PLOS ONE

RESEARCH ARTICLE

Solving the initial value problem of ordinary differential equations by Lie group based neural network method

Ying Wen[1☯], **Temuer Chaolu**[2☯*], **Xiangsheng Wang**[1]

1 College of Information Engineering, Shanghai Maritime University, Shanghai, China, **2** College of Arts and Sciences, Shanghai Maritime University, Shanghai, China

☯ These authors contributed equally to this work.
* tmchaolu@shmtu.edu.cn

Citation: Wen Y, Chaolu T, Wang X (2022) Solving the initial value problem of ordinary differential equations by Lie group based neural network method. PLoS ONE 17(4): e0265992. https://doi.org/10.1371/journal.pone.0265992

Editor: Shou-Fu Tian, China University of Mining and Technology, CHINA

Received: January 4, 2022

Accepted: March 14, 2022

Published: April 6, 2022

Data Availability Statement: All relevant data within the following GitHub repository: https://github.com/yingWWen/LieNN.

Funding: The authors thanks the supporting of National Natural Science Foundation of China with grand number 11571008. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: As far as we know, there are no conflicts of interest, financial or other conflicts

Abstract

To combine a feedforward neural network (FNN) and Lie group (symmetry) theory of differential equations (DEs), an alternative artificial NN approach is proposed to solve the initial value problems (IVPs) of ordinary DEs (ODEs). Introducing the Lie group expressions of the solution, the trial solution of ODEs is split into two parts. The first part is a solution of other ODEs with initial values of original IVP. This is easily solved using the Lie group and known symbolic or numerical methods without any network parameters (weights and biases). The second part consists of an FNN with adjustable parameters. This is trained using the error back propagation method by minimizing an error (loss) function and updating the parameters. The method significantly reduces the number of the trainable parameters and can more quickly and accurately learn the real solution, compared to the existing similar methods. The numerical method is applied to several cases, including physical oscillation problems. The results have been graphically represented, and some conclusions have been made.

1 Introduction

Various fields, such as science, finance, and engineering, can transform several problems into a set of ordinary differential equations (ODEs) or partial DEs (PDEs) through mathematical modeling. Usually, the analytical solutions of these equations are unavailable. Therefore, solving the numerical solution of ODEs becomes particularly important to explore real world problems. Several off-the-shelf methods (e.g., Euler [1], Runge-Kutta [2], Finite difference [3], Adomian decomposition [4], and Lie group [5] methods) are available for solving numerical solutions of DEs [6]. Several new studies are being conducted to obtain more efficient algorithms. A class of them is the methods based on artificial neural networks (ANNs) or deep learning models [7–22]. The main idea used in these methods is the use of the highly accurate approximation capability of an ANN to a continuous function [8]. This has been used in various ways. Considering [9, 10], a representative ANN method for solving ODEs and PDEs is presented. Regarding the method, the trial solution is expressed as the sum of two terms. The

between the designated author and the editors, reviewers and readers of this magazine.

first term satisfies the initial or boundary values, and it does not contain network parameters. The second term is a feedforward NN (FNN) to be trained to satisfy the given ODEs or PDEs. However, the first term is constructed synthetically by observing the given initial or boundary values. Such free selection of the first term in the trial solution may not be suitable for capturing nonlinearity information of the real solution in the domain of the initial point. Regarding [11], He et al. used the extended back propagation algorithm to train the derivative of an FNN to solve a class of first-order PDEs. Moreover, Li-ying et al. proposed an ANN algorithm based on cosine basis functions to solve ODEs [12]. Ioannis et al. created a trial solution in the form of a neural network based on a syntactic evolution scheme to solve ODEs and PDEs [13]. Furthermore, Shekari et al. determined the approximate solutions of time-dependent PDEs based on ANNs and a hybrid method of minimization techniques and configuration methods [14]. Chakraver et al. proposed a regression-based NN model to solve the initial or boundary value problems (IBVPs) for ODEs [15]. Additionally, Mall et al. proposed a new method based on a single-layer Legendre NN model to solve the IBVP for nonlinear ODEs [16]. Dockhorn proved that small NNs (parameters <500) can accurately learn the solutions of practical problems (Poisson and Navier-Stokes equations) [17]. Shangjie [18] obtained the approximate solutions of DEs through an ANN. In [19], a deep learning algorithm is provided to solve the IBVP of a class of high dimensional stochastic PDEs. Multi-layer physical information NN deep learning was used to investigate data-driven peakon solutions and periodic peakon solutions of Camassa-Holm (CH) equation, Degasperis-Procesi equation, etc, with initial conditions [20].

The goal of using ANN to solve ODEs or PDEs is to make it computationally cheaper. Generally, performing ANNs to solve ODEs or PDEs involves a number of parameters to be trained with a lot of data. Considering several DEs, there is prior knowledge that is currently not being used in ANN algorithms or machine learning practice. These include some mathematical principles, solution expressions, or physical information. This prior information can act as a regularization agent that reduces the number of trainable parameters or the demand for a large number of training data. Therefore, using such useful information in a network algorithm results in amplifying the information content of the solution that the algorithm guides itself towards the accuracy approximation, even when only small network models or a few training samples are available. The physics-inspired NNs for solving DEs are proposed, where physical conservation laws and prior physical knowledge are encoded into the NNs (refer to [21] and references therein). For example, the logarithmic nonlinear Schrödinger (LNLS) equation is solved by a deep learning method of physical information NN. This is an important physical model in several fields such as quantum optics and nuclear physics in [22]. On the other hand, because the solution of DE is a continuous function, it is also important to find a trial function expression that is closer to the properties of the true solution. This makes the ANN effective and can quickly capture the implied properties of the real solution.

Lie group (symmetry) of DEs, universally known as one parameter transformation group method of DEs, has had a profound impact on all areas of mathematics (both pure and applied), physics, engineering, and other mathematically based sciences [5, 23, 24]. The groups can be found using symbolic computational method, and they are used to construct the explicit solutions of the corresponding DEs [25]. They also reduce the dimension in order of equations or number of involved variables, making the method efficiently solve nonlinear DEs within a certain range. Particularly, regarding an IVP of a first-order system of ODEs, this method also provides an explicit formal formula of the solution. This point is used in the present study. The advantages of the Lie groups in numerical analysis are extensively discussed in [26–28].

This study presents a method for solving the IVP of ODEs by combining the Lie group theory of ODEs and an FNN, which is significantly computationally cost effective. Moreover, the method leads to a differentiable, closed analytic form solution of the problem on whole interval

of independent variables. Considering the method, using Lie group theory, the trial solutions to the IVP are expressed by the sum of two parts. The first part is the solution to an IVP of other ODEs with the initial value of the original problem. The ODEs in the IVP are from the part terms of the original ODEs selected by obeying some solvable principles so that the first part of the solution can easily be solved in advance by Lie group method, some known symbolic or numerical methods. Owing to the specific feature of the IVP, the first part itself can capture the essential properties of the real solution in an interval of initial point of independent variable. Moreover, the first part contains no training parameters because it does not take part in training the networks. The second part of the solution is constructed by an FNN, without being required to satisfy any specific initial value.

The study makes the following contributions. This is first time that the Lie theory and ANN method are being combined to solve IVP of ODEs. Because the first part of the solution bears part of the workload and detects the nonlinearity of solution, small-scale networks and a small amount of training data (samples) can solve the IVP more accurately. The algorithm significantly increases efficiency of the network method to solve the IVP of ODEs. On the other hand, the method provides new idea and enlighten on how to add "complementary elements" and design an ANN to increase the efficiency of ANN algorithm to solve DEs by incorporating with mathematical theory. Furthermore, the method can be applied to more a general form IVP or BVP of ODEs. Moreover, after modifying or combining some of the existing methods [29], the approach can be applied to IVP and BVP of PDEs.

The rest of the paper is structured as following. In Section 2, we introduce the Lie group method for solving IVP of ODEs and an expression of solution to the IVP is proved. In Section 3, the general process of the proposed method is given. In Section 4, numerical experiments on the application of our method to some problems are given, including the oscillation model in physical problems. Finally, in Section 5, some discussions as well as future research directions are presented.

2 Lie group expression for solution to the IVP of a DEs

2.1 A basic theorem

Let $G_1 = \{T_a\}$ be a one parameter (denoted by a) transformations $x^* = T_a(x)$ from $x = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$ to $x^* = (x_1^*, x_2^*, \cdots, x_n^*) \in \mathbb{R}^n$ with components

$$T_a : x_i^* = f_i(x; a), i = 1, 2, \ldots, n. \tag{1}$$

Here functions $f = (f_1, f_2, \cdots, f_n)$ are continuous and differentiable with respect to (w.r.t) independent variables x_i and parameter a varying in a neighborhood $\mathcal{O}$ of $0 \in \mathbb{R}^1$.

Let

$$D = \sum_{i=1} \xi_i(x) \partial x_i \tag{2}$$

be a linear partial differential operator generated from the auxiliary functions defined by

$$\xi^i(x) = \left. \frac{\partial f_i(x; a)}{\partial a} \right|_{a=0}, i = 1, 2, \ldots, n. \tag{3}$$

From [24], we know that set G_1 form an one parameter transformation group with generator D from $\mathbb{R}^n$ to itself if f_i satisfy additional conditions: i) $f_i(x, 0) = x_i$; ii) $f_i(f(x, a), b) = f_i(x, a + b)$; iii) f_i is differential w.r.t parameter a. In the group case, by Taylor expansion at $a = 0$, we

have expressions of x_i^* as follows

$$x_i^* = f_i(x;a) = x_i + \xi_i(x)a + O(a^2) = e^{aD}x_i, i = 1, 2, \cdots, n. \tag{4}$$

Here formal notation $e^{aD} = \sum_{k=0}^{\infty} \frac{a^k}{k!} D^k$ is used.

