axioms

Article

Study of Burgers–Huxley Equation Using Neural Network Method

Ying Wen [1,*,†] and Temuer Chaolu [2,†]

[1] College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China
[2] College of Sciences and Arts, Shanghai Maritime University, Shanghai 201306, China; tmchaolu@shmtu.edu.cn
* Correspondence: 201840310002@stu.shmtu.edu.cn
† These authors contributed equally to this work.

Abstract: The study of non-linear partial differential equations is a complex task requiring sophisticated methods and techniques. In this context, we propose a neural network approach based on Lie series in Lie groups of differential equations (symmetry) for solving Burgers–Huxley nonlinear partial differential equations, considering initial or boundary value terms in the loss functions. The proposed technique yields closed analytic solutions that possess excellent generalization properties. Our approach differs from existing deep neural networks in that it employs only shallow neural networks. This choice significantly reduces the parameter cost while retaining the dynamic behavior and accuracy of the solution. A thorough comparison with its exact solution was carried out to validate the practicality and effectiveness of our proposed method, using vivid graphics and detailed analysis to present the results.

Keywords: Burgers–Huxley equation; optimization; neural network method; Lie groups; Lie series

MSC: 65M99

Citation: Wen, Y.; Chaolu, T. Study of Burgers–Huxley Equation Using Neural Network Method. *Axioms* **2023**, *12*, 429. https://doi.org/10.3390/axioms12050429

Academic Editor: Azhar Ali Zafar and Nehad Ali Shah

Received: 15 March 2023
Revised: 20 April 2023
Accepted: 25 April 2023
Published: 26 April 2023

1. Introduction

Partial differential equations (PDEs) are ubiquitous and fundamental to understanding and modeling the complexities of natural phenomena. From mathematics to physics to economics and beyond, PDEs play a critical role in virtually all fields of engineering and science [1–3]. Through their mathematical representation of physical phenomena, PDEs provide a powerful means of gaining insight into complex systems, enabling researchers and engineers to predict behavior and uncover hidden relationships. However, solving PDEs can be a daunting and challenging task. The complexity of these equations often requires sophisticated numerical methods that must balance accuracy and efficiency while solving high-dimensional PDEs. Despite these challenges, PDEs remain a cornerstone of modern science, enabling researchers to unlock discoveries and technological advancements across disciplines.

As numerical and computational techniques continue to rapidly develop, the study of PDEs has become increasingly vital. In recent years, advances in numerical methods and high-performance computing techniques have made it possible to solve complex PDEs more accurately and efficiently than ever before. These new tools can precisely solve specific problems across a broader range of equations while simultaneously computing data faster, reducing the time and cost of solving pending problems. Moreover, these new techniques have allowed researchers to gain deeper insights into the physical meaning behind PDEs, enabling them to revisit natural phenomena from fresh perspectives and explore those that prove challenging to explain by traditional methods. This has led to groundbreaking research discoveries and innovations in various fields of science and engineering.

Machine learning methods [4,5], particularly in the area of artificial neural networks (ANNs) [6,7], have piqued considerable interest in recent years due to their potential to solve differential equations. ANNs are well-known for their exceptional approximation capabilities and have emerged as a promising alternative to traditional algorithms [8]. These methods have a significantly smaller memory footprint and generate numerical solutions that are both closed and continuous over the integration domain without requiring interpolation. ANNs have been applied to differential equations, including ordinary differential equations (ODEs) [9,10], PDEs [11,12], and stochastic differential equations (SDEs) [13,14], making them a valuable tool for researchers and engineers alike. Neural networks have become a powerful and versatile tool for solving differential equations due to their ability to learn intricate mappings from input–output data, further cementing their role as a critical component in the machine learning fields.

In recent years, the application of neural networks in solving differential equations has gained significant attention in the scientific community. One prominent model is the neural ordinary differential equations, which approximates the derivative of an unknown solution using neural networks, parameterizing the derivatives of the hidden states of the network with the help of the differential equation, thus creating a new type of neural network [15]. Another approach is the deep Galerkin method [16], which uses neural networks to approximate the solution of the differential equation in a bid to minimize error. Gorikhovskii et al. [17] introduced a practical approach for solving ODEs using neural networks in the TensorFlow machine-learning framework. In addition, Huang et al. [18] introduce an additive self-attention mechanism to the numerical solution of differential equations based on the dynamical system perspective of the residual neural network.

By utilizing neural network functions to approximate the solutions, neural networks have also been used to solve PDEs. The physics-informed neural network (PINN) method uses the underlying physics of the problem to incorporate constraints into the solution of the neural network, resulting in successful applications to various PDEs such as the Burgers and Poisson equations [19]. Compared to traditional numerical methods, PINNs offer several advantages, including higher accuracy and more efficient computation. Berg et al. [20] introduced a new deep learning-based approach to solve PDEs on complex geometries. They use a feed-forward neural network and an unconstrained gradient-based optimization method to predict PDE solutions. Furthermore, Another exciting development in the field of neural networks and PDEs is the use of convolutional neural networks (CNNs). Ruthotto et al. [21] used a CNN to learn to solve elliptic PDEs and incorporated a residual block structure to improve network performance. Quan et al. [22] presented an innovative approach to addressing the challenge of solving diffusion PDEs, by introducing a novel learning method built on the foundation of the extreme learning machine algorithm. By leveraging this advanced technique, the parameters of the neural network are precisely calculated by solving a linear system of equations. Furthermore, the loss function is ingeniously constructed from three crucial components: the PDE, initial conditions, and boundary conditions. Tang et al. [23] demonstrate through numerical cases that the proposed depth adaptive sampling (DAS-PINNs) method can be used for solving PDEs. Overall, the advancements made in the domain of neural networks have revolutionized how we approach solving complex PDEs in unimaginable ways. These developments suggest that neural networks are a promising tool for solving complex PDEs and that there is great potential for further research and innovation in this area.

This paper proposes a novel approach for solving the Burgers–Huxley equation , which uses a neural network based on the Lie series in the Lie groups of differential equations, adding initial or boundary value terms to the loss function to approximate the solution of the equation by minimization. Slavova et al. [24] constructed a cellular neural network model to study the Burgers–Huxley equation. Shagun et al. [25] employed a feed-forward neural network to solve the Burgers-Huxley equation and investigated the impact of the number of training points on the accuracy of the solution. Kumar et al. [26] proposed a deep learning algorithm based on the deep Galerkin method for solving the

Burgers–Huxley equation, which outperformed traditional numerical methods. These studies demonstrate the potential of neural networks in solving differential equations. Nonetheless, it is simple to ignore the underlying nature of these equations, in other words, to fail to capture the nonlinear nature of the equations, which is essential to comprehend the behavior of complex systems. To address this issue, the aim of our proposed method is to approximate the solution of the differential equations by combining the Lie series in Lie groups of differential equations and the power of neural networks. Our proposed method accurately simulates the physical behavior of complicated systems, and the first part of the constructed solution has well captured the nonlinear nature of the equation while reducing the parameter cost of the subsequent neural network and by minimizing the loss function, making the solution converge quickly by introducing initial or boundary value terms required for exact approximation. This work demonstrates the effectiveness of combining neural networks with Lie series to solve differential equations and provides insights into the physical behavior of complex dynamical systems.

The essay is set up as follows. The basic framework and fundamental theory of neural network algorithms based on Lie series in Lie groups of differential equations are introduced in Section 2. The specific steps for the Lie-series-based neural network method to solve the Burgers–Huxley equation are described in Section 3. The method is also applied to the Burgers–Fisher equation and the Huxley equation. Summary and outlook are presented in Section 4.

2. Basic Idea of a Lie-Series-Based Neural Network Algorithm

2.1. Differential Forms and Lie Series Solution

With respect to the Lie group transformation of the parameter ε,

$$u^* = T(\varepsilon; u) \in G, \quad u^*(0) = u \tag{1}$$

where G is a Lie group, and ε is a group parameter.

By employing Taylor expansion about neighborhood of $\varepsilon = 0$,

$$u^* = T(\varepsilon; u) = u + \left.\frac{\partial T(\varepsilon; u)}{\partial \varepsilon}\right|_{\varepsilon=0} \varepsilon + O\left(\varepsilon^2\right). \tag{2}$$

Then, $u^* = u + \varepsilon\zeta$ is known as the infinitesimal transformation. $D = \zeta(u)\partial u$ is called the infinitesimal operator, where $\zeta(u) = \left.\frac{\partial T(\varepsilon;u)}{\partial \varepsilon}\right|_{\varepsilon=0}$.

