scientific reports

Check for updates

OPEN

An improved water strider algorithm for solving the inverse Burgers Huxley equation

Hassan Dana Mazraeh[1], Kourosh Parand[1,2]✉, Mehdi Hosseinzadeh[3,4], Jan Lansky[5] & Vladimír Nulíček[5]

In this paper, we introduce an improved water strider algorithm designed to solve the inverse form of the Burgers-Huxley equation, a nonlinear partial differential equation. Additionally, we propose a physics-informed neural network to address the same inverse problem. To demonstrate the effectiveness of the new algorithm and conduct a comparative analysis, we compare the results obtained using the improved water strider algorithm against those derived from the original water strider algorithm, a genetic algorithm, and a physics-informed neural network with three hidden layers. Solving the inverse form of nonlinear partial differential equations is crucial in many scientific and engineering applications, as it allows us to infer unknown parameters or initial conditions from observed data. This process is often challenging due to the complexity and nonlinearity of the equations involved. Meta-heuristic algorithms and neural networks have proven to be effective tools in addressing these challenges. The numerical results affirm the efficiency of our proposed method in solving the inverse form of the Burgers-Huxley equation. The best results were obtained using the improved water strider algorithm and the physics-informed neural network with 10,000 iterations. With this iteration count, the mean absolute error of these algorithms is $O(10^{-4})$. Additionally, the improved water strider algorithm is nearly four times faster than the physics-informed neural network. All algorithms were executed on a computing system equipped with an Intel(R) Core(TM) i7-7500U processor and 12.00 GB of RAM, and were implemented in MATLAB.

Keywords Burgers-Huxley equation, Water strider algorithm, Genetic algorithm, Physics-informed neural networks, Artificial intelligence

Partial differential equations (PDEs) are equations that involve partial derivatives of an unknown function with respect to several independent variables. The formulation of PDEs with the unknown function $u(x_1, x_2, x_3, \ldots, x_n)$ is as follows:

$$h\left(x_1, x_2, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_n}, \ldots\right) = 0, \tag{1}$$

PDEs are widely employed to model various phenomena in physics, engineering, biology, and other fields. PDEs can be classified into two types: linear and nonlinear. In a linear PDE, the unknown function and its partial derivatives appear linearly, meaning they are raised to the first power and are not multiplied or divided by each other[1]. In other words, the function $h(\cdot)$ represents a linear function of $u(\cdot)$ and its partial derivatives. In contrast, in a nonlinear PDE, the unknown function and its partial derivatives can appear in a nonlinear fashion, meaning they may be raised to powers other than one or be multiplied or divided by each other. In other words, $h(\cdot)$ represents a nonlinear function of $u(\cdot)$ and its partial derivatives. Some examples of nonlinear PDEs include the Burgers equation, the Burgers-Huxley equation, the Korteweg-de Vries equation, and the Navier-Stokes equation.

[1]Department of Computer and Data Sciences, Faculty of Mathematical Sciences, Shahid Beheshti University, G.C. Tehran, Iran. [2]Department of Cognitive Modeling, Institute for Cognitive and Brain Sciences, Shahid Beheshti University, G.C. Tehran, Iran. [3]School of Computer Science, Duy Tan University, Da Nang, Vietnam. [4]Jadara University Research Center, Jadara University, Irbid, Jordan. [5]Department of Computer Science and Mathematics, University of Finance and Administration, Prague, Czech Republic. ✉email: k_parand@sbu.ac.ir

The primary goal of solving linear or nonlinear PDEs is to find the unknown function $u(\cdot)$ in the equation, which includes all parameters, coefficients, boundary, and initial conditions. However, in some applications, in addition to the unknown function $u(\cdot)$, there may be other missing components such as parameters, coefficients, boundary, or initial conditions that need to be determined. In such cases, we are dealing with an inverse form of a PDE. The main objective of solving an inverse PDE is to estimate the missing components using observed data[2]. In this paper, we address an inverse form of the Burgers-Huxley equation, which is a nonlinear partial differential equation. The inverse form of the Burgers-Huxley equation is analyzed in contexts where initial conditions are either unknown or cannot be directly measured, yet the system's temporal behavior can be tracked. This scenario frequently arises in disciplines such as physics, engineering, and applied mathematics. Solving inverse problems with unknown initial conditions is intricate, often necessitating novel mathematical and computational strategies. Addressing these problems is vital as they can reveal insights into system dynamics and parameters that are not directly observable[3].

Before discussing the inverse form of the Burgers-Huxley equation, we will examine the generalized form of the Burgers-Huxley equation. The generalized form of the Burgers-Huxley equation for $0 \leq x \leq 1$ and $t \geq 0$ is as follows:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - \alpha u^{\delta} \frac{\partial u}{\partial x} + \beta u(1-u^{\delta})(u^{\delta}-\gamma), \tag{2}$$

with the initial condition:

$$u(x,0) = f(x) = \left[\frac{\gamma}{2} + \frac{\gamma}{2}\tanh(A_1 x)\right]^{\frac{1}{\delta}}, \tag{3}$$

and the boundary conditions:

$$u(0,t) = g_1(t) = \left[\frac{\gamma}{2} + \frac{\gamma}{2}\tanh(-A_1 A_2 t)\right]^{\frac{1}{\delta}}, \tag{4}$$

$$u(1,t) = g_2(t) = \left[\frac{\gamma}{2} + \frac{\gamma}{2}\tanh(A_1(1-A_2)t)\right]^{\frac{1}{\delta}}, \tag{5}$$

where:

$$A_1 = \frac{-\alpha\delta + \delta\sqrt{\alpha^2 + 4\beta(1+\delta)}}{4(1+\delta)}\gamma, \tag{6}$$

$$A_2 = \frac{\gamma\alpha}{1+\delta} - \frac{(1+\delta-\gamma)(-\alpha + \sqrt{\alpha^2 + 4\beta(1+\delta)})}{2(1+\delta)} \tag{7}$$

here $\alpha, \beta > 0$ are real constants, δ is a positive integer, and $\gamma \in (0,1)$[4]. In this paper, we assume that the initial condition described in equation (3) is missing. Consequently, we are dealing with an inverse form of the Burgers-Huxley equation. The main objective of this research is to find the missing initial condition when we have the main equation indicated in Equation (2) and boundary conditions (4) and (5). In solving an inverse form of a nonlinear partial differential equation to compensate for the missing part of the equation, a sensor is used at the point $x = a_0$ to collect data of $u(a_0,t)$ for $0 \leq t \leq T_M$. This data is referred to as the observed data or the over-specified condition. In this paper, to simulate the observed data, we utilize the exact solution of the equation to generate the data at $x = a_0$ as $u(a_0,t_j) + R_j$, where $j = 1,\ldots,M$, and $a_0 = 0.5$. Since data rarely comes without noise in the real world, we introduce random values R_j to the simulated observed data to simulate the noise. Therefore, the over-specified condition (data coming from a sensor) is as follows:

$$U(a_0,t_j) + R_j = S(t_j), \quad t_j = k \times j, \quad j = 1,2,3,\ldots,M. \tag{8}$$

In real-world applications, a sensor is used at an interior point a_0 to measure and collect data about a system modeled by a partial differential equation. This data helps approximate the unknown function of the system under consideration. To clarify the role of observed data, let us consider a heat conduction problem, which is a simple linear PDE. For example, consider the following heat conduction problem where the initial condition $f(x)$ is unknown:

$$U_t(x,t) = U_{xx}(x,t), \quad 0 \leq x \leq 1, \quad t \geq 1, \tag{9}$$

with boundary and initial conditions:

$$\begin{aligned} u(0,t) &= p(t), \\ u(1,t) &= q(t), \\ u(0,x) &= f(x). \end{aligned}$$

where $p(t) = 0$, $q(t) = \sin(1)e^{-t}$, and the initial condition $f(x)$ is unknown. This system describes heat conduction in a solid bar of length 1 unit. Since the initial condition is unknown, we are dealing with an inverse problem. The main goal is to find $u(x, t)$, which represents the temperature at time t and point x, and to determine the unknown initial condition. To compensate for the unknown initial condition, a sensor is used to collect data at a point in the interval $[a, b]$. Figure 1 illustrates this scenario.