The correspondence between set G_1 and an one parameter transformation group is given in the following theorem (The more details can be seen in [5, 23, 24]).

Theorem 1 The set G_1 with functions $x_i^* = f_i(x;a)$ in (1) defining an one parameter transformation group satisfies the IVP of ODEs

$$\frac{dx^*}{da} = Dx^*, x^*|_{a=0} = x. \tag{5}$$

Conversely, for any given continuously differentiable functions $\xi_i(x)$, i.e., the operator D determined by (2), the function $x^* = f(x;a)$ obtained by solving the problem (5) determines an one parameter transformation group with D as generator. Moreover, the transformations $T_a : x^* = f(x;a)$, i.e. the solution of (5) can be expressed as formula (4), i.e., $x^* = e^{aD}x$.

2.2 An expression of solution to IVP of an ODEs

Effectively finding solution of IVP (5) depends on the complexity of the operator D. To reduce this difficulty, let's assume that D in (5) can be decomposed into summands of two parts as

$$D = D_1 + D_2. \tag{6}$$

Thus, we have

Theorem 2 (A solution expression) For an operator D in (5) with decomposition part $D_1 = \sum_{i=1}^{n} g_i(x)\partial_{x_i}$ in (6) and initial values $x \in \mathbb{R}^n$, the solutions to (5) have expression as

$$x^* = e^{aD}x = \bar{x}(x;a) + \tilde{\mathcal{N}}(x;a), \tag{7}$$

where $\bar{x} = \bar{x}(x;a) = e^{aD_1}x$ and $\tilde{\mathcal{N}}(x;a) = \int_0^a D_2(e^{(a-\tau)D}x)|_{x\to\bar{x}(\tau;x)}d\tau$.

The proof of the theorem is obtained by Grobner's formula in [26] given by

$$e^{aD} = e^{aD_1} + \sum_{\alpha=1}^{\infty}\sum_{k=\alpha}^{\infty} \frac{a^k}{k!} D_1^{k-\alpha} D_2 D^{\alpha-1}.$$

Obviously, the first part $\bar{x} = e^{aD_1}x$ of the above formula (7) is the solution to new IVP

$$\frac{d\bar{x}}{da} = D_1\bar{x}, \bar{x}|_{a=0} = x, \tag{8}$$

by Theorem 1.

Evidently, by suitably choosing the first part D_1 of D in (6), IVP (8) is more easily solved than original IVP (5) by Lie group method as well as various symbolic or numerical methods [6]. Moreover, in [24, 26], it is proved that under probably selection D_1, the solution of (8) higher accurately approximates the real solution of (5) in some interval of initial point $a = 0$.

Although the second part $\tilde{\mathcal{N}}$ in (7) will computed by symbolically in some simple cases, in general it is extremely complicate if one directly solve it. In this article, we use the formula (7) to design a FNN to solve the term.

Remark 1: It is notice that from (7), we have $\tilde{\mathcal{N}}(x,0) = 0$. Hence without loss of generality we suppose $\tilde{\mathcal{N}}(x;a) = a\bar{\mathcal{N}}(x;a)$ for a differential function $\bar{\mathcal{N}}(x;a)$.

3 Method

In the section, we describe the scheme of our Lie group based on FNN method which relies on the expressions (7) of the solutions to an IVP of ODEs given in Theorem 2.

3.1 Scheme

Our method, first, is designed for the IVP of an autonomous system of ODEs

$$\frac{dy_i}{dx} = f_i(y_1, y_2, \cdots, y_n),\ y_i(0) = \alpha_i \in \mathbb{R}^1, i = 1, 2, \cdots, n, \tag{9}$$

where $x \in \mathcal{O} \in \mathbb{R}^1$ is independent variable and $y_i = y_i(x)$ are dependent variables and f_i are differential functions of own arguments. Then, we extend the method to solve different form ODEs or PDEs problems.

We rewrite IVP (9) as operator form

$$\frac{dy}{dx} = Dy,\ y(0) = \alpha, \tag{10}$$

by introducing notations $y = (y_1, y_2, \cdots, y_n)$, $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ and operator $D = \sum_{i=1}^{n} f_i(y) \frac{\partial}{\partial y_i}$, which has the same form with (5). Consequently, by Theorem 2, the solution of IVP (9) can be written as

$$y = e^{xD}\alpha = e^{xD_1}\alpha + x\bar{\mathcal{N}}(x; \alpha) \tag{11}$$

for a decomposition $D = D_1 + D_2$ and functions $\bar{\mathcal{N}}(x; \alpha) = \{\bar{\mathcal{N}}_i(x, \alpha)\}_{i=1}^{n}$ with x as group parameter. Here we understand that $e^{xD}\alpha = e^{xD} y|_{y\to\alpha}$.

We use this expression of the solution y to design a FNN method to approximate $y(x)$ by a neural network $\hat{y}(x, \theta)$, where θ is the neural network parameters. To this end, we require that the network (trial solution) has the expression

$$\hat{y}(x) = \bar{y}(x) + x\mathcal{N}(x; \theta), \tag{12}$$

where $\mathcal{N}(x; \theta)$ is a differential function to be determined and $\bar{y}(x) = e^{xD_1}\alpha$ which is solution of the following new IVP

$$\frac{d\bar{y}}{dx} = D_1\bar{y} = g(\bar{y}),\ \bar{y}(0) = \alpha, \tag{13}$$

by Theorem 1 for $D_1 = \sum_{i=1}^{n} g_i(\bar{y})\partial_{\bar{y}_i}$.

Therefore, the deviations between the trial solution $\hat{y}$ in (12) and exact solution y in (11) is characterized by

$$\Delta y = |x(y(x) - \hat{y}(x))| \tag{14}$$

for x in considered interval.

Remark 2: We call IVP (13) as an associated IVP of original IVP (9) or (10). Its solution $\bar{y}(x)$ is determined in advance by solving IVP (13) and by applying the Lie group method or other acknowledged various methods after appropriately selecting operator D_1. The solvability and good approximation capability of the associated IVP solution $\bar{y}$ are main considerations of choosing operator D_1.

Consequently, compared with synthetic ways to give the first part of the trial solution in literatures, here the first part of the trial solution is constructed by a solution of the associated IVP which more accurately approximates the real solution of IVP (9) to be solved in an interval

of initial point $x = 0$. Due to the promising feature of efficient approximation of the first term in (12) to exact solution, the construction of the second term in (12) by networks algorithm takes much less a computational burden than that of solving the overall solution with big scale training parameters. Hence, in automatic learning (training) procedure, the network detecting quickly tunes itself to approach the natural properties of the solution. This is the main our motivation why we combine the Lie group method with ANN. Therefore, our attention focuses on determining the second part of solution (12).

This unknown function $\mathcal{N}(x;\theta)$ in the second part of the trial solution (12) is constructed by a FNN through optimizing the loss (error) function

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{k}^{N}\sum_{i=1}^{n}\left\{\frac{d\hat{y}_i^k}{dx} - f_i(\hat{y}_1^k, \hat{y}_2^k, \cdots, \hat{y}_n^k)\right\}^2, \tag{15}$$

where $\hat{y}_i^k = \hat{y}_i(x_k, \theta)$, $\frac{d\hat{y}_i^k}{dx} = \frac{d\hat{y}_i^k(x,\theta)}{dx}|_{x\to x_k}$ and the data set $S = \{x_k\}_{k=1}^N$ is a set of train points obtained from training interval $\mathcal{O}$ in some distributive sense.

Assuming that the operator $D = D_1 + D_2$ in (10) has expressions

$$D_1 = \sum_{i=1}^{n} g_i(y)\partial y_i,\ D_2 = \sum_{i=1}^{n} h_i(y)\partial y_i \text{ with } f_i = g_i + h_i, \tag{16}$$

Thus the loss function (15) is rewritten as

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{k}^{N}\sum_{i=1}^{n}\left(g_i(\bar{y}^k) + \left(\frac{d(xN_i(x;\theta))}{dx}|_{x=x_k} - f_i(\hat{y}^k)\right)\right)^2, \tag{17}$$

obtained by substituting (12), (13) and (16) into (15).

In the loss function (17), the $g_i(\bar{y}^k)$ is independent of training parameters θ. This saves more computational effort. This is another benefit of combining Lie expression of DE solution and ANN algorithm.

In our case (9), the network models should have one- input x and an n- output $\mathcal{N}(x;\theta) = \{\mathcal{N}_i(x;\theta)\}_{i=1}^n$ layer. We use the network possessing m neurons as model to present our proposed method scheme.

For each input x, there are outputs $\mathcal{N}_i(x;\theta)$ as follows

$$\mathcal{N}(x;\theta) = \delta(\hat{\mathcal{N}}_1, \hat{\mathcal{N}}_2, \cdots, \hat{\mathcal{N}}_n,),\ \ \hat{\mathcal{N}}_j = \sum_{i=1}^{m} v_{ji}\cdot\sigma(x\cdot w_i + b_i) + c_j, \tag{18}$$

where trainable parameters $\theta = \{w_i, v_{ji}, b_k, c_j\}$ and w_i is the weight from the input x to the ith neuron in the hidden layer and $v_j = \{v_{j1}, v_{j2}, \cdots, v_{jm}\}$ and v_{ji} is the weight vector from the ith neuron in the hidden layer to the jth neuron in output layer, b_j is the bias of the jth neuron in hidden layer and c_j is the bias of the jth output neuron. The both $\sigma(\cdot)$ and $\delta(\cdot)$ are activation functions for outlets of hidden and output layers respectively. In our examples given next section, we use a Sigmoid function as hidden layer activation and take a linear activation in the output layer for our networks.