The following differential equation is given

$$u' = F(\xi, u), \quad u(0) = u_0 \tag{3}$$

$F(\xi, u)$ is a differentiable function, and if (2) is a symmetry of (3), then it has a Lie series solution to the initial value problem (3) and can be written as [27]

$$u = e^{\xi D} u|_{\xi=0} \tag{4}$$

2.2. Algorithm of a Lie-Series-Based Neural Network

The idea of Lie groups is based on the study of continuous symmetry, which at first may seem abstract and complex. However, in the realm of solving differential equations, Lie group methods are a unique approach that goes beyond traditional mathematical techniques. Lie series in the Lie transform groups of differential equations can be used to construct approximate solutions of PDEs and to study their symmetries and other properties. Lie series provide a powerful framework for studying the behavior of differential equations and have many important applications in various fields of science and engineering.

From [28], it is known that

$$D = D_1 + D_2 \tag{5}$$

The solution of (3) can be written as $u = e^{\xi D}u|_{\xi=0} = e^{\xi(D_1+D_2)}u|_{\xi=0}$.

Theorem 1. *$\bar{u}(\xi;u) = e^{\xi D_1}u|_{\xi=0}$, $\xi \in \mathbb{R}^n$, is the decomposition part of D. The solution of problem (3) belonging to D expanded as follows:*

$$u = \bar{u}(\xi;u) + \int_0^{\xi} D_2\left(e^{(\xi-\tau)D}u\right)|_{u\to\bar{u}(\tau;u)}d\tau \tag{6}$$

The proof is given below and is detailed in the literature [28].

Proof.

$$\begin{aligned} u = e^{\xi D}u = e^{\xi(D_1+D_2)}u = \sum_{v=0}^{\infty}\frac{\xi^v}{v!}D_1^v u + \sum_{v=1}^{\infty}\frac{\xi^v}{v!}D_1^{v-1}D_2 u \\ + \sum_{v=2}^{\infty}\frac{\xi^v}{v!}D_1^{v-2}D_2 D u + \ldots + \sum_{v=\alpha}^{\infty}\frac{\xi^v}{v!}D_1^{v-\alpha}D_2 D^{\alpha-1}u + \ldots \end{aligned} \tag{7}$$

It is known that

$$\frac{\xi^v}{v!} = \int_0^{\xi}\frac{(\xi-\tau)^{\alpha-1}}{(\alpha-1)!}\frac{\tau^{v-\alpha}}{(v-\alpha)!}d\tau, \quad (v \ge \alpha \ge 1, integers)$$

Equation (7) is rewritten as

$$\begin{aligned} u = \bar{u} + \int_0^{\xi}\sum_{v=0}^{\infty}\frac{\tau^v}{v!}D_1^v D_2 u d\tau + \int_0^{\xi}(\xi-\tau)\sum_{v=0}^{\infty}\frac{\tau^v}{v!}D_1^v D_2 D u d\tau + \ldots \\ + \int_0^{\xi}\frac{(\xi-\tau)^{\alpha-1}}{(\alpha-1)!}\sum_{v=0}^{\infty}\frac{\tau^v}{v!}D_1^v D_2 D^{\alpha-1}u d\tau + \ldots \end{aligned}$$

From the form of the series solution [27], it follows that

$$\sum_{v=0}^{\infty}\frac{\tau^v}{v!}D_1^v\left(D_2 D^{\alpha-1}u\right) = \left(D_2 D^{\alpha-1}u\right)_{u\to\bar{u}(\tau;u)}$$

Hence,

$$u = \bar{u} + \sum_{\alpha=1}^{\infty}\int_0^{\xi}\frac{(\xi-\tau)^{\alpha-1}}{(\alpha-1)!}\left(D_2 D^{\alpha-1}u\right)_{u\to\bar{u}(\tau;u)}d\tau$$

after commuting the signs of the series and the integral which is allowed within the circle of absolute convergence, the formula

$$u = \bar{u} + \int_0^{\xi}\left(D_2\sum_{\alpha=0}^{\infty}\frac{(\xi-\tau)^{\alpha}}{\alpha!}D^{\alpha}u\right)_{u\to\bar{u}(\tau;u)}d\tau \tag{8}$$

is obtained, which may also be written as follows:

$$e^{\xi D}u = e^{\xi D_1}u + \int_0^{\xi}\left(D_2 e^{(\xi-\tau)D}u\right)_{u\to\bar{u}(\tau;u)}d\tau \tag{9}$$

□

The complexities inherent in the integration of the second component, as elucidated by Equation (6), necessitates a sophisticated approach to computation. To tackle this daunting challenge head-on, as elaborated in the reference [29] of our previous work, the functional form of the neural network is utilized to simplify this part and ensure the accuracy of our results.

From [29], $\hat{u} = e^{\xi D}u|_{\xi=0} = \bar{u} + \xi N(\theta;\xi)$. The determination of $\bar{u}$ from the equation $\bar{u}' = D_1\bar{u}$ is inspired by the idea of the Lie series solution of the first-order ODE, where the initial value of $\bar{u}(0) = u(0) = u_0$ is kept constant throughout the process, ensuring the reliability and truthfulness of our results. $N(\theta;\xi)$ is a single output neural network with a single input of ξ, the parameter θ consists of the weight $\mathbf{W}$ and the bias $\mathbf{b}$. The Algorithm 1 is described in detail below, as shown in Figure 1.

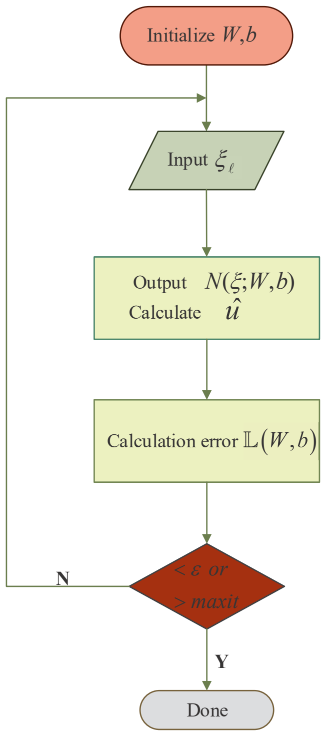


Figure 1. Flow chart of Lie-series-based neural network algorithm.

Algorithm 1: A Lie-series-based neural network algorithm for problem (3)

```
Require Determine the operator $D$ according to (3), and solve it with the decomposed part $D_1$ to obtain $\bar{u}$.
Begin
  1. Consider a uniformly spaced distribution of discrete points within the initial condition $\xi_\ell(\ell = 1,2,\dots,\lambda)$.
  2. Determining the structure of a neural network. (The number of hidden layers and the number of neurons, the selection of the activation function $\sigma$.)
  3. Initialization of the neural networks parameters $\mathbf{W}$, $\mathbf{b}$.
  4. Get $\hat{u} = \bar{u} + \xi N(\theta;\xi)$ and substitute back into (3).
  5. Minimize the loss function $\mathbb{L}(\theta)$.
  6. Update the parameter $\theta$ so that $\hat{u}$ approximates the solution $u$ of problem (3).
End
```

In general, the loss function $\mathbb{L}(\theta)$ is defined as follows:

$$\begin{aligned}\mathbb{L}(\theta) &= \mathbb{L}_F + \mathbb{L}_I \\ &= \frac{1}{\lambda}\sum_{\ell=1}^{\lambda}\sum_{i=1}^{n}\left\| \frac{\partial}{\partial\xi}\hat{u}_i(\xi,\theta)\Big|_{\xi=\xi_\ell} - F_i(\hat{u}_1,\hat{u}_2,\ldots,\hat{u}_i)\right\|_2^2 \\ &+ \frac{1}{2}\sum_{l=1}^{p}\sum_{i=1}^{n}\left\|\left(\hat{u}_i(\xi,\theta)|_{\xi=\xi_l} - K(\xi)|_{\xi=\xi_l}\right)\right\|_2^2\end{aligned} \tag{10}$$

as additional terms with $K(\xi_l)$, $l = 1, 2, \ldots, p$ as initial value or boundary conditions. The $\mathbb{L}_F$ part of the loss function is derived by substituting the network solution $\hat{u}$ into the mean squared error generated on both sides of the problem (3). In addition, the mean squared error generated by the network solution $\hat{u}$ under the initial or boundary value terms are also used to derive the $\mathbb{L}_I$ component of our loss function. By constructing the components of $\mathbb{L}_F$ and $\mathbb{L}_I$, we can satisfy both the differential equations and the initial values or boundary conditions of the problem under study.