In real-world applications, there are always measurement errors due to inaccuracies in the sensors used. In this paper, we assume a sensor is placed at $x = a_0 = 0.5$. We simulate the data collected by the sensor using the function $S(t_j)$, represented as $S_j = U(a_0, t_j) + R_j$, where R_j is a small random number that serves as random noise due to sensor inaccuracies.

The Burgers-Huxley equation is of very importance in science and engineering, such as neurophysiology, mathematical biology, chemical engineering, and more[5]. For example, equation (2) models the interaction between reaction mechanisms, convection effects, and diffusion transports[6].

Linear PDEs can be solved by using various methods, such as the separation of variables, Fourier series, Laplace transform, and more. Some examples of linear PDEs are the heat equation, the wave equation, and the Laplace equation. Nonlinear PDEs including the Burgers-Huxley equation are more difficult to solve than linear PDEs, and often require numerical methods or approximation techniques. In recent decades, many methods have been introduced to solve the Burgers-Huxley equation such as finite difference methods[7], finite element methods[8], spectral methods[9], semi-analytical approaches[10], Elzaki transform[11], and deep neural network[12].

In light of the absence of a solution for the inverse form of the Burgers-Huxley equation in cases where the initial condition is unknown, we were motivated to tackle the challenge of solving the inverse form of this equation with a missing initial condition. To address the problem, we employed an enhanced version of the recently proposed meta-heuristic algorithm known as the water strider algorithm and subsequently compared the results with those obtained from the original water strider algorithm, a multi-layer neural network, and a genetic algorithm. The criterion for comparing these methods is the mean absolute error (MAE) between the approximate and exact initial condition within the range $[a, b]$. To calculate the MAE, we calculate the absolute errors at the points $a = x_0 < x_1 < \cdots < x_{N-1} < x_N = b$ between the approximated $\hat{f}(x)$ and the exact $f(x)$, then calculated their mean. The improved water strider algorithm, the original water strider algorithm, and a genetic algorithm employ an evolutionary approach to find the unknown initial condition. In contrast, the multi-layer neural network uses a physics-informed neural network approach to determine the unknown condition.

In fact, our rationale for selecting the water strider algorithm lies in its demonstrated robust performance and significantly reduced computational time when contrasted with other meta-heuristic algorithms[13]. Furthermore, we have designed and implemented a physics-informed neural network (PINN) to address the inverse problem of this paper. This choice is motivated by the fact that neural networks have garnered significant attention for solving partial differential equations[14]. Furthermore, meta-heuristic algorithms and neural networks are integral components of computational intelligence methods. Consequently, it is reasonable to compare the results obtained from these two approaches.

In recent years, extensive research in machine learning and computational intelligence has addressed various scientific problems and real-world applications. One study utilized a machine learning approach combined with an optimization technique to fine-tune the parameters of a neural network model, predicting accurate and reliable solutions for traffic flow jamming transitions[15]. M. Sulaiman et al. investigated the saturation of two immiscible fluids (oil and water) flowing through homogeneous porous media during secondary oil recovery. They solved the modeled partial differential equation using supervised machine learning algorithms, specifically feedforward back-propagated neural networks and the Levenberg-Marquardt optimization algorithm[16]. In another project, M. Sulaiman et al. conducted a numerical investigation of heat transfer and flow of micropolar fluid in porous Darcy structures with isothermal and isoflux wall boundary conditions on a stretching sheet[17]. Dana Mazraeh et al. introduced an innovative combination of genetic programming and neural networks to solve nonlinear differential equations in astrophysics[18]. Umar et al. presented numerical simulations of a dynamical HIV model, including the effects of prevention, using an advanced computational framework that combines Meyer neural networks with local and global search methods, specifically genetic algorithms and interior-point algorithms, to solve the HIPV nonlinear mathematical system[19]. Mukdasaithe et al. aimed to present numerical simulations of a novel fractional order Leptospirosis model using stochastic numerical supervised neural networks[20]. Baty solved Lane-Emden type equations in astrophysics using recent deep learning methods with physics-

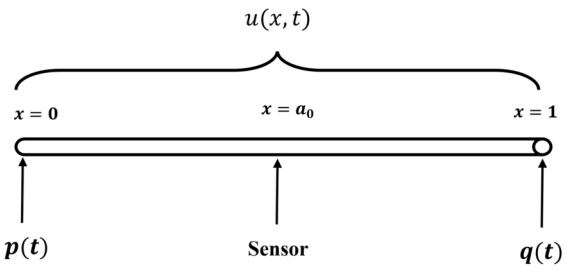


Fig. 1. A one dimensional heat conduction problem.

constrained neural networks[21]. Ahmad Khan et al. analyzed mathematical models for flow and heat transfer of a non-Newtonian fluid in axisymmetric channels with porous walls using continuity and momentum equations along with artificial intelligence-based feedforward neural networks[22]. Shahzad et al. investigated flow and heat transfer in a thin film of Cu-nanofluid over a stretching sheet, considering different shape factors along with slip and convective boundary conditions. The governing partial differential equations were transformed into nonlinear ordinary differential equations using similarity transformation and solved with MATLAB's BVP4C[23]. Dana Mazraeh et al. presented an improved imperialist competitive algorithm for solving an inverse form of the Huxley equation[24]. Khaled Alarfaj et al. explored deep neural networks with optimization algorithms to find approximate solutions for nonlinear fractional differential equations[25]. Sadaf et al. theoretically investigated the Chaffee-Infante equation to determine wave structure variations by finding exact closed-form solutions of the considered equations[26]. Ali et al. examined the perturbed Fokas-Lenells equation using the Bernoulli sub-equation function method and the $\frac{1}{G'}$ expansion method[27]. Mazraeh et al. proposed an improved cuckoo optimization algorithm to determine the unknown function $u(x)$ in Fredholm integral equations of the second kind[28]. Waqas et al. studied the numerical modeling of hybrid nanofluid with gold and silver nanoparticles across a stenotic artery using computational fluid dynamics[29]. Ali et al. investigated an extended (2+1)-dimensional perturbed nonlinear Schrödinger equation with Kerr law nonlinearity in a nano-optical fiber using fourth-order spatial derivatives[30]. Ali researched the Ivancevic option pricing model, an alternative to the traditional Black-Scholes model, formalized by adaptive nonlinear Schrödinger equations to characterize the option-pricing wave function in terms of stock price and time[31]. Molai et al. solved the mixed fuzzy relation programming with a nonlinear objective function using a modified imperialist competitive algorithm[32]. Zafar et al. explored new soliton solutions of the truncated M-fractional (1+1)-dimensional nonlinear Kaup-Boussinesq system using the exp_a function, modified simplest equation, and Sardar sub-equation techniques[33]. Khan et al. analyzed the thermal attributes of conductive, convective, and radiative moving fins with thermal conductivity and constant velocity, using the basic Darcy model and a soft computing paradigm based on feedforward artificial neural networks and meta-heuristic optimization techniques to evaluate the effects of significant parameters[34]. Mazraeh et al. combined a genetic algorithm with the Sinc-Galerkin method to solve an inverse diffusion problem[35]. Khan et al. investigated the model of swinging oscillation of a solid circular sector in various engineering applications using cascade learning[36].

In the following section, we present a discretization of the Burgers-Huxley equation using the finite difference method. This discretization is employed to solve the direct form of the Burgers-Huxley equation and evaluate the fitness value of a candidate solution accordingly.