In training the networks, the values of loss function are obtained by the forward training with formula (18), and the gradient descents of loss function (17) are obtained in the backpropagation; minimizing the loss function involves not only the network output $\mathcal{N}(x;\theta)$, but also the derivatives of the network output $\mathcal{N}(x;\theta)$ w.r.t the input and network parameters; while these derivatives are given recursively by the ones of $\hat{\mathcal{N}}_i$ and functions δ and σ; the update

network parameters are using unified formula as

$$\theta^{k+1} = \theta^k + \Delta\theta^k = \theta^k - \eta \frac{\partial \mathcal{L}(\theta)}{\partial \theta^k},$$

where η is the learning rate and k is the iteration step (refer to [9]).

3.2 Principle to select operator D_1

The efficiency of the proposed method above depends on the selecting of operator D_1 in decomposition $D = D_1 + D_2$. The main principles in choosing D_1 are given in the following considerations.

1. The associated IVP (13) is easily solved explicitly or numerically. In particular, we choose g_i in (16) as part terms of f_i in (9) so that the associated IVP (13) yields explicit or numerical solutions.
2. The operations of calculation on loss function (17), such as calculation of derivatives and values of them, are not expansive as possible as.

3.3 Extension

The method proposed above can be applied to any IVP or BVP of DEs which as long as can be transformed to the form of (9).

3.3.1 Non-autonomous case. For the non-autonomous case

$$\frac{dy_i}{dx} = f_i(x, y_1, y_2, \cdots, y_n),\ y_i(0) = \alpha_i \in \mathbb{R}^1, i = 1, 2, \cdots, n, \tag{19}$$

we introduce a new variable $y_0 = x$ and add an equation $\frac{dy_0}{dx} = 1$ and initial value $y_0(0) = 0$ to (19). Then, the current IVP turns to the standard IVP (9) with $n + 1$ unknown functions $y_0, y_1, \cdots, y_n$ and operator $D = \sum_{i=1}^{n} f_i \partial_{y_i} + \partial_x$.

3.3.2 High-order case. For the general IVP of an n-order non-autonomous ODE

$$\begin{aligned} &\frac{d^n y(x)}{dx^n} = f(x, y, y', \ldots, y^{(n-1)}), \\ &y^{(k)} = a_{k+1},\ \ k = 0, 1, \ldots, n-1, \end{aligned} \tag{20}$$

we can transform it to the standard IVP (9) by introducing transformations $y_0 = x, y_1 = y, y_2 = y_1', \ldots, y_n = y_{n-1}'$ with correspondingly $f_0 = 1, f_1 = y_2, \ldots, f_{n-1} = y_n, f_n = f(y_0, y_1, \ldots, y_n)$.

3.3.3 Nonzero start points. In nonzero start point $x = x_0$ in (5), we do a translation $x \to x - x_0$ to change the problem into standard form.

3.3.4 PDE case. The most intuitive idea is to convert the PDE to the ODE and then solve the original problem indirectly. For example, use symmetries of PDE, traveling wave transformations, etc to decrease the order of the PDEs and number of independent variables involved in the PDEs. Another idea is to combine existing semi-discrete methods for solving PDE [29]. Discrete the underline PDEs in spatial variables approximately so that the resulting semi-discrete equation can be cast into an ODEs (9), then use the proposed method on the resulted ODEs.

3.3.5 Two point problem case. To two points problem of ODEs, we firstly regard the problem as an IVP with using a point value as initial value problem, then put another point value in the loss function so that the network automatically learn the boundary value.

Next section, by applying the given method on solving some physical problems, we illustrate the superiority of the method.

4 Numerical experiments and applications

In this section, we provide some examples to show the effectiveness of our proposed method. Considering each example, we evaluate the method by calculating the error between our results and the exact solutions (if available) or numerical solutions obtained from known numerical methods. Regarding all the examples, a small architecture FNN with one hidden layer of only three neurons, a linear output, and a small training data is used. To illustrate the feature of the solution obtained by our method, we provide figures displaying the graph of the solution and the exact solution of the training interval. In addition, we draw the compared figures of first terms of the trial solutions given in present article and literatures to show the capabilities of the terms to capture nonlinear properties of the real solutions. Moreover, we consider points outside the training interval (some extensions of the training interval) to show the generalization and stability of our method.

Example 1. The first example shows the procedure and efficiency of our proposed method by comparing it with the methods provided in previous studies. We solve the IVP of two coupled first-order nonlinear ODEs considered in [9].

$$\begin{aligned}\frac{dy_1}{dx} &= \cos(x) + y_1^2 + y_2 - (1 + x^2 + \sin^2(x)),\\ \frac{dy_2}{dx} &= 2x - (1 + x^2)\sin(x) + y_1 y_2,\end{aligned} \tag{21}$$

with $x \in [-1, 1]$ (training interval) and initial values $y_1(0) = 0$ and $y_2(0) = 1$. The IVP has exact solutions $y_1(x) = \sin(x)$ and $y_2(x) = 1 + x^2$.

To apply our method, let's turn the problem into the form of (9), i.e., an autonomous system of equations

$$\begin{cases} y_1'(x) = f_1(y_0, y_1, y_2) = \cos(y_0) + y_1^2 + y_2 - (1 + y_0^2 + \sin^2(y_0)),\\ y_2'(x) = f_2(y_0, y_1, y_2) = 2y_0 - (1 + y_0^2)\sin(y_0) + y_1 y_2,\\ y_0'(x) = f_0(y_0, y_1, y_2) = 1, \end{cases} \tag{22}$$

with introducing variables $y_0 = x$ and using initial values $y_0(0) = y_1(0) = 0$, $y_2(0) = 1$.

From (16), the differential operator D of this equation is $D = f_1\partial y_1 + f_2\partial y_2 + f_0\partial y_0$. We select the linear parts of functions f_i to construct operator $D_1 = g_1\partial y_1 + g_2\partial y_2 + g_0\partial y_0$ in which $g_1(y_0, y_1, y_2) = \cos(y_0) + y_2 - (1 + y_0^2 + \sin^2(y_0))$, $g_2 = 2y_0 - (1 + y_0^2)\sin(y_0)$, $g_3 = 1$. Hence, the associated IVP of (22) corresponding to operator D_1 is a linear system

$$\frac{d\bar{y}_i}{dx} = e^{xD_1}\bar{y}_i = g_i(\bar{y}_0, \bar{y}_1, \bar{y}_2), i = 0, 1, 2, \tag{23}$$

with initial values $\bar{y}_0(0) = \bar{y}_1(0) = 0, \bar{y}_2(0) = 1$. Its exact solutions $\bar{y}_0 = x$ and

$$\begin{aligned}\bar{y}_1(x) &= x^2\sin(x) + x/2 - 4\sin(x) + \sin(2x)/4 + 4x\cos(x),\\ \bar{y}_2(x) &= 2 + x^2 + x^2\cos(x) - 2x\sin(x) - \cos(x),\end{aligned}$$

are easily obtained by Computer algebra system Mathematica. Now, the trial solutions of the

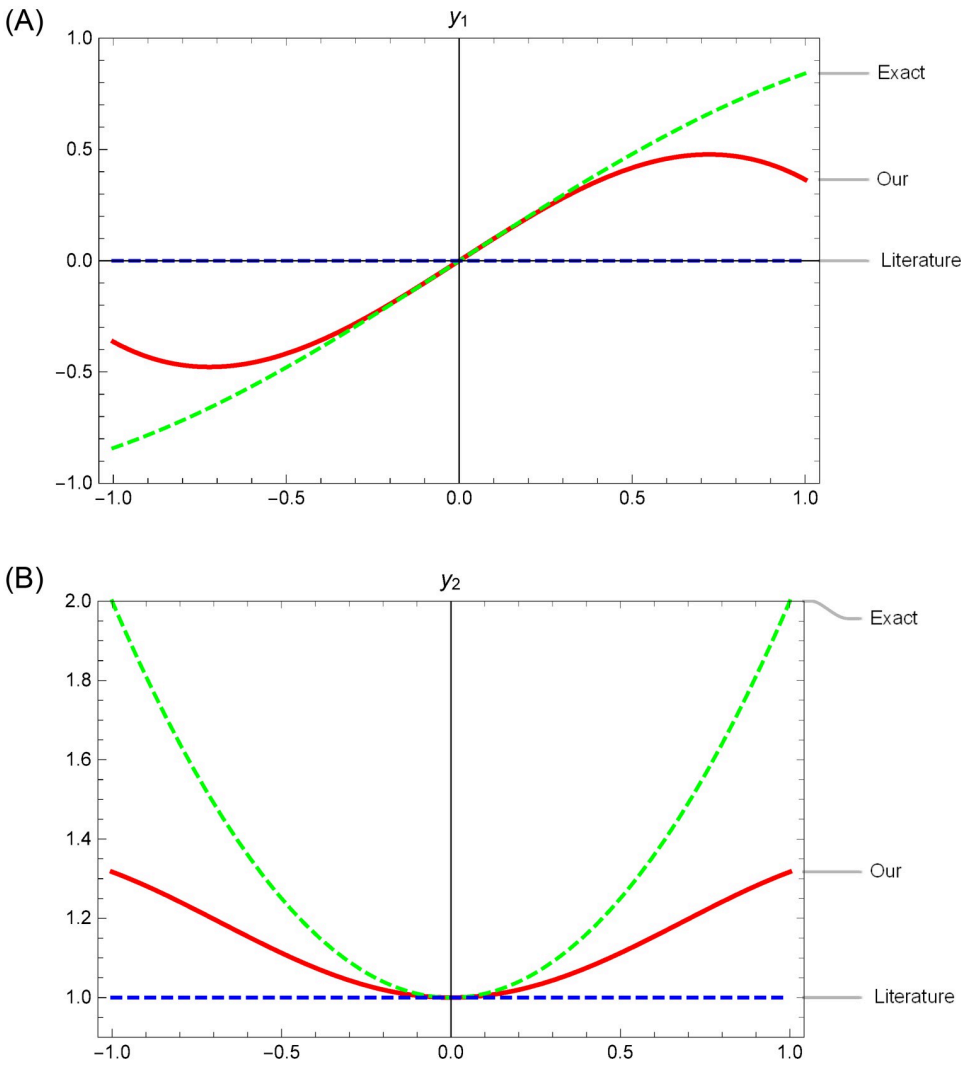

Fig 1. Comparison of the efficiencies of the first terms $\bar{y}_{1,2}$ in the trial solutions approximating the exact ones $y_{1,2}$ in an interval of $x = 0$ in Example 1. A:Comparison of $\bar{y}_1$ and y_1 at the initial point $t = 0$. B:Comparison of $\bar{y}_1$ and y_1 at the initial point $t = 0$.