The above algorithm also applies to the system of differential equations $\frac{du_i}{d\xi} = F_i(u_1, u_2, \ldots, u_n)$, $u_i(0) = \alpha_i \in \mathbb{R}^1, i = 1, 2, \ldots, n$, where $D = \sum_{i=1}^{n} F_i(u_i)\frac{\partial}{\partial u_i}$. For higher-order ODEs or PDEs, the above form can also be transformed with the help of some transformations or calculations.

2.3. The General Structure of the Neural Network

As depicted in Figure 2, our study delves into the complexities of multilayer perceptrons and their unique characteristics, with a particular emphasis on those with a single input unit, m hidden layers of H neurons, a neural network with activation function σ in the hidden layer, and a linear output unit. We present a detailed analysis of this neural network architecture. Specifically, for a given input vector $\xi_\ell(\ell = 1, 2, \ldots, \lambda)$, the output of the network $N = \sum_{i=1}^{H} \mathbf{W}^{m+1}\sigma(Z_i^m) + \mathbf{b}^{m+1}$, $Z_i^m = \sum_{j=1}^{H} w_{ji}^m \sigma(Z_j^{m-1}) + b_i^m$, where w_{ji}^m is the weight of the jth neuron in layer $m-1$ to the ith neuron in layer m, and b_i^m is the bias of the ith neuron in layer m. It can be seen that $Z_1^1 = w_{11}^1\xi_\ell + b_1^1$. In this paper, the activation function σ is chosen $\tanh(Z) = \frac{e^Z - e^{-Z}}{e^Z + e^{-Z}}$.

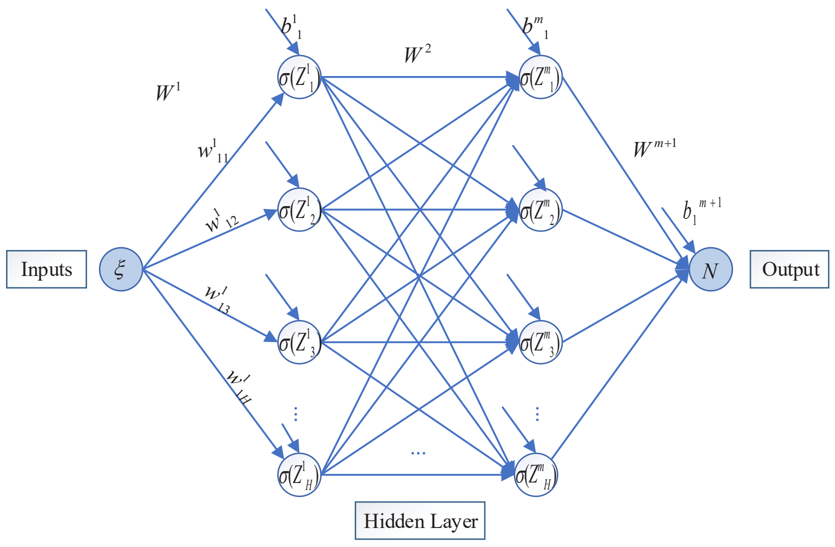


Figure 2. Neural network structure.

3. Lie-Series-Based Neural Network Algorithm for Solving Burgers Huxley Equation

The generalized Burgers–Huxley equation [30] is a nonlinear PDE that describes the propagation of electrical impulses in excitable media, such as nerve and muscle cells. It is a widely used mathematical framework for modeling intricate dynamical phenomena and has been instrumental in advancing research across multiple domains including physics, biology, economics, and ecology. The equation takes the form

$$\frac{\partial u}{\partial t} + \alpha u^{\delta}\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} = \beta u\left(1 - u^{\delta}\right)\left(\eta u^{\delta} - \lambda\right) \tag{11}$$

where α, β, λ, η are constants and δ is a positive constant.

When $\alpha = -1$, $\beta = 1$, $\lambda = 1$, $\eta = 1$, $\delta = 1$, the Burgers–Huxley equation is as follows:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + u\frac{\partial u}{\partial x} + u(1-u)(u-1), \quad u(0,x) = \frac{1}{2}\left(1 - \tanh\frac{x}{4}\right) \tag{12}$$

The exact solution of (12) is $u(t,x) = \frac{1}{2}\left(1 - \tanh\left(\frac{x}{4} + \frac{3t}{8}\right)\right)$. Using the traveling wave transform $\xi = x - ct$, problem (12) is transformed into an ODE, $u'' + cu' + uu' + u(1-u)(u-1) = 0$. Naturally, it is transformed into the form of the following system of ODEs

$$u_1' = u_2,\ u_2' = \frac{3}{2}u_2 - u_1u_2 - u_1(1-u_1)(u_1-1) \tag{13}$$

with $c = -\frac{3}{2}$, $u_1(\xi) = u$ and initial values $u_1(0) = \frac{1}{2}$, $u_2(0) = -\frac{1}{8}$.

In this study, we address the problem of solving the Burgers–Huxley equation using a Lie-series-based neural network algorithm. The operator $D = u_2\partial_{u_1} + (\frac{3}{2}u_2 - u_1u_2 - u_1(1-u_1)(u_1-1))\partial_{u_2}$ of (13) is chosen as $D_1 = u_2\partial_{u_1} + \frac{3}{2}u_2\partial_{u_2}$, and the solution of the corresponding initial value problem is $\bar{u}_1(\xi) = \frac{1}{12}\left(7 - \cosh\left(\frac{3\xi}{2}\right) - \sinh\left(\frac{3\xi}{2}\right)\right)$, $\bar{u}_2(\xi) = -\frac{1}{8}\cosh\left(\frac{3\xi}{2}\right) - \frac{1}{8}\sinh\left(\frac{3\xi}{2}\right)$. The solution of this part has been able to capture the nonlinear nature of the equation within a certain range, as shown in Figure 3. To minimize the loss function $\mathbb{L}(\theta)$, we employ two structurally identical neural networks and boundary value terms, each with 30 neurons in a single hidden layer, and the input $\xi_\ell(\ell = 1, 2, \ldots, 100)$ is 100 training points spaced equally in the interval $[-5, 3]$, making $\hat{u}_1(\xi)$ as close as possible to the exact solution $u(\xi)$ of the equation. The generalization ability of the neural network was confirmed in 120 test points at equidistant intervals of $\xi_\ell \in [-5, 3.3]$. The Lie-series-based neural network algorithm solves the Burgers–Huxley equation model as shown in Figure 4. Furthermore, we demonstrate the ability of neural networks to fit the training and test sets in Figure 5. By plotting the loss function $\mathbb{L}(\theta) = \mathbb{L}_F + \mathbb{L}_I$ against the number of iterations in Figure 6, where $\mathbb{L}_F = \frac{1}{\lambda}\sum_{\ell=1}^{\lambda}\Big(\left(\hat{u}_1'(\xi_\ell) - \hat{u}_2(\xi_\ell)\right)^2 + \big(\hat{u}_2'(\xi_\ell) - \frac{3}{2}\hat{u}_2(\xi_\ell) + \hat{u}_1(\xi_\ell)\hat{u}_2(\xi_\ell) + \hat{u}_1(\xi_\ell)(1 - \hat{u}_1(\xi_\ell))$ $(\hat{u}_1(\xi_\ell) - 1)\big)^2\Big)$, and $\mathbb{L}_I = \frac{1}{2}(\hat{u}_1(-5) - u(-5))^2 + \frac{1}{2}(\hat{u}_2(-5) - u'(-5))^2$, $\lambda = 100$. Some 1100 iterations later, $\mathbb{L}(\theta) = 3.042 \times 10^{-8}$.