Discritization of the Burgers-Huxley equation

In this study, we utilize the finite difference method to discretize Equation (2) for $\alpha = -1$ and $\delta = 1$. Consequently, we derive the following discretized representation for the Huxley equation:

$$\begin{aligned} -rU_{i-1,j+1} + (2r+1)U_{i,j+1} - rU_{i+1,j+1} &= rU_{i-1,j} - (2r + k\gamma)U_{i,j} \\ &+ (z + k + k\gamma)U_{i,j}^2 - kU_{i,j}^3 + rU_{i+1,j}(1 + zU_{i,j}), \\ i = 1, \cdots, N-1,\ j &= 0, \cdots, N-1 \end{aligned} \tag{10a}$$

$$U_{i,0} = f(ih),\ i = 1, ..., N-1, \tag{10b}$$

$$U_{0,j} = p(jk),\ j = 0, 1, ..., N-1, \tag{10c}$$

$$U_{N,j} = q(jk),\ Nh = 1,\ j = 0, 1, ..., N-1, \tag{10d}$$

where $x = ih$, $t = jk$, $r = k/h^2$, and $z = k/h$. In this study, the IWSA, WSA, and GA are used to approximate the unknown function $f(x)$ in Equation (2). Specifically, $f(x)$ is treated as a candidate solution represented as a real-valued vector (coefficients of a polynomial), which is then input into the fitness function for assessment. To evaluate the fitness of a candidate solution, System (10) is solved, and the numerical values $\hat{U}(x_i, t_j)$ are computed. Subsequently, the vector $\hat{s}(t_j) = \hat{U}(x = a_0, t_j)$ is compared to the vector $s(t_j)$ as described in Equation (8). To perform this comparison, the mean squared error is calculated. Smaller values of the mean squared error between $\hat{s}(t_j)$ and $s(t_j)$ indicate a better approximation of the unknown function $f(x)$. The pseudo-code of the fitness function in this study is as follows:

Data: Input values: Coefficients of a polynomial approximating $f(x)$
Result: Fitness value of the input values
Function `Fitness` (*Coefficients of a polynomial approximating* $f(x)$) **:**
    Calculate $\hat{U}(x_i, t_j)$ using System (10)
    return $\frac{1}{\sum_{j=1}^{m}(\hat{U}(a_0,t_j) - s_{t_j})^2}$;

Algorithm 1. Pseudo-code of the fitness function.

In Algorithm 1, as the approximation of $f(x)$ converges towards the exact $f(x)$, the denominator decreases. Consequently, the value of the fitness function increases.

Methods

In this section, first, we explain briefly the original water strider algorithm, then we introduce our improved water strider algorithm (IWSA) and demonstrate how our improvements enhance its accuracy and exploration rate. Furthermore, we illustrate how the IWSA is employed to solve the inverse form of the Burgers-Huxley equation. Considering that the genetic algorithm is a well-known and extensively studied method, we present a concise explanation of the real-valued genetic algorithm for the purpose of solving the inverse form of the Burgers-Huxley equation. Furthermore, we present a multi-layer physics-informed neural network designed to solve the inverse form of the Burgers-Huxley equation.

Original water strider algorithm

The Water Strider Algorithm (WSA) is a population-based optimization technique that was inspired by the life cycle of water strider insects. Water striders, small bugs known for their ability to traverse water surfaces using their extended legs and water-repellent hairs, exhibit a range of behaviors, including territoriality, communication, mating, feeding, and succession. These behaviors serve as a basis for simulating the search process of an optimization problem within the algorithm[13]. The main steps of the WSA are as follows:

1. Initialization: Starts by creating an initial population (water striders) of potential solutions or candidate solutions to the optimization problem. These solutions are generated randomly as follows:

$$WS_i^0 = Ub + R \times (Ub - Lb), i = 1, 2, \cdots, nws, \tag{11}$$

where WS_i^0 is the initial position of i^{th} water strider. *Ub* and *Lb* are the maximum and minimum values that are allowed, respectively. *R* is a random number between 0 and 1. The number of WSs is nws. The fitness of initial WSs is assessed by applying a fitness function to determine how suitable their positions are within the search space.

2. Territoriality: WSs establish territories for their living, mating, and feeding activities. To form a total of nt territories, the following method is employed to assign the WSs to these territories:

   - Sort the WSs in descending order based on their fitness.
   - Divide sorted WSs into $\frac{nws}{nt}$ group orderly.
   - The j^{th} member of each group is assigned to j^{th} territory, where $j = 1, 2, ..., nt$. In every region, the top-ranked and bottom-ranked positions in terms of fitness are identified as the female and male (keystone) roles, respectively.

3. Communication: By creating ripples of different amplitudes, durations, and frequencies on the water surface, water striders can communicate various information. They also use this system to sense potential predators and prey. They make waves by moving their legs on the water. Each wave has a specific meaning, such as courtship, warding off threats, distinguishing genders, and locating prey, and so on. They detect the communication waves through sensors on their legs and respond accordingly.
4. Mating: The males make signals to show they want to mate, and the females answer them with signals that mean they are interested or not. If the female's response is positive, they will mate. Otherwise, the male will try to force her to mate. The females who do not want to mate use a hardcover to block the male's hold and escape. The male may still manage to mate after some tries, but he usually gives up and leaves. This process uses a lot of energy and the male needs food. The keystone (male) may mate or not, either way, the new position of keystone will be calculated as follows:

$$\begin{cases} \mathrm{WS}_i^{t+1} = \mathrm{WS}_i^t + random\_number \times R, & \text{If breeding occurs (with a likelihood of p)} \\ \mathrm{WS}_i^{t+1} = \mathrm{WS}_i^t + R.(1 + \ \mathrm{random\_number}), & \text{otherwise} \end{cases} \tag{12}$$

here WS_i^t is the position of $i^{\ \mathrm{th}}$ WS in the $t^{\ \mathrm{th}}$ iteration, random_number is a random number in [0, 1], *R* represents a vector that starts at the location of a male (denoted as $\left(\mathrm{WS}_i^{t-1}\right)$) and ends at the location of a female within the identical territory ($\left(\mathrm{WS}_F^{t-1}\right)$).

5. Foraging: The mating process consumes considerable energy, regardless of its success. Hence, after completing mating behavior and relocating to a new position, WSs actively seek food sources. Typically, females inhabit areas abundant in food resources, leading males to go toward these locations. Although these areas offer easy access to food, males may encounter competition from other males who want to mate. In the algorithm, their assessment of the position's suitability for food availability involves using an objective function. Should the current position post-mating yield a higher objective function value than the previous one, they

have likely found food. Conversely, if the post-mating position yields a lower objective function value, it prompts them to seek the best habitat boasting the highest fitness. Employing the following equation, they navigate toward a new position around the lake's most optimal WS (WS_{BL}^{t}), known for its abundant food resources.

$$\mathrm{WS}_{i}^{t+1} = \mathrm{WS}_{i}^{t} + 2\mathrm{rand}.\left(\mathrm{WS}_{BL}^{t} - \mathrm{WS}_{i}^{t}\right) \tag{13}$$

6. Death or survive: Following the foraging stage, the objective function assesses the outcome of the food gathering process against the previous position. Should the new fitness be lower, the WS will perish due to its inability to secure food and heightened potential for confrontation with destination territory WSs. Subsequently, a newly matured larva assumes the role of the deceased WS as the keystone, positioned randomly within the territory utilizing the following equation. If the new fitness is higher, the keystone will survive.

$$\mathrm{WS}_{i}^{t+1} = Lb_{j}^{t} + 2\mathrm{rand}\left(Ub_{j}^{t} - Lb_{j}^{t}\right) \tag{14}$$

In this context, Ub_{j}^{t} and Lb_{j}^{t} represent the upper and lower limits of the WS's position within the j^{th} territory, indicating the territorial boundaries where the WS perished.

7. Termination of Algorithm: Steps 2 to 6 are repeated until the termination condition is satisfied.

An improved water strider algorithm

Since the original WSA has shown great performance, in recent years, many researchers have attempted to enhance this algorithm as a more powerful method for addressing specific problem domains. Following, we provide a review of the efforts made by others to improve the original WSA. However, the novelty of our presented improvement lies in our focus on the procedure of replacing new larvae with a dead keystone. Furthermore, this study represents the first attempt to solve an inverse form of nonlinear partial differential equations using the original WSA and, IWSA, and PINN.

In the article by Xu et al.[37], they proposed a modified version of the WSA by incorporating two enhancements: the Quasi Opposition-Based Learning (QOBL) technique and an elite-guide evolution mechanism. Duan et al.[38] introduced two improvements to the original WSA. Their enhancements involve integrating an opposition-based learning (OBL) mechanism, which considers both an individual's value and its opposite value to choose the better candidate for enhancing search diversity. Additionally, they included an adaptive parameter in the foraging process. Liao et al.[39] employed Levy Fight (LF) as a chaotic mechanism to enhance the exploration and convergence rate of the original algorithm. Bi et al.[40] modified the foraging update formula using a chaotic mechanism to augment the exploration capabilities of the original WSA. Hu et al.[41] utilized opposition-based learning to accelerate optimization speed and incorporated a chaos map mechanism to increase the exploration rate of the original algorithm. Kaveh et al.[42] have used opposition-based learning to improve the optimization phase of the algorithm and also used a mutation step to increase the exploration rate of the original algorithm. Liu et al.[43] leveraged the Quasi-opposition learning strategy during the learning phase and integrated an elite-guide evolution procedure to enhance the optimization and exploration rates of the algorithm. Lastly, Syah et al.[44] implemented a chaotic mechanism to amplify the exploration capabilities of the algorithm.