https://doi.org/10.1371/journal.pone.0265992.g001

two neural networks are supposed to be

$$\bar{y}_1(x) = \bar{y}_1 + x\mathcal{N}_1(x;\theta), \quad \bar{y}_2(x) = \bar{y}_2 + x\mathcal{N}_2(x;\theta). \tag{24}$$

The first parts of the trial solutions are simply adopted as $\bar{y}_1 = 0$ and $\bar{y}_2 = 1$ [9]. Considering Fig 1, the degrees to which the various first parts $\bar{y}_{1,2}$ used in both our trial solutions (24) and reference [9] approximate the true solutions near the initial point $x = 0$ are shown. The first parts used in (24) efficiently capture the nonlinear properties of the real solutions. This accelerates the convergence of the subsequent neural computations in our method.

Our networks consist of one hidden layer with three neurons trained on a set of only 21 equidistant points in the interval [−1, 1]. They yield higher accuracy approximate solutions of IVP (24) as shown in the left figure of Fig 2. Moreover, the solutions obtained by our method can approximate the exact solutions very well outside the training set till at least interval [−1.5,

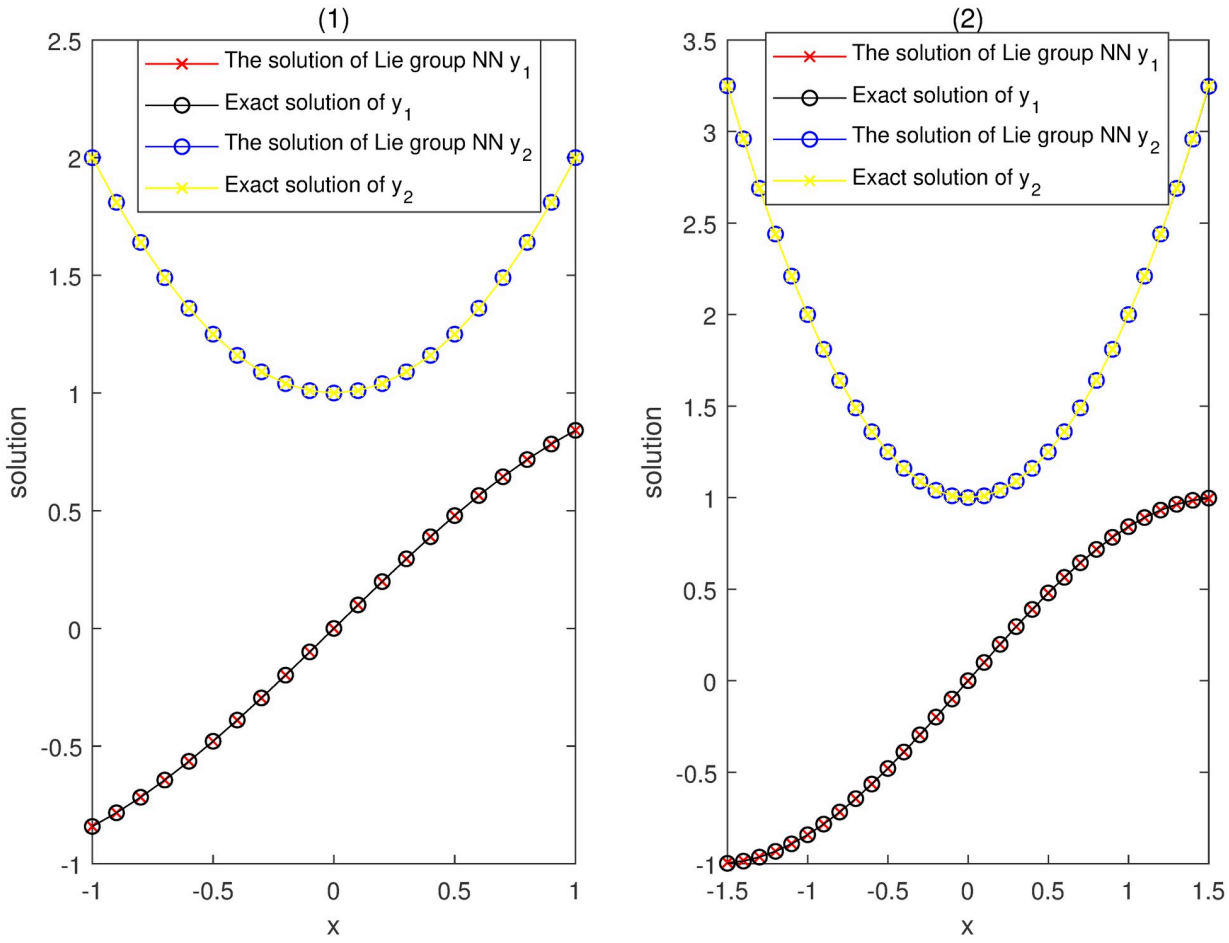

Fig 2. Comparison between our solutions $\hat{y}_{1,2}$ and the exact solutions $y_{1,2}$ in Example 1.

https://doi.org/10.1371/journal.pone.0265992.g002

1.5] as shown in the right figure of Fig 2. This indicates that this method has a higher accuracy on the whole training interval, better generalization (inertia), and stability, even when using small-scale networks and data set for the training.

Considering Fig 3, the accuracies by computing the deviations $\Delta_i = y_i - \hat{y}_i$ between the trial solutions $\hat{y}_i$ obtained by our method and the exact solutions y_i on the training set are controlled in 10^{-5} order of magnitudes. This shows that our method admits strong robustness on the training interval by fewer training samples. Regarding the predicted (no training) intervals $[-1.5, -1]$ and $[1, 1.5]$, although the errors are increasing, the accuracies are maintained in 10^{-3} order of magnitudes without the training data.

Regarding Fig 4, the dependence of the network errors on the number of iteration steps in our methods is shown. Considering the figure, when the iteration is approximately 130 step, the error is close to 0, and this state is persisted. This shows that the convergence of our method is faster, demonstrating the stability of our algorithm.

Compared to the method in a previous study [9] in which network architecture consisted of one hidden with ten neurons, our method achieves higher accuracy solutions in a shorter computational time by using fewer network parameters and small scale training data.

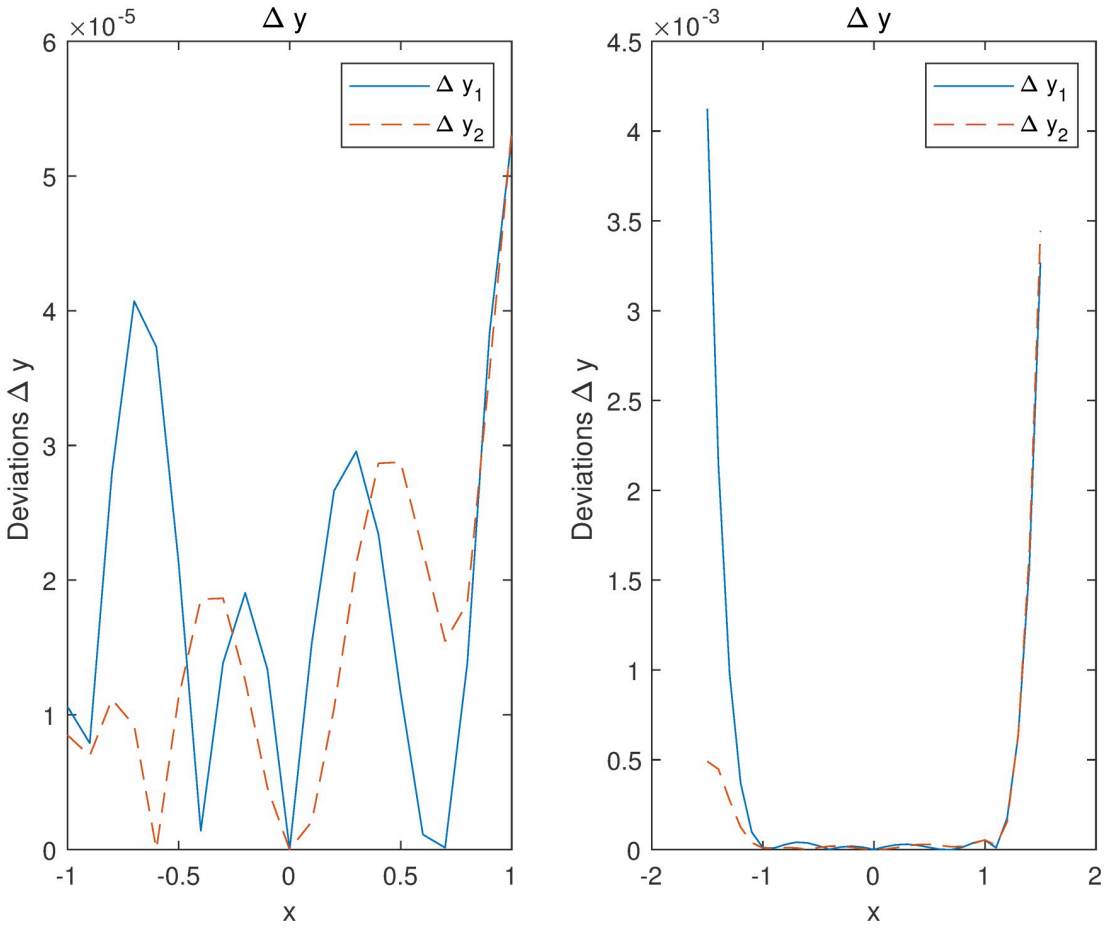

Fig 3. Accuracies of the $\hat{y}_{1,2}$ approximating exact ones $y_{1,2}$ in Example 1.