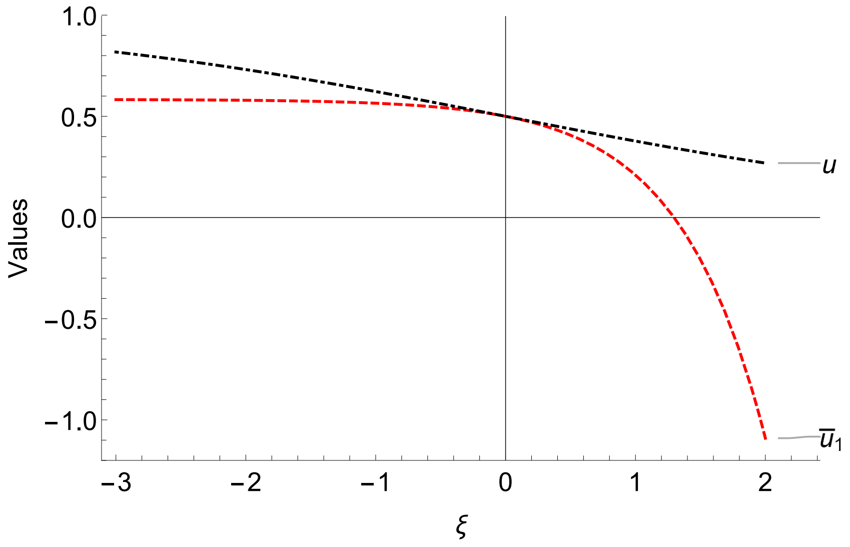
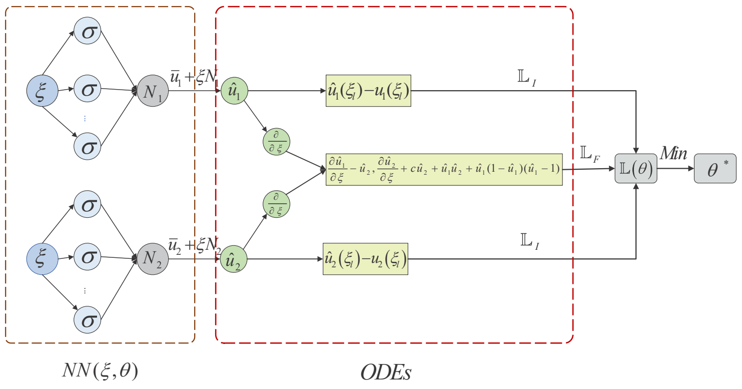


Figure 3. Comparison of the $\bar{u}_1(\xi)$ solution of the Burgers–Huxley equation with the exact solution $u(\xi)$.

We compare the solution $\hat{u}(t,x)$ containing the neural network training and the exact solution $u(t,x)$ in the interval $t \in [0,1]$, $x \in [-5,2]$ in the upper panel of Figure 7. Additionally, the lower panel displays the behavior of the solution at $t = 0.3, 0.5, 0.8$, demonstrating the solitary wave solution of the Burgers–Huxley equation. The contour plots for solution $\hat{u}_1(t,x)$ and the exact solution $u(t,x)$ are shown in Figure 8, further illustrating the accuracy of our proposed algorithm.



ξ
σ
N_1
N_2
$\bar{u}_1 + \xi N_1$
$\bar{u}_2 + \xi N_2$
$\hat{u}_1$
$\hat{u}_2$
$\frac{\partial}{\partial \xi}$
$\hat{u}_1(\xi_l) - u_1(\xi_l)$
$\frac{\partial \hat{u}_1}{\partial \xi} - \hat{u}_2, \frac{\partial \hat{u}_2}{\partial \xi} + c\hat{u}_2 + \hat{u}_1\hat{u}_2 + \hat{u}_1(1-\hat{u}_1)(\hat{u}_1 - 1)$
$\hat{u}_2(\xi_l) - u_2(\xi_l)$
$\mathbb{L}_I$
$\mathbb{L}_F$
$\mathbb{L}(\theta)$
Min
θ^*
$NN(\xi,\theta)$
ODEs

Figure 4. Schematic diagram of a Lie series-based neural network algorithm for solving Burgers–Huxley equation.

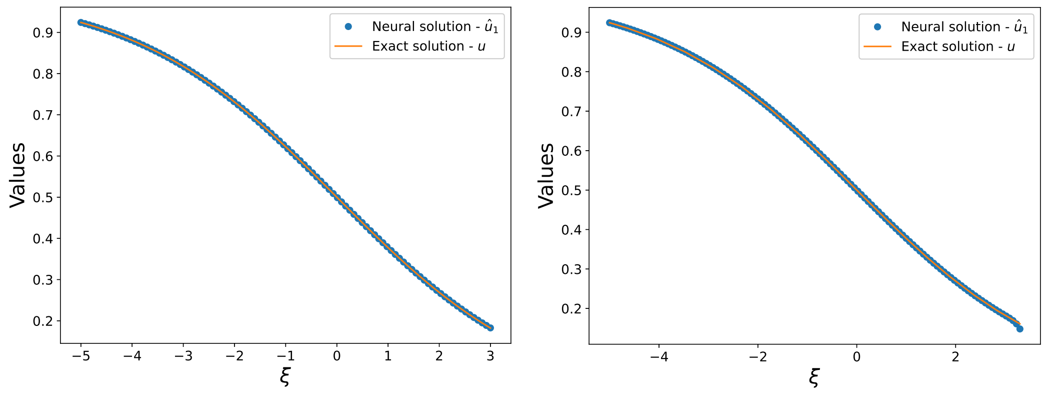


Figure 5. (**Left**) Comparison of solution $\hat{u}_1(\xi)$ with the exact solution $u(\xi) = \frac{1}{2}\left(1 - \tanh\left(\frac{\xi}{4}\right)\right)$ of (13) in the training set. (**Right**) Comparison of solution $\hat{u}_1(\xi)$ with the exact solution $u(\xi) = \frac{1}{2}\left(1 - \tanh\left(\frac{\xi}{4}\right)\right)$ of (13) in the test set.

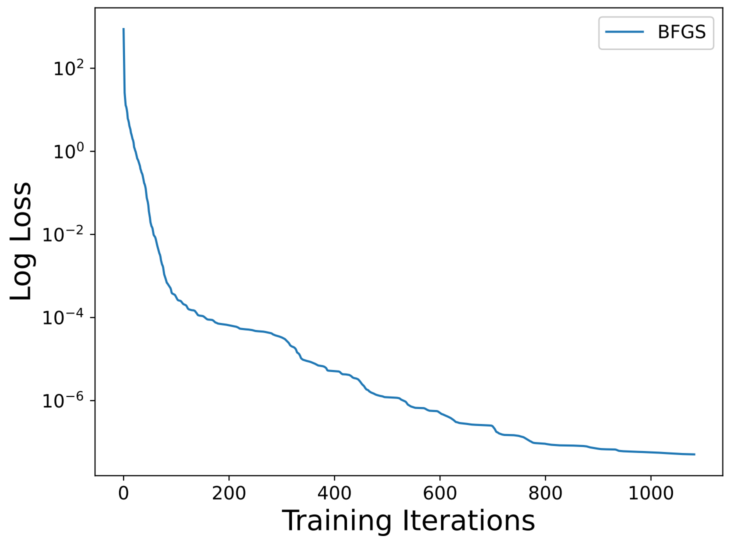


Figure 6. Curves of Loss function versus number of iterations for Burgers–Huxley equation.