Now, we present our improved WSA as follows: In Step 6 of the original WSA, when a keystone dies, a new larva is generated randomly to replace the dead WS. However, in nature, newborns and larvae generally do not have equal opportunities to grow and enter a territory. Typically, weaker offspring perish due to competition for food or through selection mechanisms. Thus, this study accounts for this reality by modifying Step 6 of the original WSA. Initially, *K* larvae are generated randomly, and then one is selected through tournament selection. Rather than generating just one larva, *k* larvae are randomly created, thereby enhancing the exploration rate of the original WSA. Furthermore, the most superior among these larvae takes the place of the deceased WS, aligning more closely with natural occurrences and increasing the exploitation rate. In fact, in nature, an adult female insect typically lays several eggs in her territory. Among the hatched larvae, a number are destroyed based on competition or weakness, and the stronger ones replace the weaker ones. Similarly, in our improvement, generating multiple larvae randomly in the search space increases the exploration rate. The competitive selection mechanism (based on the tournament selection) ensures that the powerful individuals in the search space replace the weaker ones, thereby enhancing the exploitation rate.

To compare the performance of the original WSA and the improved version of WSA, we solved an Ackley function using both algorithms. The Ackley function is a well-known benchmark function used to evaluate optimization algorithms. It is highly multi-modal, meaning it has many local minima, which makes it challenging for optimization algorithms to find the global minimum. By using the Ackley function, we can effectively compare the performance of the original WSA and the improved WSA in terms of their ability to navigate complex landscapes and avoid local minima, demonstrating the improvements in exploration and exploitation capabilities. The Ackley function under consideration is as follows:

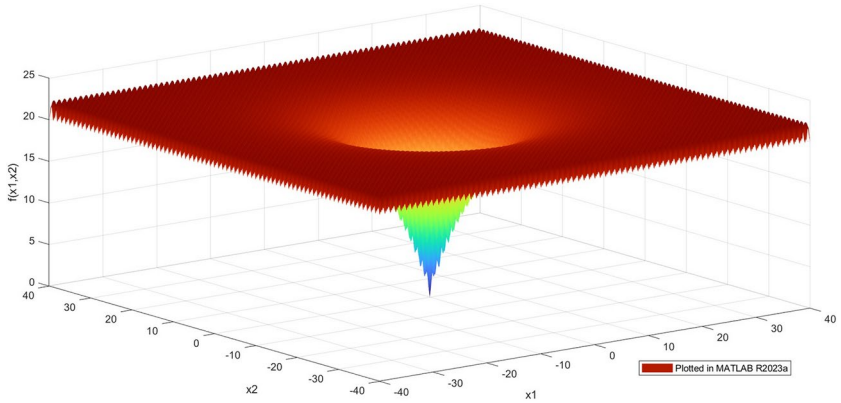


Fig. 2. Ackley function for two variables, x_1 and x_2, in the range $[-40, 40]$. This figure was plotted in MATLAB R2023a.

Iterations	$A(x)_{\mathrm{WSA}}$	$Time(S)_{\mathrm{WSA}}$	$A(x)_{\mathrm{IWSA}}$	$Time(S)_{\mathrm{IWSA}}$
100	1.7837	0.048	0.7430	0.061
200	1.1126	0.094	0.4198	0.103
300	1.5127	0.151	0.5474	0.221
400	1.3898	0.228	0.3105	0.243
400	1.1157	0.252	0.2615	0.264
600	0.9369	0.283	0.1477	0.312
700	0.9357	0.338	0.1106	0.420
800	1.0571	0.383	0.0956	0.431
900	0.9899	0.426	0.0963	0.475
1000	0.6134	0.448	0.0533	0.611

Table 1. Comparison between the original WSA and the IWSA for minimizing Function (15).

$$A(x) = -20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{\sqrt{\frac{1}{d}\sum_{i=1}^{d} \cos 2\pi x_i}} + 20 - e^1, \tag{15}$$

Figure 2 illustrates Function (15) for $x_i \in [-40, 40], \quad i = 1, 2$. For illustration purposes, we have plotted Function (15) for only two variables, x_1 and x_2. In fact, the length of vector x for comparison between the IWSA and the original WSA is 20. These algorithms aim to find a vector x that minimizes Function (15). The global minimum of Function (15) is at $x^* = [x_1, x_2, \ldots, x_n] = [0, 0, \ldots, 0]$. Furthermore, The global minimum of the this function is 0. Therefore, the algorithms aim to find a vector x such that the value of the Ackley function approaches 0. In this context, each vector x represents a water strider. Table 1 shows the results obtained from the original WSA and the IWSA for different iterations. The original WSA and the improved WSA aim to minimize the Ackley function, as shown in Equation (15). As illustrated in Fig. 2, finding the minimum point is challenging due to the function's numerous local minima. Table 1 presents the absolute values obtained by these algorithms for minimizing Equation (15) across different iteration numbers. It is evident that the improved WSA is more effective at minimizing the Ackley function compared to the original WSA. It's important to note that for each result reported in this paper, the algorithms were run five times, and the best result among those runs is reported. Since all algorithms used in this paper are stochastic, different runs might yield different results, especially with a low number of iterations. Therefore, to facilitate the reproduction of results, or at least obtain results similar to those reported in this paper, we have run the algorithms five times. We did this in the hope that the results obtained by other researchers will be close to those presented here. It is worth mentioning that, based on our experiments, the results tend to converge with a higher number of iterations.

Figure 3 illustrates the values of Function (15) extracted from Table 1 for different iterations. As it is evident from Table 1 and Fig. 3, the accuracy of the IWSA is better than the original WSA.

A real-valued genetic algorithm for solving the inverse form of the Burgers-Huxley equation

The genetic algorithm, primarily formulated by Holland[45], relies on principles of biological evolution pioneered by Darwin. This approach has proven effective in tackling a range of optimization challenges. It operates as a stochastic optimization technique, employing a collection of chromosomes, each embodying a potential solution. Through the application of genetic operations, these chromosomes progressively refine, serving as the foundation for subsequent generations. This iterative process persists until either the specified number

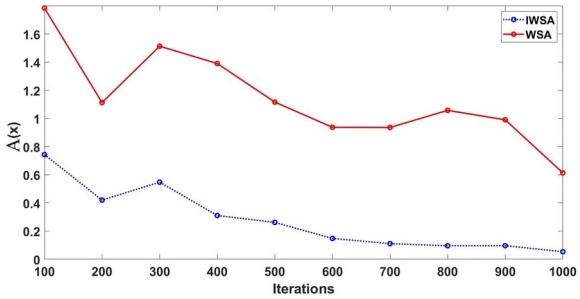


Fig. 3. The values of Fnction (15) extracted from Table 1 for different iterations.

Representation	Real valued vectores
Length of chromosomes	Degree of a polynomial
Recombination	One point crossover
Recombination probability	100%
Mutation	Adding a random value
Mutation probability	$1/n$
Parent selection	Roulette wheel
Survivor selection	Replace the worst
Number of offspring	1
Initialization	Random
Termination condition	Number of generation

Table 2. Parameters of the genetic algorithm.

of generations is attained or the predetermined fitness threshold is met. The sequence of steps in a genetic algorithm includes:

1. Create an initial set of chromosomes randomly.
2. Assess the fitness of every chromosome within the population.
3. Select some chromosomes as parents.
4. Apply recombination operation on parents.
5. Apply mutation operation on the offspring.
6. Evaluate the fitness of offspring.
7. Update the population.
8. Repeat Step 3 to Step 7, until predefined number of iterations is not satisfied.

Table 2 presents the parameters of a real-valued GA used in this paper.