https://doi.org/10.1371/journal.pone.0265992.g003

Example 2. We consider the linearly forced oscillation problem

$$\ddot{y} + 2\epsilon\dot{y} + y = f(t), y(0) = a, \dot{y}(0) = b, \tag{25}$$

where $f(t)$ is a known function and a and b are constants and ϵ is damping parameter with $0 < \epsilon < 1$. Since the solution of the problem is sensitive to the parameter $\epsilon \to 0$. It is well known that it was solved by Multiple scales and averaging methods [30]. To apply our method, let's turn the problem into the form of (9)

$$\dot{y}_0 = 1, \ \ \dot{y}_1 = y_2, \dot{y}_2 + 2\epsilon y_2 + y_1 = f(y_0), \tag{26}$$

with initial values $y_0(0) = 0$, $y_1(0) = a$, $y_2(0) = b$ by letting $y_0 = t$, $y_1 = y$, which yields the operator $D = y_2\partial_{y_1} + (f(y_0) - y_1 - 2\epsilon y_2)\partial_{y_2} + \partial_{y_0}$. In order to its associated IVP more accurately approximate the real solution of (26) and be solved easily, we take $D_1 = y_2\partial_{y_1} - (y_1 + 2\epsilon y_2)\partial_{y_2} + \partial_{y_0}$ and $D_2 = f(y_0)\partial_{y_2}$ as in (16).

By symbolic computation, one obtains the exact solutions of the associated IVP of (26) for selected the D_1 as $\bar{y} = (\bar{y}_0, \bar{y}_1, \bar{y}_2) = e^{tD_1}(y_0, y_1, y_2)|_{y_0 \to 0, y_1 \to a, y_2 \to b}$, i.e., $\bar{y}_0 = t, \bar{y}_1 = \int_0^t \bar{y}_2(t)dt$ with

$$\bar{y}_1 = \sigma^{-1}e^{-\epsilon t}((a\epsilon + b)\sin(\sigma t) + a\sigma\cos(\sigma t)), \ \ \text{with } \sigma = \sqrt{1 - \epsilon^2}.$$

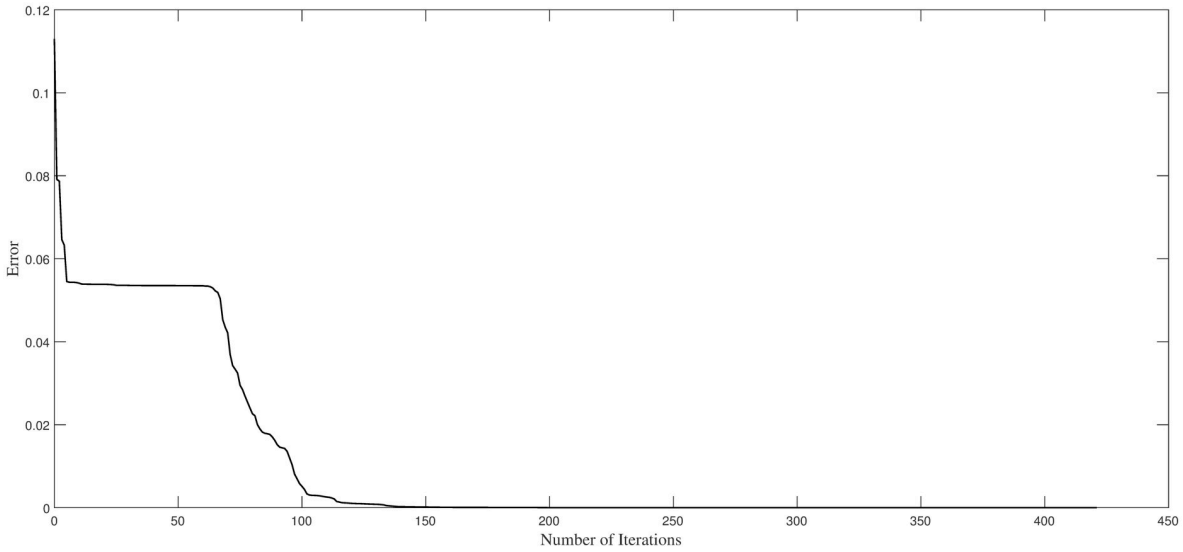

Fig 4. Trend of training errors with iteration steps in Example 1.

https://doi.org/10.1371/journal.pone.0265992.g004

Therefore, we have trial solution as

$$\hat{y}_1 = \bar{y}_1 + t\mathcal{N}(t, \theta).$$

To concretely computation, we take example $\epsilon = 0.2$, $a = 0$, $b = 1$ and $f(t) = -0.4e^{-0.4t}\cos t$. In this situation, IVP (25) has exact solution

$$y(t) = e^{-0.4t}\sin(t)$$

Regarding the previous studies, the first part of the trial solutions should be simply considered $\bar{y}_1 = 0$ or x or $x(1 + x)$. The degrees to which these first terms of the trial solutions approximate the exact solution in the neighborhood of initial point are shown in the Fig 5. This shows that the first term in the trial solution $\hat{y}_1$ more accurately captures the nonlinearity of the exact solution y in an interval of the initial point.

We use a grid of 21 equidistant points in [0, 2] as the training and testing data for training our networks $\mathcal{N}(t, \theta)$. Using such a small-scale network and fewer training data, our method obtains more accurate results.

The comparisons between the exact solutions $y_{1,2}$ and approximated ones $\hat{y}_{1,2}$ provided by our method on the training interval [0, 2] and its extension [0, 2.5] are shown in Fig 6.

Considering Fig 7, the trend of the network errors $\mathcal{L}(\theta)$ as the number of iteration steps increases is shown. The graph reflects the rapid decline in the algorithm, and the error is close to zero at approximately 220 iterations, reaching 10^{-7}. This indicates that the algorithm converges quickly and is stable.

Considering Fig 8, the deviations Δy_i of solution y_i on the training and test points are shown. It can be observed that the algorithm has a good generalization performance and a higher accuracy even on few training samples and a small network structure.

To show the robustness of our method, we investigate the performances of our method on different values of parameter ϵ in (25) for the same nonhomogeneous term $f(t)$. The results are shown in the following Table 1. For more information, see S1 File.

From the table, we can see that the strong stability of our algorithm is presented.

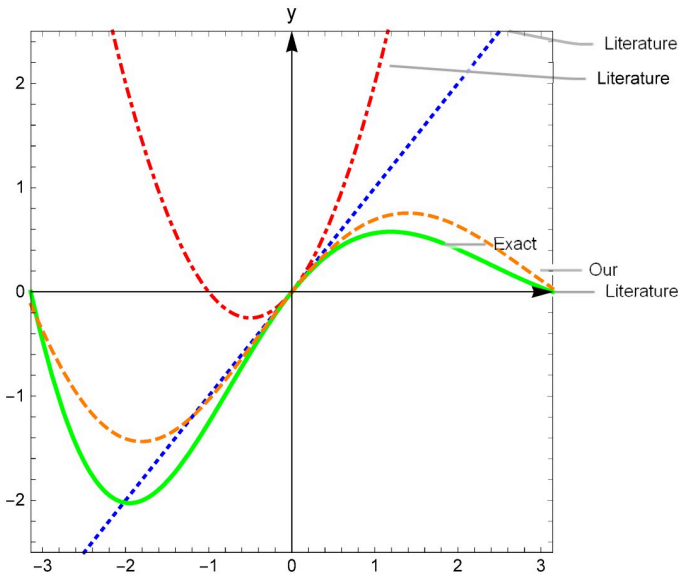

Fig 5. Comparison of the efficiencies of first term $\bar{y}_1$ in trial solutions $\hat{y}_1$ approximating the exact one y in an interval of $t = 0$ in Example 2.