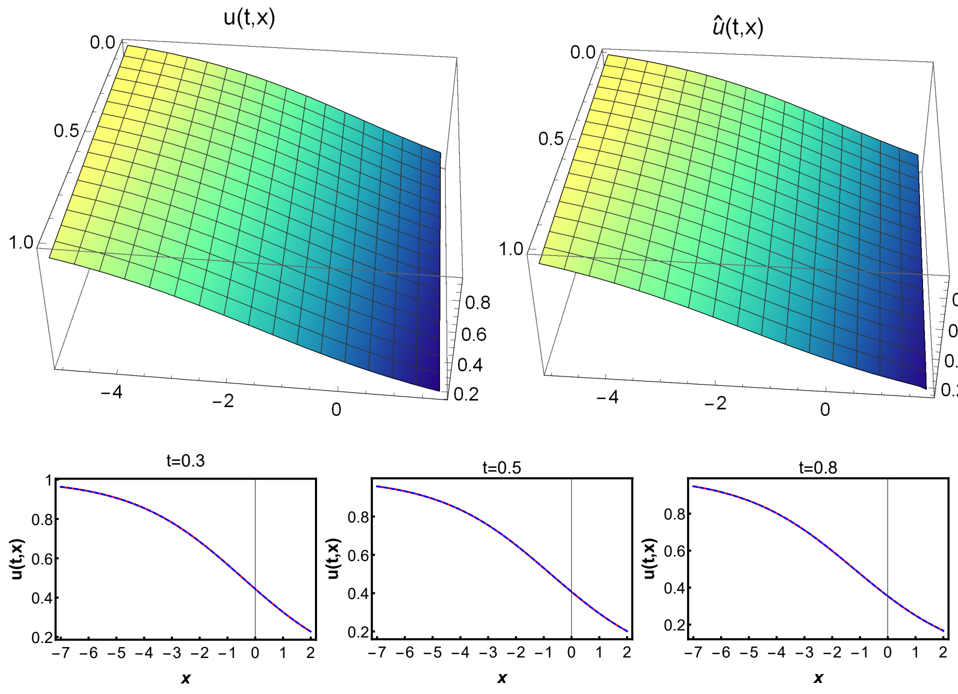


Figure 7. (**Top**) The true solution $u(t,x) = \frac{1}{2}\left(1 - \tanh\left(\frac{x}{4} + \frac{3t}{8}\right)\right)$ of the Burgers–Huxley equation is on the left, the predicted solution $\hat{u}(t,x)$ is on the right. (**Bottom**) Comparison of predicted and exact solutions at time $t = 0.3$, 0.5, and 0.8. (The dashed blue line indicates the exact solution $u(t,x)$, and the solid red line indicates the predicted solution $\hat{u}(t,x)$).

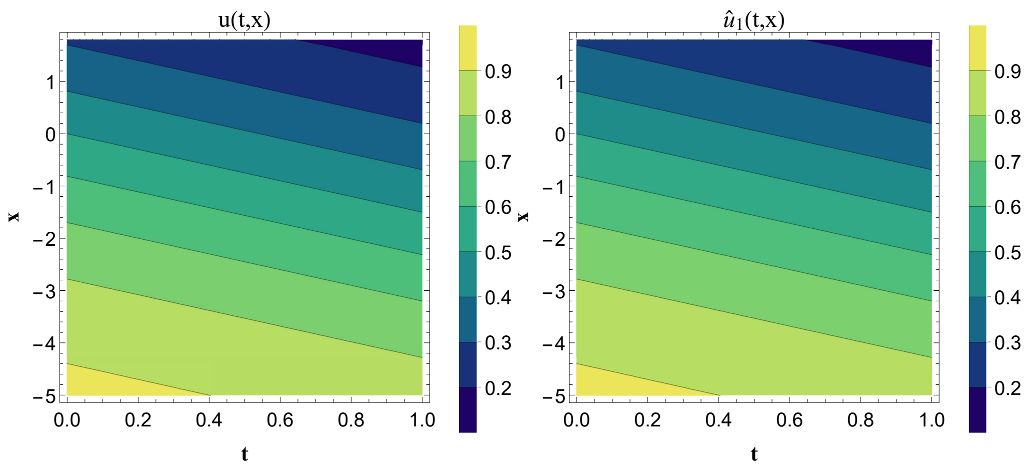


Figure 8. Contour plot of the Burgers–Huxley equation with respect to the solution $\hat{u}_1(t,x)$ and the exact solution $u(t,x)$.

To verify the validity and generality of our proposed equation, the method was applied to two classical equations, the Burgers–Fisher, and the Huxley equations. For this purpose, we performed a thorough analysis and obtained strong results that proved the validity of our method. Specifically, when $\alpha = -1$, $\beta = 1$, $\lambda = -1$, $\eta = 0$, $\delta = 1$, the Burgers–Fisher equation is as follows:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + u\frac{\partial u}{\partial x} + u(1-u), \quad u(0,x) = \frac{1}{2}\left(1 + \tanh\frac{x}{4}\right) \tag{14}$$

The exact solution of (14) is $u(t,x) = \frac{1}{2}\left(1 + \tanh\left(\frac{x}{4} + \frac{5t}{8}\right)\right)$. Similarly, using the traveling wave transform $\xi = x - ct$, problem (14) is transformed into an ODE, $u'' + cu' + uu' +$

$u(1-u)=0$, with initial value $u(0)=\frac{1}{2}$, $u'(0)=\frac{1}{8}$. Transformation of ODEs into the form of a system of differential equations,

$$u_1' = u_2,\ u_2' = \frac{5}{2}u_2 - u_1u_2 - u_1(1-u_1) \tag{15}$$

where $u_1(\xi) = u$, $c = -\frac{5}{2}$, and initial values $u_1(0) = \frac{1}{2}$, $u_2(0) = \frac{1}{8}$. The operator $D = u_2\partial_{u_1} + (\frac{5}{2}u_2 - u_1u_2 - u_1(1-u_1))\partial_{u_2}$ of (15), D_1 is chosen as $u_2\partial_{u_1} + \frac{5}{2}u_2\partial_{u_2} - u_1\partial_{u_2}$, the predicted solution $\hat{u}_1(\xi_\ell) = -\frac{1}{12}e^{\xi_\ell/2}(-7+e^{3\xi_\ell/2}) + \xi_\ell N_1$, $\hat{u}_2(\xi_\ell) = \frac{1}{24}(7e^{\xi_\ell/2}) - \frac{1}{6}e^{2\xi_\ell} + \xi_\ell N_2$, where the structure of the neural network is a single hidden layer containing 30 neurons with inputs $\xi_\ell \in [-5,2]$ of equidistant intervals of 100 training points and test points are 120 points of the interval $[-5,2.2]$, and the training results are shown in Figure 9. As shown in Figure 10, our method achieves an impressive performance with the loss function $\mathbb{L}(\theta)$ reaches 8.861×10^{-8} in about 700 iterations. This exceptional result again illustrates that the solution of the D_1 part of our proposed method captures the nonlinear nature of the solution, thereby reducing the computational cost associated with additional parameters which are evident from Figure 11. In addition, we provide a three-dimensional representation of the dynamics of the predicted solution $\hat{u}(t,x)$ with the exact solution $u(t,x)$ in the interval $t \in [0,1]$ and $x \in [-5,2]$, as shown in Figure 12.

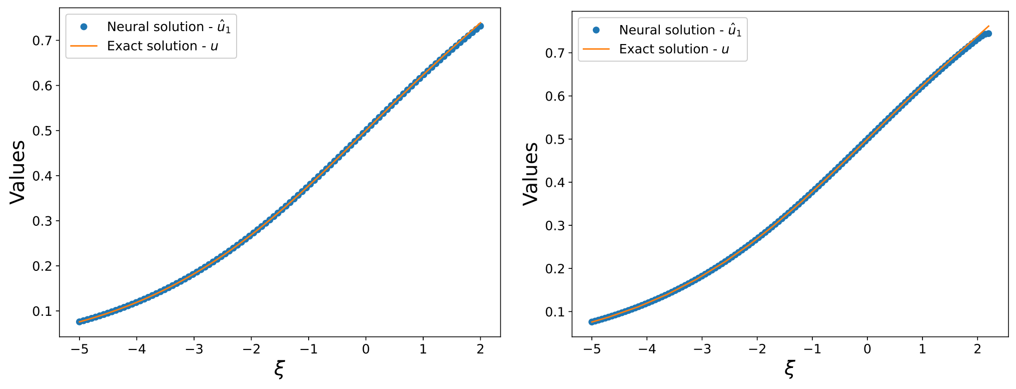


Figure 9. (**Left**) Comparison of solution $\hat{u}_1(\xi)$ with the exact solution $u(\xi) = \frac{1}{2}\left(1+\tanh\left(\frac{\xi}{4}\right)\right)$ of (15) in the training set. (**Right**) Comparison of $\hat{u}_1(\xi)$ with the exact solution $u(\xi) = \frac{1}{2}\left(1+\tanh\left(\frac{\xi}{4}\right)\right)$ of (15) in the test set.