A multi-layer physics-informed network for solving the inverse form of Burgers-Huxley equation

Physics-informed neural networks[14] are new methods that use physics knowledge to design and train neural networks. These networks combine the power of neural networks with the basics of physics, allowing them to learn and use physical laws, constraints, or equations in their structure. By adding this prior knowledge, physics-informed neural networks improve their learning and generalization from limited data while making sure the solutions follow the physical principles of the system. This integration not only makes the predictions more precise but also explains how the learned features and the physics are related, offering a potential way to solve hard scientific problems and physical modeling tasks. In this study, a physics-informed neural network, with the architecture illustrated in Fig. 4, is utilized to solve Equation 2 for $\alpha = -1$, $\beta = 1$, and $\delta = 1$:

In this work, the first hidden layer consists of 10 nodes, the second hidden layer comprises 20 nodes, and the third hidden layer comprises 10 nodes. All activation functions in the hidden layers are 'ReLU', and the 'loss' function is as follows:

$$Loss = Loss_{PDE}(x,t) + Boundary_1(0,t) + Boundary_2(1,t) + Loss_{data}, \tag{16a}$$

$$Loss_{PDE}(x,t) = \left(\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} + \alpha u^{\delta}\frac{\partial u}{\partial x} - \beta u(1-u^{\delta})(u^{\delta}-\gamma)\right)^2, \tag{16b}$$

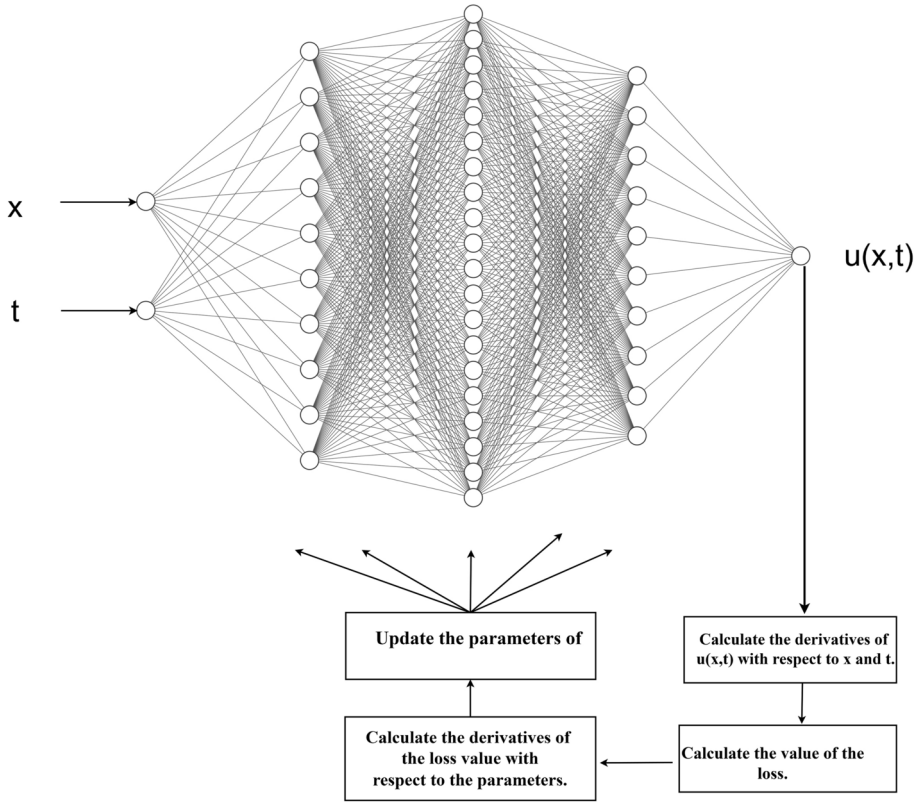


Fig. 4. The architecture of the physics-informed neural network used in this study.

$$Boundary_1(0,t) = \sum_{j=0}^{N} (U(0,t_j) - g_1(t_j))^2, \tag{16c}$$

$$Boundary_2(1,t) = \sum_{j=0}^{N} (U(1,t_j) - g_2(t_j))^2, \tag{16d}$$

$$Loss_{data} = \sum_{j=0}^{N} (U(a_0,t_j) - s(t_j))^2 \tag{16e}$$

Results

In this study, an inverse form of the Burgers-Huxley equations for $\alpha = -1$, $\beta = 1$, $\delta = 1$, $0 \leq x \leq 1$, and $0 \leq t \leq 1$ is considered as follows[46]:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + u\frac{\partial u}{\partial x} + u(1-u)(u-\gamma), \tag{17}$$

with the boundary conditions:

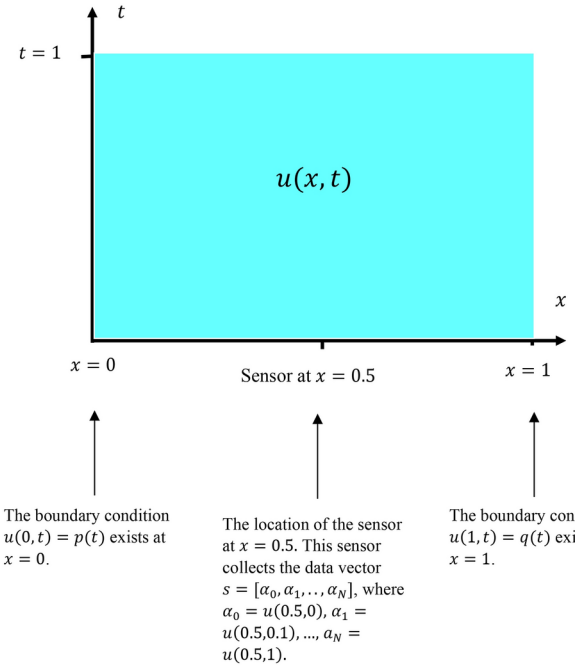
$$u(0,t) = \frac{1}{2} - \frac{1}{2}\tanh\frac{3}{8}t, \tag{18}$$

$$u(1,t) = \frac{1}{2} - \frac{1}{2}\tanh\frac{1}{4}\left(1 + \frac{3}{2}t\right), \tag{19}$$

and the over-specified condition (data coming from sensors):

$$u(a_0,t) + R = s(t_j), \quad t_j = k \times j, \quad j = 1,2,3,\ldots,M. \tag{20}$$

where $\gamma = 2.2204^{-16}$ and $a_0 = 0.5$. The initial condition is considered unknown in this study. The exact initial condition is given by $u(x,0) = f(x) = \frac{1}{2} - \frac{1}{2}\tanh\frac{1}{4}x$, and it is the target for our algorithms to determine.



The total loss is calculated as follows:

Suppose we have an approximated $\hat{u}(x,t)$ and suppose we want to solve the problem described in Eq. (9) for $0 \le x \le 1$, $0 \le t \le 1$. The loss values and the total loss are calculated as follows:

$$loss_{pde} = \sum_{i=0}^{N}\sum_{j=0}^{M}(\hat{u}_t(x_i,t_j) - \hat{u}_{xx}(x_i,t_j))^2,$$

$$loss_{boundary_1} = \sum_{j=0}^{M}(\hat{u}(0,t_j) - p(t_j))^2,$$

$$loss_{boundary_2} = \sum_{j=0}^{M}\left(\hat{u}(1,t_j) - q(t_j)\right)^2,$$

$$loss_{over-specified} = \sum_{j=1}^{M}\left(\hat{u}(0.5,t_j) - s(t_j)\right)^2,$$

$$loss_{total} = loss_{pde} + loss_{boundary_1} + loss_{boundary_2} + loss_{over-specified}$$

Fig. 5. Incorporation of the collected data into the total loss value calculation.

Iterations	MAE_{GA}	MAE_{WSA}	MAE_{IWSA}	MAE_{PINN}
1000	0.00577	0.01704	0.01082	0.00883
2000	0.00201	0.01129	0.00778	0.00747
3000	0.00309	0.00693	0.00820	0.00324
4000	0.00411	0.00250	0.00662	0.00670
5000	0.00258	0.00961	0.00694	0.00226
6000	0.00413	0.00767	0.00957	0.00147
7000	0.00445	0.00609	0.00704	0.00232
8000	0.00339	0.00559	0.00354	0.00125
9000	0.00295	0.00218	0.00210	0.00053
10000	0.00289	0.00164	0.00088	0.00023

Table 3. Mean absolute error between the exact $f(x)$ and approximated $f(x)$ obtained from the original WSA, the IWSA, the genetic algorithm, and the physics-informed neural network explained in Section Methods.