https://doi.org/10.1371/journal.pone.0265992.g005

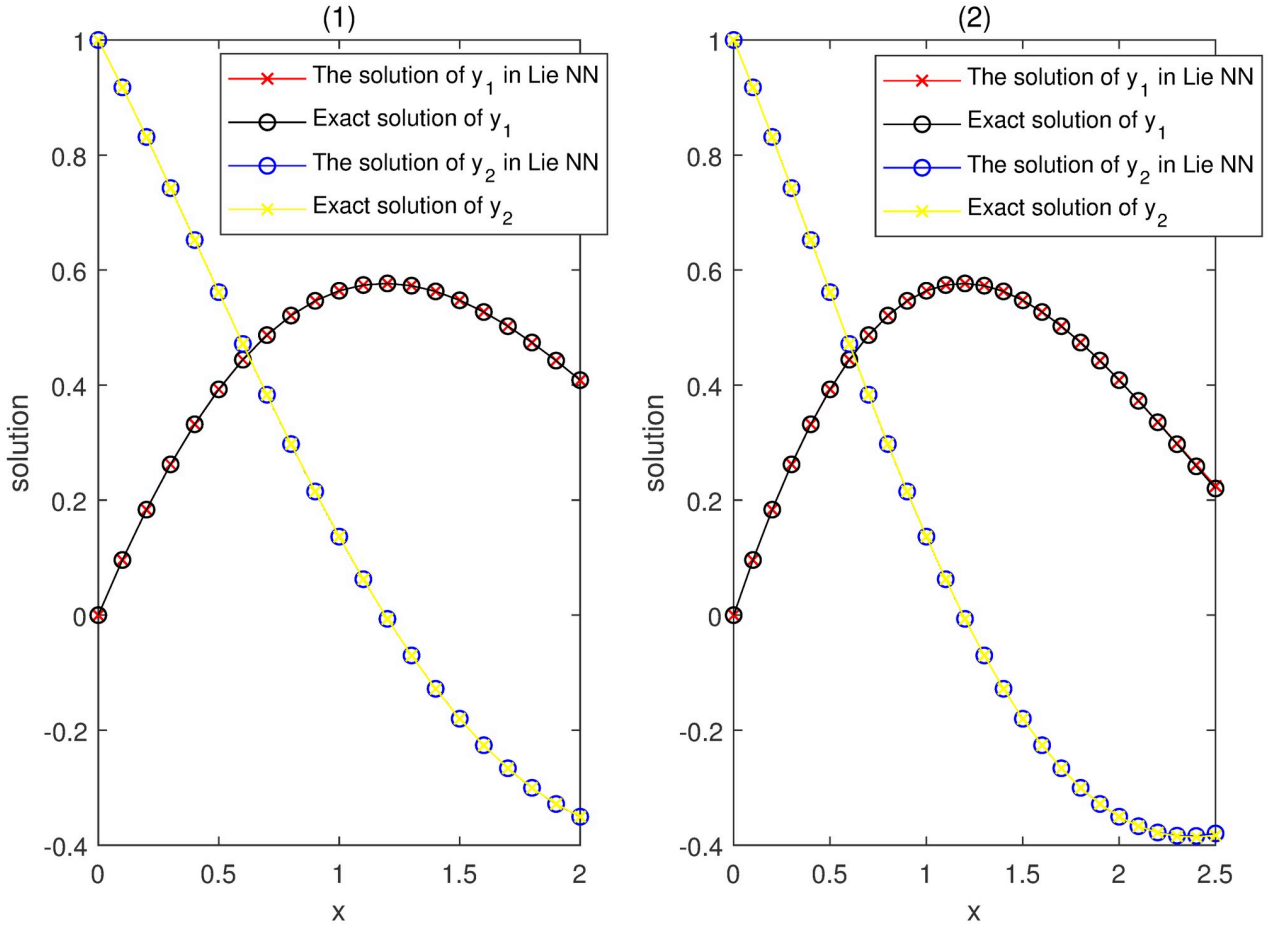

Fig 6. Comparison of the exact solutions y_i and our $\hat{y}_i$ in Example 2.

https://doi.org/10.1371/journal.pone.0265992.g006

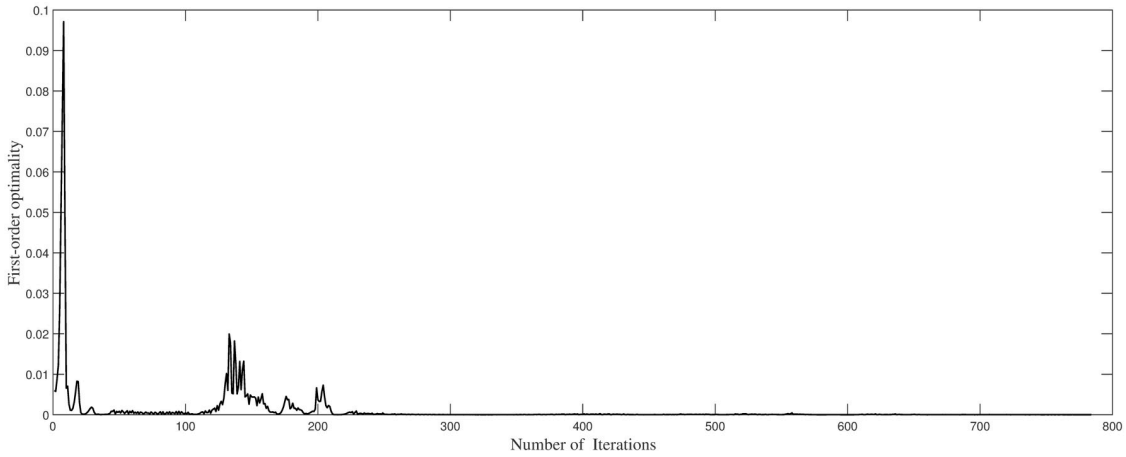

Fig 7. Trend of training the errors with iteration steps in Example 2.

https://doi.org/10.1371/journal.pone.0265992.g007

Example 3. Consider the following nonlinear initial value problem of Duffing equation [30]

$$u'' + u + 2\epsilon u^3 = 0 \tag{27}$$

with initial conditions $u(0) = 1$, $u'(0) = 0$ and $0 < \epsilon < 1$. In mechanics, it is solved by

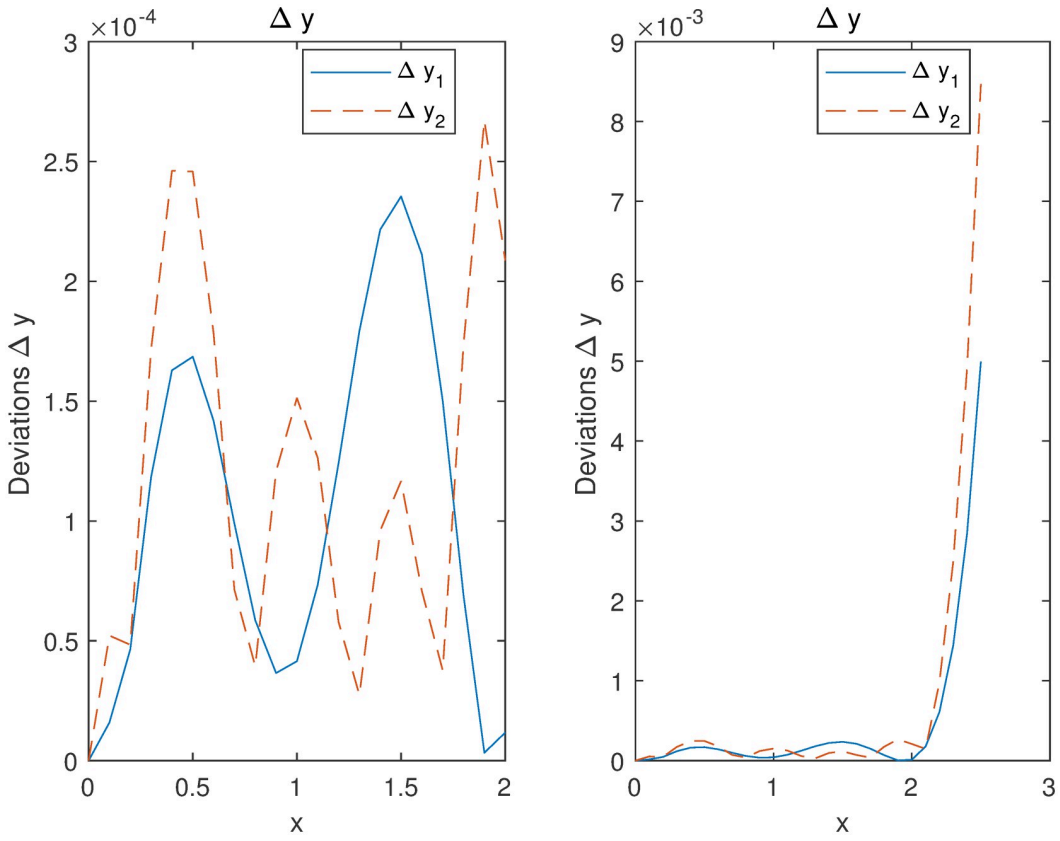

Fig 8. Accuracies of the $\hat{y}_{1,2}$ to exact ones $y_{1,2}$ in Example 2.

https://doi.org/10.1371/journal.pone.0265992.g008

Table 1. Robustness of our method.

	ϵ	Ave.Error	Ave.Deviation
Training Data (21 samples)	0.01	7.604×10^{-6}	1.958×10^{-3}
	0.2	4.736×10^{-7}	9.122×10^{-5}
	0.5	2.174×10^{-6}	3.166×10^{-5}
	0.8	1.646×10^{-6}	8.282×10^{-5}
Predict Data (21 samples)	0.01	6.671×10^{-6}	9.546×10^{-4}
	0.2	2.507×10^{-7}	6.366×10^{-5}
	0.5	1.268×10^{-6}	3.565×10^{-5}
	0.8	2.168×10^{-6}	6.601×10^{-5}
Test Data (26 samples)	0.01	2.318×10^{-5}	1.925×10^{-3}
	0.2	6.549×10^{-5}	5.018×10^{-4}
	0.5	2.806×10^{-4}	1.409×10^{-4}
	0.8	1.922×10^{-5}	1.870×10^{-4}

The Training Data are 21 sample points uniformly distributed in the training interval, the Predict Data are 21 sample points randomly distributed in the training interval, and the Test Data are 26 sample points uniformly distributed in the test interval.

https://doi.org/10.1371/journal.pone.0265992.t001

perturbation techniques, such as strightforward expansion, multiple scales, averaging methods, etc to analysis the different resonance phenomenons.

Here, as an application of our proposed method, we consider the case with parameter values $\epsilon = 0.5$.