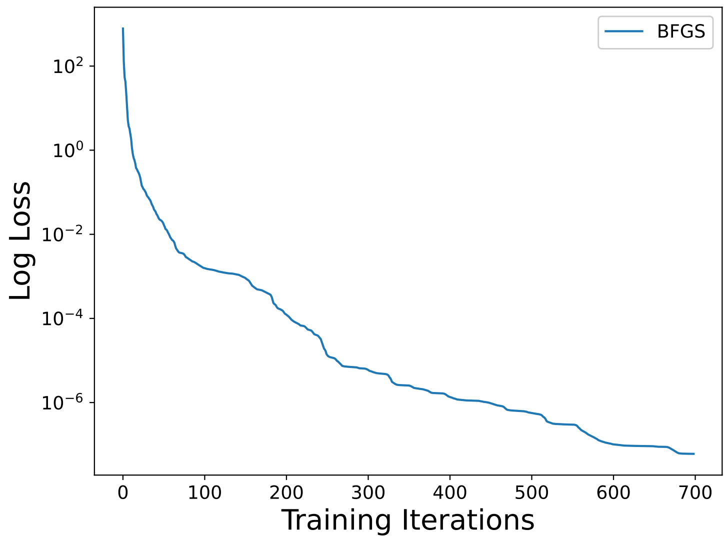


Figure 10. Curves of loss function versus number of iterations for Burgers–Fisher equation.

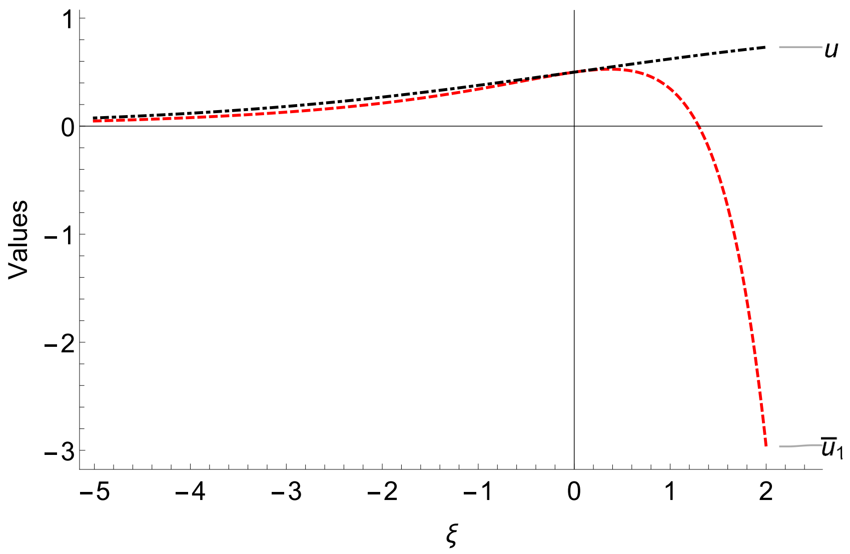


Figure 11. Comparison of the $\bar{u}_1(\xi)$ solution of the Burgers–Fisher equation with the exact solution $u(\xi)$.

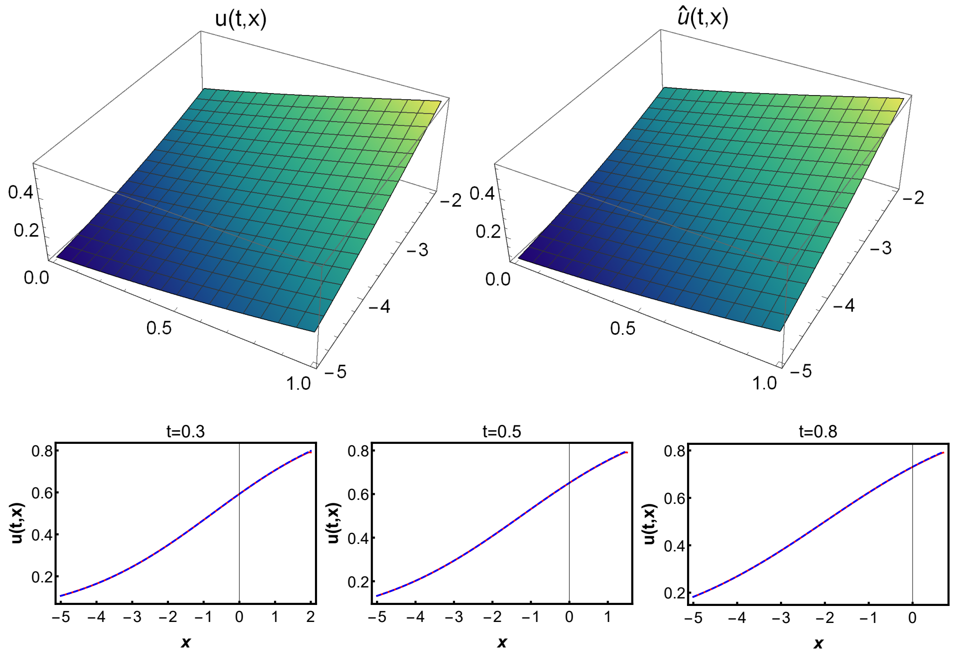


Figure 12. (**Top**) The true solution $u(t,x) = \frac{1}{2}\left(1 + \tanh\left(\frac{x}{4} + \frac{5t}{8}\right)\right)$ of the Burgers–Fisher equation is on the left, the predicted solution $\hat{u}(t,x)$ is on the right. (**Bottom**) Comparison of predicted and exact solutions at time $t = 0.3$, 0.5, and 0.8. (The dashed blue line indicates the exact solution $u(t,x)$, and the solid red line indicates the predicted solution $\hat{u}(t,x)$).

We investigate the Huxley equation under the conditions where $\alpha = 0$, $\beta = 1$, $\lambda = 1$, $\eta = 1$, $\delta = 1$. The equations are as follows:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + u(1-u)(u-1), \quad u(0,x) = \frac{1}{2}\left(1 + \tanh\frac{x}{2\sqrt{2}}\right) \tag{16}$$

The exact solution of (16) is $u(t,x) = \frac{1}{2}\left(1 + \tanh\left(\frac{\sqrt{2}x}{4} - \frac{t}{4}\right)\right)$. Similarly, using the traveling wave transform $\xi = x - ct$, problem (16) is transformed into an ODE, $u'' + cu' + u(1-u)(u-1) = 0$. It is transformed into the following differential equation form

$$u_1' = u_2,\ u_2' = -\frac{\sqrt{2}}{2}u_2 - u_1(1-u_1)(u_1-1) \tag{17}$$

where initial values $u_1(0) = \frac{1}{2}$, $u_2(0) = \frac{1}{4\sqrt{2}}$, and $c = \frac{\sqrt{2}}{2}$, it is clear that $u_1(\xi) = u(\xi)$, $u_2(\xi) = u'(\xi)$. In the case of $D_1 = u_2\partial_{u_1} - \frac{\sqrt{2}}{2}u_2\partial_{u_2}$, the system of differential equations $\bar{u}_1' = \bar{u}_2$, $\bar{u}_2' = -\frac{\sqrt{2}}{2}\bar{u}_2$, the initial values are $\bar{u}_1(0) = \frac{1}{2}$ and $\bar{u}_2(0) = \frac{1}{4\sqrt{2}}$, this time $\bar{u}_1(\xi) = \frac{3}{4} - \frac{1}{4}e^{-\xi/\sqrt{2}}$, $\bar{u}_2(\xi) = \frac{1}{4\sqrt{2}}e^{-\xi/\sqrt{2}}$.

For predicting the solution $\hat{u}_1(\xi)$ and $\hat{u}_2(\xi)$, the same neural network with two single hidden layers containing 30 neurons with the same structure is trained by optimization technique Broyden–Fletcher–Goldfarb–Shanno (BFGS) minimizes the Loss function $\mathbb{L}(\theta)$. The input ξ_ℓ is the interval $[-2,7]$ equidistantly spaced by 100 points. The test set is the 150 points in the interval $[-2,7.5]$. As shown in Figure 13, our proposed method produced excellent predictions for both the trained predicted and exact solutions. The variation of the loss function throughout the process is depicted in Figure 14, and it can be observed that the loss function decreased remarkably during training. Figure 15 shows the dynamics of $\hat{u}_1(t,x)$ with the exact solution $u(t,x)$, when $\xi = x - ct$ is substituted into $\hat{u}_1(\xi)$ and the predicted solution $\hat{u}_1(t,x)$ compared with the exact solution $u(t,x)$ at $t = 0.3, 0.5, 0.8$. The contour plot in Figure 16 provides a more visualization of the network solution $\hat{u}_1(t,x)$ compared to the exact solution $u(t,x)$.