For clarification purposes, we explain how the collected data from the sensor (over-specified condition) are incorporated into loss calculation and how the unknown initial condition $f(x)$ is determined by the algorithms. To keep this explanation straightforward, we consider a simple case of partial differential equations (the heat conduction equation) as shown in Equation (9). The same approach is applied to solve the inverse form of the Burgers-Huxley equation. During the iterations of the improved WSA, original WSA, GA, and the neural network described in this paper, the over-specified condition is taken into consideration. This means the algorithms aim to minimize the loss and the error between the collected data (over-specified condition) and the values of the approximated solution at the sensor location. Figure 5 illustrates how the collected data are incorporated into the loss value calculation when solving an inverse form of a partial differential equation. The improved WSA, original WSA, and GA try to find $\hat{f}(x)$ so that the total loss is minimized. The PINN tries to find the best $\hat{u}(x,t)$ that minimizes the total loss. After finding the best approximation $\hat{u}(x,t)$ by the PINN, we calculate $\hat{u}(x,0) = \hat{f}(x)$. To evaluate the accuracy of the algorithms, we then calculate the mean absolute error between the approximated $\hat{f}(x)$ and the exact $f(x)$.

Table 3 displays the mean absolute error between the exact $f(x)$ and the approximated $\hat{f}(x)$ obtained from the original WSA, as detailed in Section Methods, the IWSA presented in Section Methods, the genetic algorithm described in Section Methods, and the physics-informed neural network explained in Section Methods. In general, increasing the number of iterations of the algorithms improves their accuracy. We have reported the

Iterations	$\text{Time(S)}_{\text{GA}}$	$\text{Time(S)}_{\text{WSA}}$	$\text{Time(S)}_{\text{IWSA}}$	$\text{Time(S)}_{\text{PINN}}$
1000	0.94	3.37	4.01	20.94
2000	2.56	7.69	8.70	44.50
3000	3.04	10.57	11.72	65.53
4000	4.92	12.98	13.65	95.84
5000	6.12	16.67	16.45	112.49
6000	7.59	19.23	21.24	155.81
7000	9.22	24.99	30.73	163.41
8000	9.87	26.79	39.91	189.84
9000	11.08	29.54	48.73	205.08
10000	12.21	32.75	51.057	225.73

Table 4. Execution time (in seconds) obtained from the original WSA, the IWSA, the genetic algorithm, and the physics-informed neural network explained in Section Methods.

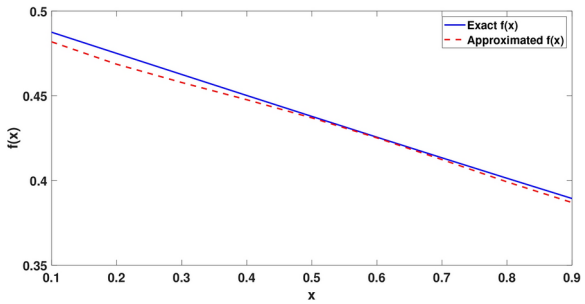


Fig. 6. The exact and approximated $f(x)$ found by the GA after 10000 iterations.

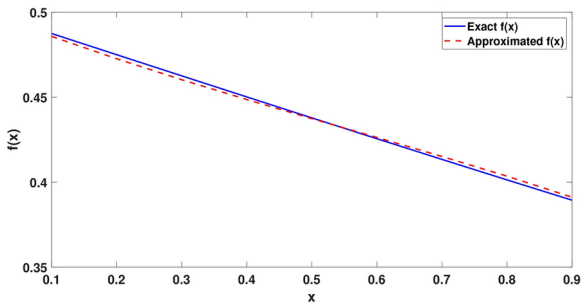


Fig. 7. The exact and approximated $f(x)$ found by the original WSA after 10000 iterations.

results for iterations ranging from 1000 to 10000 because, based on our experiments, the algorithms did not converge with fewer than 1000 iterations, and accuracy did not improve significantly beyond 10000 iterations.

According to the last entry of Table 3, when the iteration number is 10,000, the accuracy of the original WSA is better than that of the GA, and the accuracy of the improved WSA is better than the original WSA. However, the accuracy of the PINN is slightly better than the improved WSA. Conversely, the execution time of the improved WSA is almost four times shorter than that of the PINN. In fact, the improved WSA is nearly as accurate as the PINN but much faster.

Table 4 shows execution time (in seconds) obtained from the original WSA as detailed in Section Methods, the IWSA presented in Section Methods, the genetic algorithm described in Section Methods, and the physics-informed neural network explained in Section Methods.

The figures, from Figs. 6, 7, 8, 9, illustrate the exact $f(x)$ and the approximated $\hat{f}(x)$ found by the GA, WSA, IWSA, and PINN after 10000 iterations, respectively.

Figure 10 depicts the loss values of the PINN over 1000 to 10000 iterations.

Figure 11 shows the mean absolute error, extracted from Table 3, which represents the difference between the exact and approximated $f(x)$ values obtained by the GA, the original WSA, IWSA, and PINN across 1000 to 10000 iterations. As evident from Fig. 11, the accuracy of all algorithms improves as the iteration numbers

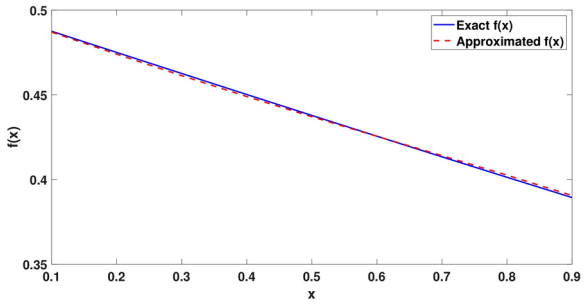


Fig. 8. The exact and approximated $f(x)$ found by the IWSA after 10000 iterations.

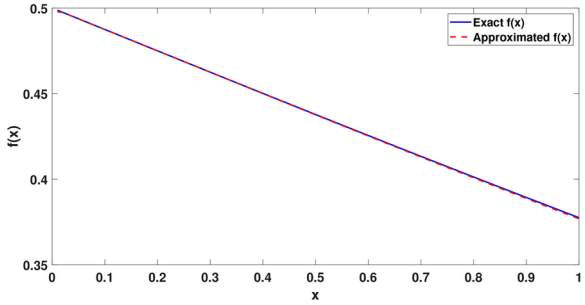


Fig. 9. The exact and approximated $f(x)$ found by the PINN after 10000 iterations.

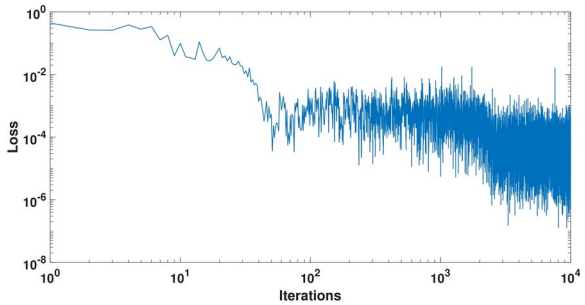


Fig. 10. Loss values of the PINN across 1000 to 10000 iterations.

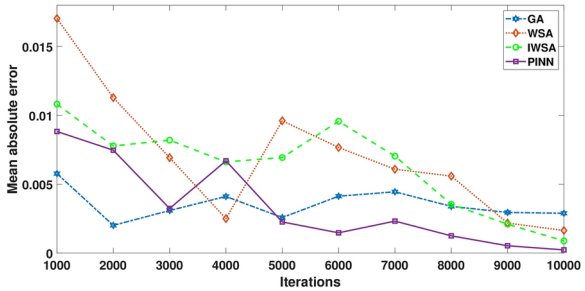


Fig. 11. The mean absolute error, extracted from Table 3, represents the difference between the exact and approximated $f(x)$ values obtained by the GA, the original WSA, IWSA, and PINN across 1000 to 10000 iterations.