One of the standard forms (9) for Eq (27) is

$$\begin{aligned} y_1'(t) &= y_2(t), \\ y_2'(t) &= -y_1(t) - y_1^3(t), \end{aligned} \tag{28}$$

with initial values $y_1(0) = 1, y_2(0) = 0$ by introducing variables $y_1 = u, y_2 = \dot{u}$. Obviously, we have operator $D = y_2\partial y_1 + (-y_1(x) - y_1^3(x))\partial y_2$. As before, one splits the operator into the sum of two parts and select first part as $D_1 = y_2\partial y_1 - y_1\partial y_2$. It produces the associated IVP

$$\dot{\bar{y}}_1 = \bar{y}_2, \ \ \dot{\bar{y}}_2 = -\bar{y}_1 - \bar{y}_1^3, \ \ \bar{y}_1(0) = 1, \bar{y}_2(0) = 0, \tag{29}$$

and the trial solutions $\hat{y}_1(t) = \bar{y}_1 + t\mathcal{N}_1(t;\theta)$ and $\hat{y}_2(t) = \bar{y}_2 + t\mathcal{N}_2(t;\theta)$ of (28) where $\bar{y}_1$ and $\bar{y}_2$ are solutions of (29). The IVP (29) has an obvious exact solution. However, it corresponds to (27) which is easily solved by the known numerical methods, such as Runge-Kutta method. This numerical solutions approximate the real solution more well than that given in literatures at initial point $t = 0$ shown in Fig 9.

By using the exact solutions as first part of the trial solutions, we train NNs for $\mathcal{N}_1(t;\theta)$ and $\mathcal{N}_2(t;\theta)$ with a grid of 21 equidistant training points in [0, 2]. Fig 10 compares the training results with the Runge-Kutta method. It shows that higher accuracy approximation and quicker convergence are made throughout the whole domain.

Considering Fig 11, the decreasing trend of train errors as iteration steps increasing is displayed showing the quick convergence and stability of our algorithm.

It shows that the proposed method can also capture the fluctuation wave properties of the solution to a strong nonlinear dynamics system.

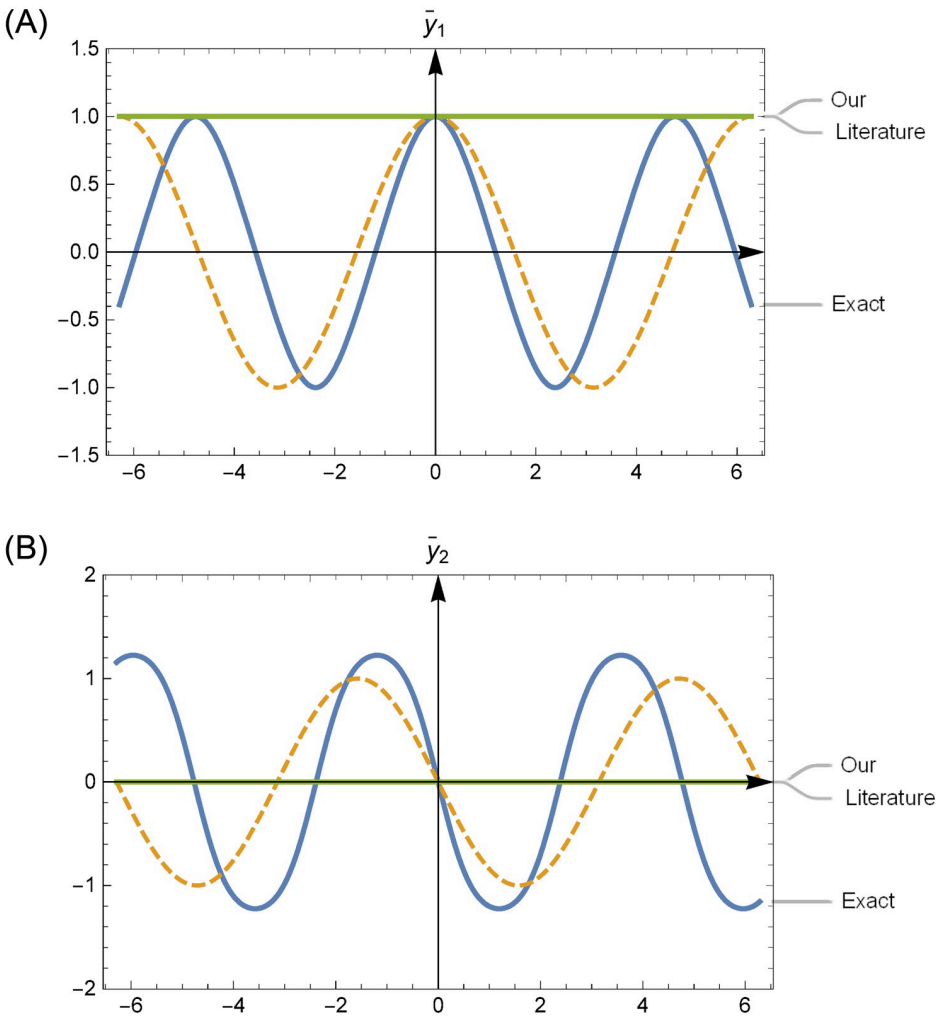

Fig 9. Comparison of the approximate efficiencies of first terms $\bar{y}_1, \bar{y}_2$ to the exact solution y_1, y_2 at initial point t = 0 in Example 3. A:Comparison of $\bar{y}_1$ and y_1 at the initial point $t = 0$. B:Comparison of $\bar{y}_2$ and y_2 at the initial point $t = 0$.

https://doi.org/10.1371/journal.pone.0265992.g009

5 Discussion and conclusions

In this study, we propose a new NN method based on the Lie group theory of ODEs to solve the IVP of ODEs. Examples are used to verify the effectiveness of the method, and the superiority of the method is proven by comparing the exact solutions and other numerical methods. The examples also prove that even when the network structure is small, this method can achieve higher accuracy solutions than that of the methods in [9]. Moreover, the combination of the Lie group and NN methods is easy to implement.

1. The success of the method can be attributed to two aspects. The first is the employment of the Lie group expression of the real solution to the IVP of ODEs. These expressions provide a two-part summation form of the solution. Considering the first part, the initial values can be easily determined before the network calculation stage and is independent of the trainable parameters. Moreover, the first part yields approximate solutions to the real solution to be determined at least in an interval of the initial point of the independent variable. These

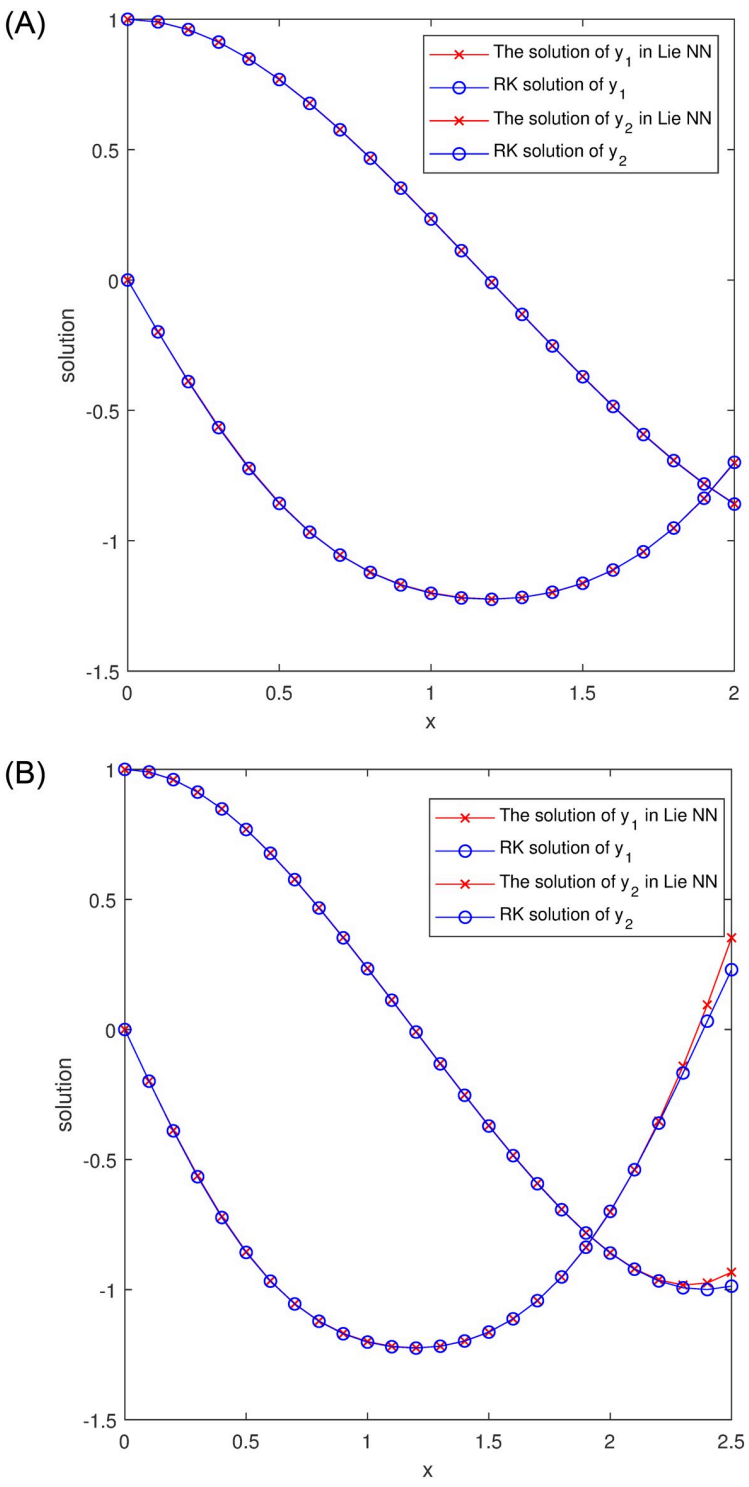

Fig 10. Comparison between our solutions $\hat{y}_i$ and Runge-Kutta results y_i in Example 3.

