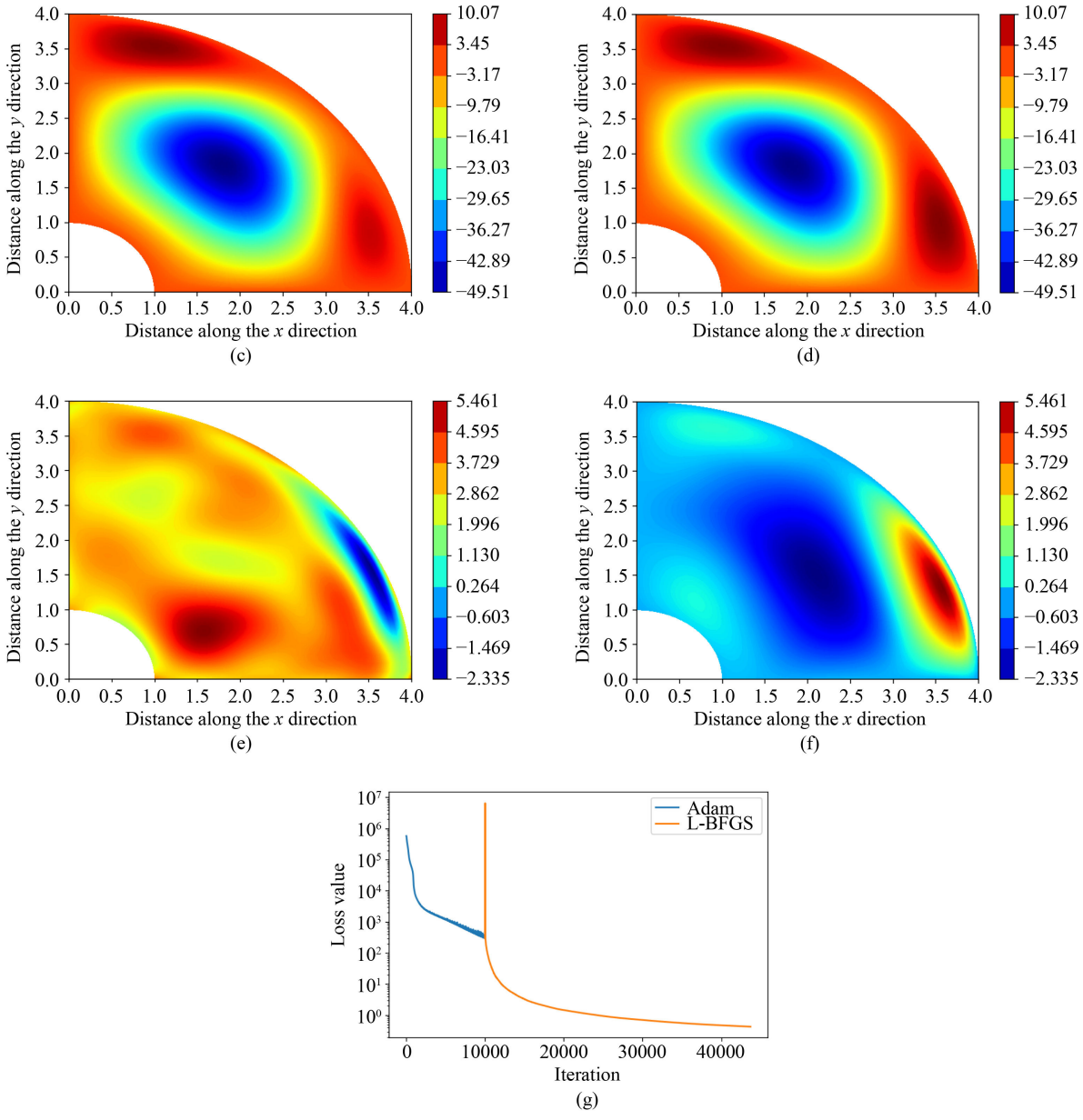


Fig. 10 Solution of 2D elastic wave equation applied on annulus of a quarter circle: (a) computed distribution of u; (b) exact distribution of u using Eq. (24); (c) computed distribution of v; (d) exact distribution of v using Eq. (25); (e) error in u estimated as the difference between computed and exact values plotted in (a) and (b), respectively; (f) error in v estimated as the difference between computed and exact values plotted in (c) and (d), respectively; (g) distribution of the loss function.