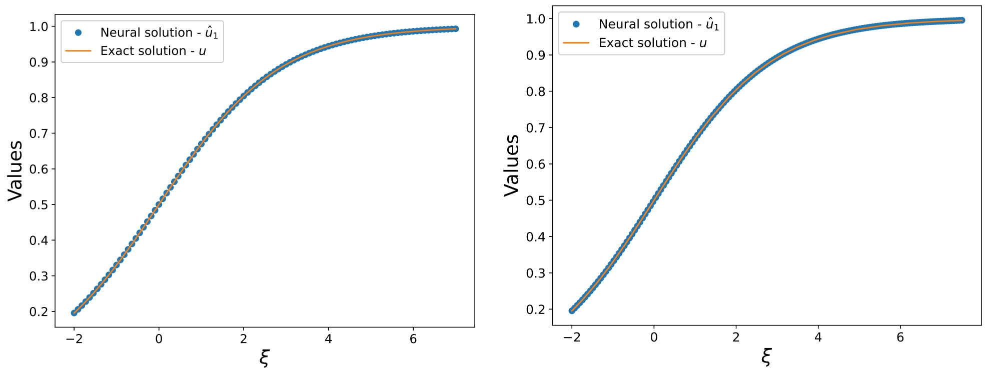


Figure 13. (**Left**) Comparison of solution $\hat{u}_1(\xi)$ with the exact solution $u(\xi) = \frac{1}{2}\left(1 + \tanh\left(\frac{\sqrt{2}\xi}{4}\right)\right)$ of (17) in the training set. (**Right**) Comparison of solution $\hat{u}_1(\xi)$ with the exact solution $u(\xi) = \frac{1}{2}\left(1 + \tanh\left(\frac{\sqrt{2}\xi}{4}\right)\right)$ of (17) in the test set.

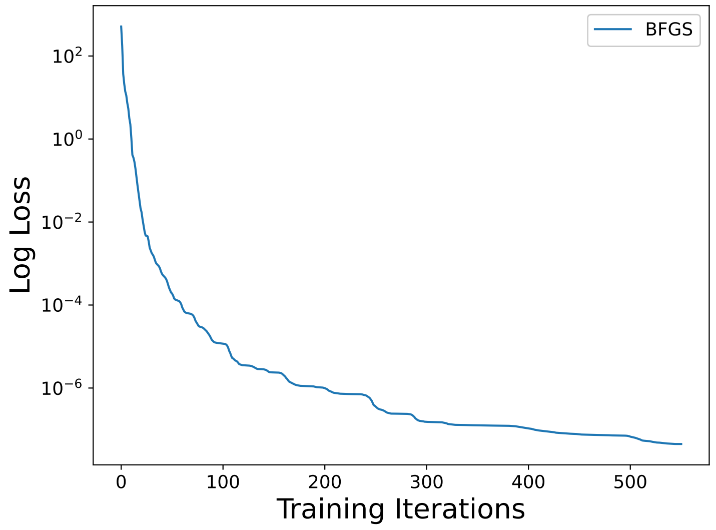


Figure 14. Curves of loss function versus number of iterations for Huxley equation.

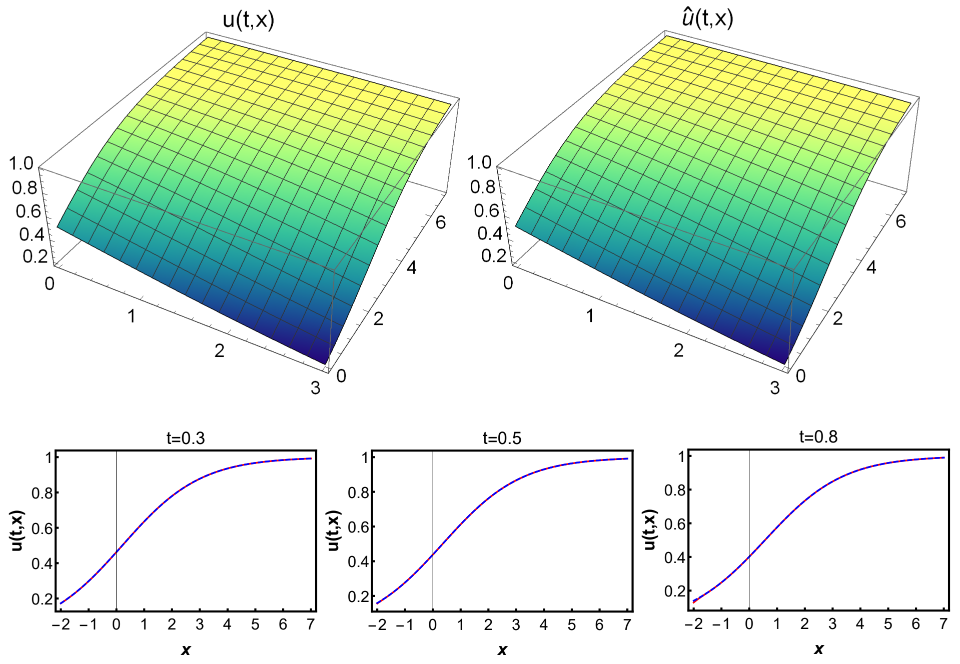


Figure 15. (**Top**) The true solution $u(t,x) = \frac{1}{2}\left(1 + \tanh\left(\frac{\sqrt{2}x}{4} - \frac{t}{4}\right)\right)$ of the Huxley equation is on the left, the predicted solution $\hat{u}(t,x)$ is on the right. (**Bottom**) Comparison of predicted and exact solutions at time $t = 0.3$, 0.5, and 0.8. (The dashed blue line indicates the exact solution $u(t,x)$, and the solid red line indicates the predicted solution $\hat{u}(t,x)$).

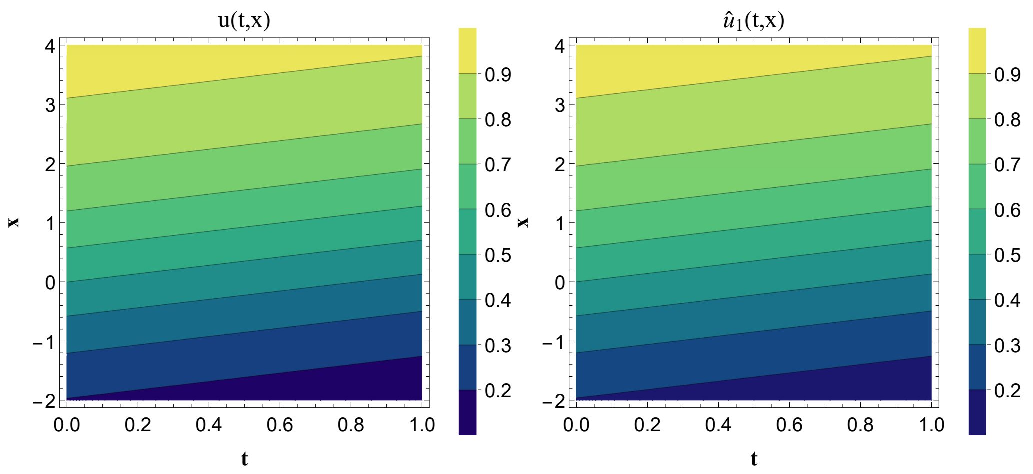


Figure 16. Contour plot of the Huxley equation with respect to the solution $\hat{u}_1(t,x)$ and the exact solution $u(t,x)$.

4. Discussion and Conclusions

The exponential growth of information data has resulted in limited data becoming a significant issue in various fields, especially in data-driven applications. Addressing this challenge has become a critical area of research in recent times. To contribute towards finding solutions to this problem, this paper proposes a novel method for resolving the Burgers-Huxley equation using a neural network based on Lie series in Lie groups of differential equations, which is an emerging field with great potential in solving complex problems. To the best of our knowledge, this study represents the first time the Burgers-Huxley equation has been solved using a Lie-series-based neural network algorithm. In physics, engineering, and biology, the Burgers–Huxley equation is a well-known mathematical model that is frequently utilized. Our novel approach offers a unique perspective on solving this equation by adding boundary or initial value items to the loss function, which leads to more accurate predictions and a better understanding of the underlying system. This research opens up new avenues for further exploration of the Lie-series-based neural network algorithm, specifically regarding its applications to other complex models beyond the Burgers–Huxley equation.