https://doi.org/10.1371/journal.pone.0265992.g010

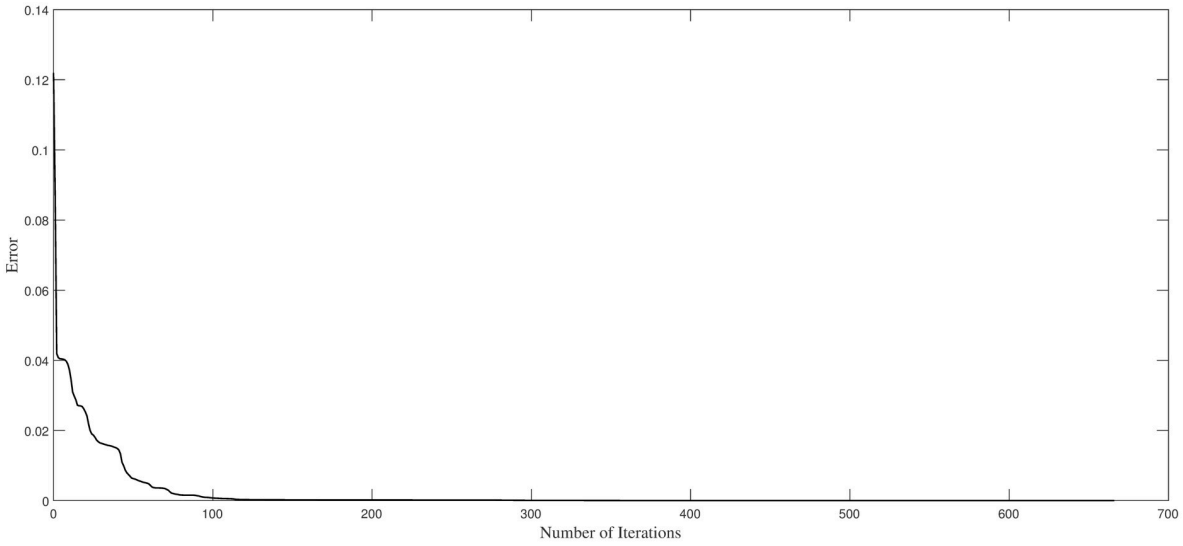

Fig 11. Trend of training errors with iteration steps in Example 3.

https://doi.org/10.1371/journal.pone.0265992.g011

features of the first part of the trial solution save much workload in machine learning, which results in higher efficiency of the method. The second is the use of NNs that are excellent function approximators to determine the second part of the solution. The training (learning) procedure is implemented by optimizing the loss functions derived from the original ODEs and the additional initial or boundary values satisfied by the trial solutions.

2. Contrary to most previous methods, the proposed method is more general and can be extended to solve various problems of ODEs and PDEs by the appropriate selection of Lie group expressions of trial solutions and the loss functions.

3. As indicated by the applications in the numerical experiments, the method exhibits excellent generalization and stability performance. This is because the deviations in the training and testing data are of lower values and are uniformly stable.

4. The NN architectures employed were one hidden layer with three neurons, indicating that the trainable parameters are fewer. This indicates that the effect of the NN architectures on the quality of the solution depends on the structure of the trial solutions. Moreover, this demonstrates the importance of using the additional information of solutions in the network method instead of directly using the network approach.

5. The applications also show that the method can be applied to strong nonlinear cases and more accurately detect the severe nonlinearities of the physical phenomena.

6. The randomness of the selection D_1 in the decomposition $D = D_1 + D_2$ may be a drawback. However, Theorem 1–2 proves that the solution expressions used always exist for any selection D_1. Hence, the method always works well at least in the accuracy range of previous similar methods. If the operator D_1 is selected well, then the efficiency of the method is ensured. Because the possible selections of D_1 are finite for a concrete IVP of ODEs, one of the efficiencies can always be selected after several times of computations. This leads to future studies of how to use more sophisticated theories to design an ANN algorithm for solving problems of DEs.

These benefits and the ideas in this study will stimulate future studies on solving DEs' problem by using the Lie theory of DEs and ANN.

Supporting information

S1 File. The result when ϵ takes different values.
(RAR)

S2 File.
(PDF)

Author Contributions

Conceptualization: Temuer Chaolu.

Methodology: Ying Wen.

Software: Ying Wen.

Writing – original draft: Ying Wen.

Writing – review & editing: Ying Wen, Temuer Chaolu, Xiangsheng Wang.

References

1. Rivertz HJ. On those ordinary differential equations that are solved exactly by the improved Euler method. Archivum Mathematicum. 2013; 49(1):29–34. https://doi.org/10.5817/AM2013-1-29
2. Kaps P, Rentrop P. Generalized Runge-Kutta methods of order four with stepsize control for stiff ordinary differential equations. Numerische Mathematik. 1979; 33(1):55–68. https://doi.org/10.1007/BF01396495
3. Noye B, Rankovic M. An accurate explicit finite diference technique for solving the one-dimensional wave equation. Communications in applied numerical methods. 1986; 2(6):557–561. https://doi.org/10.1002/cnm.1630020603
4. Yun Ys, Wen Y, Chaolu T, Rach R. A segmented Adomian algorithm for the boundary value problem of a second-order partial differential equation on a plane triangle area. Advances in Difference Equations. 2019; 2019(1):1–13. https://doi.org/10.1186/s13662-019-2329-4
5. Olver PJ. Applications of Lie groups to differential equations. vol. 107. Springer Science & Business Media; 2000.
6. Press WH, Press WH, Flannery BP, Teukolsky SA, Vetterling WT, Flannery BP, et al. Numerical recipes in Pascal: the art of scientific computing. vol. 1. Cambridge university press; 1989.
7. Dissanayake M, Phan-Thien N. Neural-network-based approximations for solving partial differential equations. communications in Numerical Methods in Engineering. 1994; 10(3):195–201. https://doi.org/10.1002/cnm.1640100303
8. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural networks. 1989; 2(5):359–366. https://doi.org/10.1016/0893-6080(89)90020-8
9. Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations. IEEE transactions on neural networks. 1998; 9(5):987–1000. https://doi.org/10.1109/72.712178 PMID: 18255782
10. Effati S, Pakdaman M. Artificial neural network approach for solving fuzzy differential equations. Information Sciences. 2010; 180(8):1434–1457. https://doi.org/10.1016/j.ins.2009.12.016
11. He S, Reif K, Unbehauen R. Multilayer neural networks for solving a class of partial differential equations. Neural networks. 2000; 13(3):385–396. https://doi.org/10.1016/S0893-6080(00)00013-7 PMID: 10937971
12. Li-ying X, Hui W, Zhe-zhao Z. The algorithm of neural networks on the initial value problems in ordinary differential equations. In: 2007 2nd IEEE Conference on Industrial Electronics and Applications. IEEE; 2007:813–816.
13. Tsoulos IG, Gavrilis D, Glavas E. Solving differential equations with constructed neural networks. Neurocomputing. 2009; 72(10-12):2385–2391. https://doi.org/10.1016/j.neucom.2008.12.004

14. Beidokhti RS, Malek A. Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques. Journal of the Franklin Institute. 2009; 346 (9):898–913. https://doi.org/10.1016/j.jfranklin.2009.05.003

15. Chakraverty S, Mall S. Regression-based weight generation algorithm in neural network for solution of initial and boundary value problems. Neural Computing and Applications. 2014; 25(3):585–594. https://doi.org/10.1007/s00521-013-1526-4

16. Mall S, Chakraverty S. Application of Legendre neural network for solving ordinary differential equations. Applied Soft Computing. 2016; 43:347–356. https://doi.org/10.1016/j.asoc.2015.10.069

17. Dockhorn T. A discussion on solving partial differential equations using neural networks. arXiv preprint arXiv:190407200. 2019.

18. Li S, Wang X. Solving ordinary differential equations using an optimization technique based on training improved artificial neural networks. Soft Computing. 2021; 25(5):3713–3723. https://doi.org/10.1007/s00500-020-05401-w

19. Weinan E, Han J, Jentzen A. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Communications in Mathematics and Statistics. 2017; 5(4):349–380. https://doi.org/10.1007/s40304-017-0117-6

20. Wang L., Yan Z. Data-driven peakon and periodic peakon solutions and parameter discovery of some nonlinear dispersive equations via deep learning. Physica D: Nonlinear Phenomena. 2021; 428:133037. https://doi.org/10.1016/j.physd.2021.133037

21. Raissi M, Perdikaris P, Karniadakis GE. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:171110561. 2017.

22. Zhou Z., Yan Z. Solving forward and inverse problems of the logarithmic nonlinear schrödinger equation with pt-symmetric harmonic potential via deep learning. Physics Letters A. 2021; 387:127010. https://doi.org/10.1016/j.physleta.2020.127010

23. Bluman GW, Cheviakov AF, Anco SC. Construction of Mappings Relating Differential Equations. In: Applications of Symmetry Methods to Partial Differential Equations. Springer; 2010:121–186.

24. Ovsyannikov LV. Lectures on the theory of group properties of differential equations. World Scientific Publishing Company; 2013.

25. Chaolu T, Bluman G. An algorithmic method for showing existence of nontrivial non-classical symmetries of partial differential equations without solving determining equations. Journal of Mathematical Analysis and Applications. 2014; 411(1):281–296. https://doi.org/10.1016/j.jmaa.2013.09.040

26. Gröbner W, Knapp H. Contributions to the method of Lie series. vol. 802. Bibliographisches Institut Mannheim; 1967.

27. Shokin II, Ianenko N. The method of differential approximation: application to gas dynamics. Novosibirsk Izdatel Nauka. 1985.

28. Dorodnitsyn V. Transformation groups in net spaces. Journal of Soviet mathematics. 1991; 55 (1):1490–1517. https://doi.org/10.1007/BF01097535

29. Sun JQ, Ma ZQ, Hua W, Qin MZ. New conservation schemes for the nonlinear Schrödinger equation. Applied mathematics and computation. 2006; 177(1):446–451. https://doi.org/10.1016/j.amc.2005.11.021

30. Nayfeh AH. Introduction to perturbation techniques. John Wiley & Sons; 2011.