4 Conclusions

A combined DML and collocation based approach to solve the transient linear and nonlinear PDEs using ANNs has been developed. Collocation points defined in the domain were used to perform all the calculations. The framework is applied to solve the PDEs in 1D and 2D problems. The developed methodology provides flexibility to try different network architectures to further reduce the error, however, it is a tedious task. Results from the developed methodology are compared to the results estimated using the RK method.

The number of collocation points are chosen by trail and error method. Increasing the number of collocation points may decrease the error, however, this leads to a increase in the computational time. On the other hand, a penalty parameter is used to reduce the *MSE* at specific points. During the network training, the hyperbolic tangent activation function is observed to be efficient in error minimization, however, consumes more computational time to predict the solution. The network is also tried with ReLu/Relu^2 activation function, although the error is comparatively higher while the computational time is lower. Therefore, hyperbolic tangent activation

function can lead to quick minimization. The Adam optimizer is observed to be accurate during the initial iterations. Therefore, a combination of L-BFGS-B and Adam optimisers are recommended, such that the later can be efficient in the later stages of solution.

The normalized error in approximation function u as compared to the exact solution, in 1D Burgers' equation and wave equation are observed to be 2.031188×10^{-2} and 3.216116×10^{-2}, respectively. The corresponding training times are found to be 583.3 and 1052.17 s, respectively. Therefore, the exact and estimated solutions are in close agreement with each other, where the normalized error is $\approx 2\%$. The estimated results are observed to be in good agreement with the results estimated using the RK method.

The methodology is extended to solve higher dimensional problems. In particular, the 2D scalar wave equation and elasto-dynamic equation were solved and the errors were compared. Based on the solutions of 2D problems, the error plots indicates the non-uniform distribution, where the error plots vary with the selected network parameters.

The advantages of solving PDEs with the help of PINNs and their limitations, as compared to traditional methods are listed below.

1) PINNs provide continuous and differentiable solutions over the domain, leading to the provision of automatic differentiation. Whereas, numerical methods such as FEM involves the piece-wise continuous functions.

2) In neural network based methods, the computational complexity doesn't increase significantly with increase in number of sampling points.

3) The trained neural networks can be readily used without the pre-processing steps, such as: developing the model and re-meshing for any changes in design parameters. This will save a significant amount of computational time and resources.

4) PINNs are not an alternatives/replacements for classical numerical methods for solving PDEs. This is because accurate prediction of the physics using PINNs is challenging in several real time problems.

Electronic Supplementary Material Supplementary material is available in the online version of this article at https://doi.org/10.1007/s11709-024-1011-4 and is accessible for authorized users.

Acknowledgements Budarapu thankfully acknowledges the funds from the Department of Science and Technology (DST), Science and Engineering Research Board (SERB), India (No. SRG/2019/001581).

Competing interests The authors declare that they have no competing interests.

References

1. Xu L, Hui W, Zeng Z. The algorithm of neural networks on the initial value problems in ordinary differential equations. In: Proceedings of the 2nd IEEE Conference on Industrial Electronics and Applications. New York: Institute of Electrical and Electronics Engineers, 2007,813–816
2. Mall S, Chakraverty S. Comparison of artificial neural network architecture in solving ordinary differential equations. Advances in Artificial Neural Systems, 2013, 2013: 12–12
3. Yadav N, Yadav A, Kumar M. An Introduction to Neural Network Methods for Differential Equations. Berlin: Springer, 2015
4. Lagaris I E, Likas A, Fotiadis D I. Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks, 1998, 9(5): 987–1000
5. Rudd K. Solving partial differential equations using artificial neural networks. Dissertation for the Doctoral Degree. Durham: Duke University, 2013
6. Al-Aradi A, Correia A, Naiff D, Jardim G, Saporito Y. Solving nonlinear and high-dimensional partial differential equations via deep learning. arXiv, 2018: 1811.08782
7. Lagaris I E, Likas A C, Papageorgiou D G. Neural-network methods for boundary value problems with irregular boundaries. IEEE Transactions on Neural Networks, 2000, 11(5): 1041–1049
8. Fuks O, Tchelepi H A. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. Journal of Machine Learning for Modeling and Computing, 2020, 1(1): 39–61
9. Chen M, Mao S, Zhang Y, Victor C M, Leung V. Big Data Generation and Acquisition. Springer, 2014, 39–61
10. Gupta S, Modgil S, Gunasekaran A. Big data in lean six sigma: A review and further research directions. International Journal of Production Research, 2020, 58(3): 947–969
11. Waheed H, Hassan S U, Aljohani N R, Hardman J, Alelyani S, Nawaz R. Predicting academic performance of students from VLE big data using deep learning models. Computers in Human Behavior, 2020, 104: 106189
12. Yang R, Yu L, Zhao Y, Yu H, Xu G, Wu Y, Liu Z. Big Data analytics for financial market volatility forecast based on support vector machine. International Journal of Information Management, 2020, 50: 452–462
13. Thongprayoon C, Kaewput W, Kovvuru K, Hansrivijit P, Kanduri S R, Bathini T, Chewcharat A, Leeaphorn N, Gonzalez-Suarez M L, Cheungpasitporn W. Promises of big data and artificial