In this study, we present a novel method for obtaining a differentiable closed analytical form to provide an effective foundation for further research. The proposed approach is straightforward to use and evaluate. To verify the effectiveness of the suggested method, we applied it to two classic models of the Burgers–Fisher and Huxley equations that have well-known exact solutions. The proposed algorithm exhibits remarkable potential in capturing the nonlinear nature of equations and accelerating the computation process of neural networks. The performance of our method is demonstrated in Figures 3 and 11, which show how the proposed algorithm can capture the nonlinear behavior of the equations more effectively and speed up the computation of subsequent neural networks. To further evaluate the effectiveness of the proposed technique, we plotted the relationship between the loss function and the number of iterations in Figures 6, 10 and 14. Our results indicate that under the influence of the Lie series in Lie groups of differential equations, our algorithm can converge quickly and achieve more precise solutions with fewer data. Moreover, the accuracy of the obtained solutions is significant, and the generalization ability of the neural network is demonstrated by its ability to maintain high accuracy even outside the training domain, as shown in Figures 5, 9 and 13. We compared the performance of each neural network using small parameters (60 weight parameters and 31 bias parameters) with the exact solution to the problem. Our results highlight that the addition of the Lie series in Lie groups of differential equations algorithm remarkably enhances the ability of the neural network to solve a given equation.

Undoubtedly, the proposed method has several limitations that need to be carefully considered. Firstly, the method requires the transformation of PDEs into ODEs before applying the suggested algorithm. Although the results obtained after this transformation are preliminary, they provide useful insights for researchers. Additionally, an inverse transformation must be employed to produce the final solution $\hat{u}(t,x)$, taking into account the range of values for various variables. The choice of the operator D_1 may also influence the outcomes. Secondly, the current study only addresses nonlinear diffusion issues of the type $F(u) = \alpha u^\delta u_x + u_{xx} + \beta u\left(1 - u^\delta\right)\left(\eta u^\delta - \lambda\right)$, and the suitability of the technique was assessed via the computation of the loss function. Therefore, the applicability of the method to other types of non-linear PDEs is yet to be investigated, and it might require further adjustments to accommodate such problems. Despite some inherent challenges, our work offers a promising strategy for solving complex mathematical models using neural network algorithms based on Lie series. The computational performance of the proposed algorithm is noteworthy, achieving high solution accuracy at a relatively low time and parameter cost. In light of these findings, it is worth considering the prospect of applying this algorithm to financial modeling, where accurate predictions can have a significant impact.

Moving forward, there is ample scope for extending and improving the proposed algorithm further. Future research could explore how to optimize the performance of the algorithm by addressing its limitations and weaknesses for nonlinear PDE problems. For example, choosing a different neural network framework, CNN or recurrent neural network, etc., may improve the efficiency and accuracy of the method. Additionally, expanding the method's applicability beyond nonlinear diffusion issues may also yield valuable insights into other areas of mathematical modeling.

In summary, we believe that our work presents an exciting avenue for future research. By building upon our findings and addressing the limitations of the proposed algorithm, we can develop more sophisticated techniques for solving complex mathematical models in finance and other areas. Solving the above problems is the main goal of our next research work.

Author Contributions: Conceptualization, Y.W. and T.C.; methodology, Y.W.; software, Y.W.; validation, Y.W.; writing—original draft preparation, Y.W.; writing—review and editing, Y.W. and T.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China grant number 11571008.

Data Availability Statement: The data used to support the findings of this study are included within the article. The link to the code is https://github.com/yingWWen/Study-of-Burger-Huxley-Equation-using-neural-network-method (accessed on 14 March 2023).

Acknowledgments: The authors thank the support of the National Natural Science Foundation of China [grant number 11571008].

Conflicts of Interest: As far as we know, there are no conflict of interest or financial or other conflicts. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Ockendon, J.R.; Howison, S.; Lacey, A.; Movchan, A. *Applied Partial Differential Equations*; Oxford University Press on Demand: Oxford, UK, 2003.
2. Mattheij, R.M.; Rienstra, S.W.; Boonkkamp, J.T.T. *Partial Differential Equations: Modeling, Analysis, Computation*; SIAM: Philadelphia, PA, USA, 2005.
3. Duffy, D.J. *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
4. Jordan, M.I.; Mitchell, T.M. Machine learning: Trends, perspectives, and prospects. *Science* **2015**, *349*, 255–260. [CrossRef]
5. Mahesh, B. Machine learning algorithms—A review. *Int. J. Sci. Res. (IJSR)* **2020**, *9*, 381–386.
6. Yegnanarayana, B. *Artificial Neural Networks*; PHI Learning Pvt. Ltd.: New Delhi, India, 2009.
7. Zou, J.; Han, Y.; So, S.S. Overview of artificial neural networks. In *Artificial Neural Networks: Methods and Applications*; Humana Press: Totowa, NJ, USA, 2009; pp. 14–22.
8. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **1991**, *4*, 251–257. [CrossRef]
9. Lagaris, I.E.; Likas, A.; Fotiadis, D.I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **1998**, *9*, 987–1000. [CrossRef] [PubMed]
10. Chakraverty, S.; Mall, S. *Artificial Neural Networks for Engineers and Scientists: Solving Ordinary Differential Equations*; CRC Press: Boca Raton, FL, USA, 2017.
11. Yang, L.; Meng, X.; Karniadakis, G.E. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *J. Comput. Phys.* **2021**, *425*, 109913. [CrossRef]
12. Blechschmidt, J.; Ernst, O.G. Three ways to solve partial differential equations with neural networks–review. *GAMM-Mitteilungen* **2021**, *44*, e202100006. [CrossRef]
13. Han, J.; Jentzen, A. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat.* **2017**, *5*, 349–380.
14. Nabian, M.A.; Meidani, H. A deep learning solution approach for high-dimensional random differential equations. *Probabilistic Eng. Mech.* **2019**, *57*, 14–25. [CrossRef]
15. Chen, R.T.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D.K. Neural ordinary differential equations. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 6572–6583.
16. Sirignano, J.; Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **2018**, *375*, 1339–1364. [CrossRef]
17. Gorikhovskii, V.; Evdokimova, T.; Poletansky, V. Neural networks in solving differential equations. *J. Phys. Conf. Ser.* **2022**, *2308*, 012008. [CrossRef]
18. Huang, Z.; Liang, M.; Lin, L. On Robust Numerical Solver for ODE via Self-Attention Mechanism. *arXiv* **2023**, arXiv:2302.10184.
19. Lu, L.; Meng, X.; Mao, Z.; Karniadakis, G.E. DeepXDE: A deep learning library for solving differential equations. *SIAM Rev.* **2021**, *63*, 208–228. [CrossRef]
20. Berg, J.; Nyström, K. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing* **2018**, *317*, 28–41. [CrossRef]
21. Ruthotto, L.; Haber, E. Deep neural networks motivated by partial differential equations. *J. Math. Imaging Vis.* **2020**, *62*, 352–364. [CrossRef]
22. Quan, H.D.; Huynh, H.T. Solving partial differential equation based on extreme learning machine. *Math. Comput. Simul.* **2023**, *205*, 697–708. [CrossRef]
23. Tang, K.; Wan, X.; Yang, C. DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations. *J. Comput. Phys.* **2023**, *476*, 111868. [CrossRef]

24. Slavova, A.; Zecc, P. Travelling wave solution of polynomial cellular neural network model for burgers-huxley equation. *C. R. l'Acad. Bulg. Sci.* **2012**, *65*, 1335–1342.
25. Panghal, S.; Kumar, M. Approximate analytic solution of Burger Huxley equation using feed-forward artificial neural network. *Neural Process Lett.* **2021**, *53*, 2147–2163. [CrossRef]
26. Kumar, H.; Yadav, N.; Nagar, A.K. Numerical solution of Generalized Burger–Huxley & Huxley's equation using Deep Galerkin neural network method. *Eng. Appl. Artif. Intell.* **2022**, *115*, 105289.
27. Olver, P.J. *Applications of Lie Groups to Differential Equations*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1993; Volume 107.
28. Gröbner, W.; Knapp, H. *Contributions to the Method of Lie Series*; Bibliographisches Institut Mannheim: Mannheim, Germany, 1967; Volume 802.
29. Wen, Y.; Chaolu, T.; Wang, X. Solving the initial value problem of ordinary differential equations by Lie group based neural network method. *PLoS ONE* **2022**, *17*, e0265992. [CrossRef] [PubMed]
30. Wang, X.; Zhu, Z.; Lu, Y. Solitary wave solutions of the generalised Burgers-Huxley equation. *J. Phys. Math. Gen.* **1990**, *23*, 271. [CrossRef]