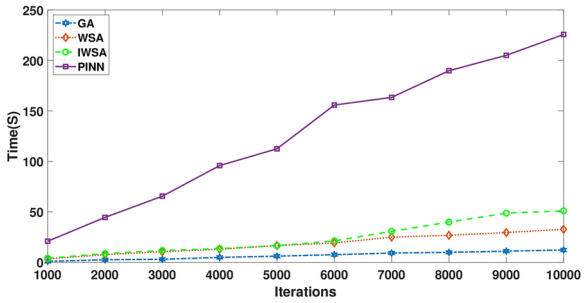


Fig. 12. The execution times (in seconds) of the GA, the original WSA, IWSA, and PINN, extracted from Table 3, across 1000 to 10000 iterations.

IWSA	Our PINN	PINN[47]	PANN[48]	ITLBO[49]	IICA[24]
$O(10^{-4})$	$O(10^{-4})$	$O(10^{-2})$	$O(10^{-4})$	$O(10^{-3})$	$O(10^{-4})$

Table 5. A comparison study (order of mean absolute error) of some recent research articles for solving an inverse form of nonlinear partial differential equations related to the subject of this research article.

increase from 1,000 to 10,000. However, the improved WSA and the PINN surpass the other algorithms in terms of accuracy and convergence rate.

Figure 12 depicts the execution times (in seconds) of the GA, the original WSA, IWSA, and PINN, extracted from Table 3, across 1000 to 10000 iterations.

Comparison study

In this section, we compare the results of this paper with some recent research on neural networks and metaheuristic algorithms for solving inverse forms of the Burgers' equation, Huxley equation, or related equations. In a study by S. Alkhadhr, the researchers introduced a method for solving the inverse problem of the Burgers' equation using a physics-informed neural network, considering some coefficients of the problem as unknown[47]. S. Pakravan's research proposes a novel composite framework to determine unknown parameters of the Burgers' equation using physics-aware neural networks (PANN). This method combines the high expressibility of deep neural networks as universal function estimators with the accuracy and reliability of existing numerical algorithms for partial differential equations as custom layers in semantic autoencoders[48]. Boroujeni et al. introduced an improved metaheuristic algorithm called the teaching-learning-based algorithm (ITLBO), which they combined with numerical methods to solve an inverse form of the Burgers-Fisher equation, with an unknown boundary condition that their algorithm aims to identify[49]. Mazraeh et al. utilized an enhanced metaheuristic algorithm known as the imperialist competitive algorithm (IICA) to solve an inverse form of the Huxley equation[24]. Table 5 presents a compact comparative study (order of mean absolute error) of the aforementioned articles for solving inverse forms of nonlinear partial differential equations related to this research.

Discussion

In this study, for the neural network described in Section Methods, we considered three hidden layers with 10, 20, and 10 nodes for each layer, respectively. We explored various numbers of hidden layers and nodes for each layer, ultimately identifying this architecture as a good trade-off between model complexity and accuracy. Additionally, the population size of the genetic algorithm was set to 16. Moreover, the number of water striders (nws) in both the IWSA and the original WSA was also set to 16. After experimenting with several values for nws, we determined that the best results were obtained when nws was set to 16.

As evidenced by Table 3 and Fig. 11, when the number of iterations reaches 10,000, the accuracy of the IWSA and the PINN reaches approximately $O(10^{-4})$, signifying high precision in solving inverse forms of nonlinear PDEs. Furthermore, both the IWSA and PINN outperformed the original WSA and the genetic algorithm. Our experiments revealed that accuracy does not significantly improve beyond 10,000 iterations, nor does it perform well for iterations fewer than 1000. Hence, we reported iteration numbers ranging from 1000 to 10,000. Additionally, it's evident from Table 3 and Fig. 11 that the genetic algorithm outperforms the original WSA for lower iterations, while the original WSA surpasses the GA for higher iterations.

According to our findings, at high iterations, the improved WSA outperforms the original WSA in terms of accuracy. This conclusion is based on implementing these algorithms to solve two different types of problems: an optimization problem (Ackley function) and an inverse form of a nonlinear partial differential equation. The main idea of this research was that including the natural process of replacing young larvae with dead insects would likely improve the optimization pattern. After implementing this concept, the results confirmed its effectiveness. We believe that following the natural placement process of newly born larvae more closely mirrors the natural evolution of these insects and improves the entire algorithm. This improvement increases the

exploration rate of the algorithm, and the initial competition between newly born larvae ensures that the fittest individuals enter the new community, leading to the overall enhancement of the proposed algorithm.

Considering execution time, as depicted in Table 4 and Fig. 12, the fastest algorithm is the GA, followed by the original WSA, then the IWSA, and finally, the slowest is the PINN. Specifically, the IWSA is almost four times faster than the PINN while maintaining almost the same accuracy. However, the drawback of the IWSA compared to the PINN is that it's a stochastic algorithm, leading to varying results in accuracy with each execution. Consequently, the advantage of the PINN lies in its result stability, maintaining consistent accuracy across different runs with the same iteration number.

Conclusion

In this paper, we introduced a novel improvement of the powerful meta-heuristic water strider algorithm and a physics-informed neural network with three hidden layers to solve an inverse form of the Burgers-Huxley equation. Given that the Burgers-Huxley equation is a nonlinear partial differential equation, the methods presented here can potentially be employed to solve a wide range of nonlinear PDEs. The methods demonstrated high accuracy and low execution times, making them suitable for real-world applications. In general, the PINN demonstrates superior accuracy across all iterations, achieving the lowest mean absolute error of 0.00023 at 10,000 iterations, compared to 0.00088 for IWSA, 0.00164 for WSA, and 0.00289 for GA. This suggests that PINN is the most precise method for approximating the exact function $f(x)$ among the algorithms tested. However, this higher accuracy comes at a higher computational cost, as PINN has the longest execution time reaching 225.73 seconds at 10,000 iterations, compared to 51.057 seconds for IWSA, 32.75 seconds for WSA, and 12.21 seconds for GA. The IWSA strikes a balance between accuracy and computational efficiency. Although slightly less accurate than PINN, IWSA significantly outperforms both the original WSA and GA in terms of accuracy, with a MAE of 0.00088 at 10,000 iterations. In terms of execution time, IWSA is approximately four times faster than PINN, making it a more practical choice when computational resources or time constraints are a consideration. Generally, the IWSA proved to be faster than the PINN; however, the PINN exhibited greater stability and consistency across different runs.

Future research could explore comparing other novel meta-heuristic algorithms and employing distinct network architectures. Additionally, the use of the Levenberg-Marquardt backpropagation algorithm for training the PINN could be considered. Exploring fractional orders of nonlinear PDEs could also be a promising area for further investigation.

Data availability

All data generated or analysed during this study are included in this published article.

Received: 21 January 2024; Accepted: 5 November 2024
Published online: 20 November 2024