intelligence in nephrology and transplantation. Journal of Clinical Medicine, 2020, 9(4): 1107
14. Manzhos S. Machine learning for the solution of the schrodinger equation. Machine Learning: Science and Technology, 2020, 1(1): 013002
15. Kissas G, Yang Y, Hwuang E, Walter R, Witschey W R, Detre J A, Perdikaris P. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering, 2020, 358: 112623
16. Ghaith M, Li Z. Propagation of parameter uncertainty in swat: A probabilistic forecasting method based on polynomial chaos expansion and machine learning. Journal of Hydrology, 2020, 586: 124854
17. Russell R D, Shampine L F. A collocation method for boundary value problems. Numerische Mathematik, 1972, 19(1): 1–28
18. Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh V M, Guo H, Hamdia K, Zhuang X, Rabczuk T. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. Computer Methods in Applied Mechanics and Engineering, 2020, 362: 112790
19. de Boor C, Swartz B. Collocation at gaussian points. SIAM Journal on Numerical Analysis, 1973, 10(4): 582–606
20. Raissi M, Perdikaris P, Karniadakis G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 2019, 378: 686–707
21. Goswami S, Anitescu C, Chakraborty S, Rabczuk T. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. Theoretical and Applied Fracture Mechanics, 2020, 106: 102447
22. Anitescu C, Atroshchenko E, Alajlan N, Rabczuk T. Artificial neural network methods for the solution of second order boundary value problems. Computers, Materials and Continua, 2019, 59(1): 345–359
23. Guo H, Zhuang X, Rabczuk T. A deep collocation method for the bending analysis of Kirchhoff plate. Computers, Materials and Continua, 2019, 59(2): 433–456
24. Nguyen-Thanh V M, Zhuang X, Rabczuk T. A deep energy method for finite deformation hyperelasticity. European Journal of Mechanics. A, Solids, 2020, 80: 103874
25. Lin J, Zhou S, Guo H. A deep collocation method for heat transfer in porous media: Verification from the finite element method. Journal of Energy Storage, 2020, 28: 101280
26. Chen A, Zhang X, Zhou Z. Machine learning: Accelerating materials development for energy storage and conversion. InfoMat, 2020, 2(3): 553–576
27. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks, 1989, 2(5): 359–366
28. Lu Z, Pu H, Wang F, Hu Z, Wang L. The expressive power of neural networks: A view from the width. In: Proceedings of the Conference and Workshop on Neural Information Processing Systems, Advances in neural information processing systems. Cambridge, MA: MIT press, 2017, 6231–6239
29. Sirignano J, Dgm K S. A deep learning algorithm for solving partial differential equations. Journal of Computational Physics, 2018, 375: 1339–1364
30. Speiser D. Discovering the Principles of Mechanics 1600–1800. Berlin: Springer Science & Business Media, 2008
31. d'Alembert J R. Tracing back on the curve formed by the rope stretched in vibration. Paris: Memoirs of the Royal Academy of Sciences and Beautiful Literature, 1747 (in French)
32. Wilson C. D'Alembert versus Euler on the precession of the equinoxes and the mechanics of rigid bodies. Archive for History of Exact Sciences, 2008, 37(3): 233–273
33. Barone A, Esposito F, Magee C J, Scott A C. Theory and applications of the Sine-Gordon equation. La Rivista del Nuovo Cimento, 1971, 1(2): 227–267
34. Bour E. Surface deformation theory. Imperial College Magazine, 1862, 19: 1–48
35. Gert E. The Classical Sine-Gordon Equation (SGE). Berlin: Springer, 1981, 93–127