References

1. Farlow, S. J. *Partial Differential Equations for Scientists and Engineers* (Wiley, 1982).
2. Isakov, V. *Inverse Problems for Partial Differential Equations Second Editions* (Springer Science+Business Media, 2006).
3. Jiang, X. et al. Physics-informed neural network for nonlinear dynamics in fiber optics. *Laser Photonics Rev.* **16**, 2100483. https://doi.org/10.1002/lpor.202100483 (2022).
4. Javidi, M. A numerical solution of the generalized Burger's-Huxley equation by spectral collocation method. *Appl. Math. Comput.* **178**, 338–344. https://doi.org/10.1016/j.amc.2005.11.051 (2006).
5. Kumar, A. & Mohan, M. Absolute continuity of the solution to stochastic generalized Burgers-Huxley equation. *Stoch. Partial Differ. Equ. Anal. Comput.*[SPACE]https://doi.org/10.1007/s40072-023-00308-7 (2023).
6. Satsuma, J. *Topics in Soliton Theory and Exactly Solvable Nonlinear Equations* (World Scientific, 1987).
7. İnan, B. & Bahadir, A. R. Numerical solutions of the generalized Burgers-Huxley equation by implicit exponential finite difference method. *J. Appl. Math. Stat. Inform.* **11**, 57–67. https://doi.org/10.1515/jamsi-2015-0012 (2015).
8. Ervin, V., Macías-Díaz, J. & Ruiz-Ramírez, J. A positive and bounded finite element approximation of the generalized Burgers-Huxley equation. *J. Math. Anal. Appl.* **424**, 1143–1160. https://doi.org/10.1016/j.jmaa.2014.11.047 (2015).
9. Pindza, E., Owolabi, M. K. & Patidar, K. Barycentric Jacobi spectral method for numerical solutions of the generalized Burgers-Huxley equation. *Int. J. Nonlinear Sci. Numer. Simul.* **18**, 67–81. https://doi.org/10.1515/ijnsns-2016-0032 (2017).
10. Nourazar, S., Soori, M. & Nazari-Golshan, A. On the exact solution of Burgers-Huxley equation using the homotopy perturbation method. *J. Appl. Math. Phys.* **3**, 285–294. https://doi.org/10.4236/jamp.2015.33042 (2015).
11. Loyinmi, A. & Akinfe, T. An algorithm for solving the Burgers-Huxley equation using the Elzaki transform. *SN Appl. Sci.*[SPACE]https://doi.org/10.1007/s42452-019-1653-3 (2020).
12. Kumar, H., Yadav, N. & Nagar, A. K. Numerical solution of generalized Burger-Huxley and Huxley's equation using deep galerkin neural network method. *Eng. Appl. Artif. Intell.* **115**, 105289. https://doi.org/10.1016/j.engappai.2022.105289 (2022).
13. Kaveh, A. & Eslamlou, A. D. Water strider algorithm: A new metaheuristic and applications. *Structures* **25**, 520–541. https://doi.org/10.1016/j.istruc.2020.03.033 (2020).
14. Raissi, M., Perdikaris, P. & Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045 (2019).
15. Khan, N. A., Laouini, G., Alshammari, F. S., Khalid, M. & Aamir, N. Supervised machine learning for jamming transition in traffic flow with fluctuations in acceleration and braking. *Comput. Electr. Eng.* **109**, 108740. https://doi.org/10.1016/j.compeleceng.2023.108740 (2023).
16. Sulaiman, M. & Khan, N. A. Predictive modeling of oil and water saturation during secondary recovery with supervised learning. *Phys. Fluids* **35**, 064110. https://doi.org/10.1063/5.0152071 (2023).
17. Sulaiman, M., Khan, N. A., Alshammari, F. S. & Laouini, G. Performance of heat transfer in micropolar fluid with isothermal and isoflux boundary conditions using supervised neural networks. *Mathematics*[SPACE]https://doi.org/10.3390/math11051173 (2023).

18. Dana Mazraeh, H. & Parand, K. Gepinn: An innovative hybrid method for a symbolic solution to the lane-emden type equation based on grammatical evolution and physics-informed neural networks. *Astron. Comput.* **48**, 100846. https://doi.org/10.1016/j.ascom.2024.100846 (2024).
19. Umar, M., Amin, F., Al-Mdallal, Q. & Ali, M. R. A stochastic computing procedure to solve the dynamics of prevention in HIV system. *Biomed. Signal Process. Control* **78**, 103888. https://doi.org/10.1016/j.bspc.2022.103888 (2022).
20. Mukdasai, K. et al. A numerical simulation of the fractional order leptospirosis model using the supervise neural network. *Alex. Eng. J.* **61**, 12431–12441. https://doi.org/10.1016/j.aej.2022.06.013 (2022).
21. Baty, H. Modelling lane-emden type equations using physics-informed neural networks. *Astron. Comput.* **44**, 100734. https://doi.org/10.1016/j.ascom.2023.100734 (2023).
22. Khan, N. A., Sulaiman, M., Kumam, P. & Alarfaj, F. K. Application of Legendre polynomials based neural networks for the analysis of heat and mass transfer of a non-Newtonian fluid in a porous channel. *Adv. Continuous Discrete Models*[SPACE]https://doi.org/10.1186/s13662-022-03676-x (2022).
23. Shahzad, A. et al. Thin film flow and heat transfer of Cu-nanofluids with slip and convective boundary condition over a stretching sheet. *Sci. Rep.* **21**, 14254 (2022).
24. Dana Mazraeh, H., Parand, K., Farahani, H. & Kheradpisheh, S. R. An improved imperialist competitive algorithm for solving an inverse form of the Huxley equation. *Iran. J. Numer. Anal. Optim.* (2024).
25. Alarfaj, F. K., Khan, N. A., Sulaiman, M. & Alomair, A. M. Application of a machine learning algorithm for evaluation of stiff fractional modeling of polytropic gas spheres and electric circuits. *Symmetry*[SPACE]https://doi.org/10.3390/sym14122482 (2022).
26. Sadaf, M., Arshed, S., Akram, G., Ali, M. R. & Bano, I. Analytical investigation and graphical simulations for the solitary wave behavior of Chaffee-Infante equation. *Results Phys.* **54**, 107097. https://doi.org/10.1016/j.rinp.2023.107097 (2023).
27. Ali, K. K., Yusuf, A., Yokus, A. & Ali, M. R. Optical waves solutions for the perturbed Fokas-Lenells equation through two different methods. *Results Phys.* **53**, 106869. https://doi.org/10.1016/j.rinp.2023.106869 (2023).
28. DanaMazraeh, H. et al. Solving Fredholm integral equations of the second kind using an improved cuckoo optimization algorithm. *Glob. Anal. Discrete Math.* **7**, 33–52. https://doi.org/10.22128/gadm.2022.447.1051 (2022).
29. Waqas, H. et al. Numerical and computational simulation of blood flow on hybrid nanofluid with heat transfer through a stenotic artery: Silver and gold nanoparticles. *Results Phys.* **44**, 106152. https://doi.org/10.1016/j.rinp.2022.106152 (2023).
30. Ali, K. K., Tarla, S., Ali, M. R. & Yusuf, A. Modulation instability analysis and optical solutions of an extended (2+1)-dimensional perturbed nonlinear Schrödinger equation. *Results Phys.* **45**, 106255. https://doi.org/10.1016/j.rinp.2023.106255 (2023).
31. Ali, K. K., Tarla, S., Ali, M. R., Yusuf, A. & Yilmazer, R. Physical wave propagation and dynamics of the Ivancevic option pricing model. *Results Phys.* **52**, 106751. https://doi.org/10.1016/j.rinp.2023.106751 (2023).
32. Abbasi Molai, A. & Dana Mazraeh, H. A modified imperialist competitive algorithm for solving nonlinear programming problems subject to mixed fuzzy relation equations. *Int. J. Nonlinear Anal. Appl.* **14**, 19–32. https://doi.org/10.22075/ijnaa.2023.28390.3876 (2023).
33. Zafar, A. et al. Exploring the new soliton solutions to the nonlinear m-fractional evolution equations in shallow water by three analytical techniques. *Results Phys.* **54**, 107092. https://doi.org/10.1016/j.rinp.2023.107092 (2023).
34. Khan, N. A., Sulaiman, M. & Alshammari, F. S. Analysis of heat transmission in convective, radiative and moving rod with thermal conductivity using meta-heuristic-driven soft computing technique. *Struct. Multidiscip. Optim.* **65**, 107092. https://doi.org/10.1007/s00158-022-03414-7 (2022).
35. Mazraeh, H. D., Pourgholi, R. & houlari, T. Combining genetic algorithm and Sinc-Galerkin method for solving an inverse diffusion problem. *TWMS J. Appl. Eng. Math.* **7**, 33–50 (2017).
36. Khan, N. A., Sulaiman, M., Seidu, J. & Alshammari, F. S. Investigation of nonlinear vibrational analysis of circular sector oscillator by using cascade learning. *Adv. Mater. Sci. Eng.* **2022**, 1898124. https://doi.org/10.1155/2022/1898124 (2022).
37. Xu, Y.-P. et al. Optimal structure design of a PV/FC HRES using amended water strider algorithm. *Energy Rep.* **7**, 2057–2067. https://doi.org/10.1016/j.egyr.2021.04.016 (2021).
38. Duan, F. & Hayati, H. Optimal fractional model identification of the polymer membrane fuel cells based on a new developed version of water strider algorithm. *Energy Rep.* **7**, 1847–1856. https://doi.org/10.1016/j.egyr.2021.03.033 (2021).
39. Liao, S. & Jimenez, G. A new optimal prediction technique for energy demand based on CNN and improved water strider algorithm: a study on socio-economic-climatic parameters. *Evol. Syst.* **13**, 759–775. https://doi.org/10.1007/s12530-021-09409-x (2022).
40. Bi, D., Liu, Y., Youssefi, N., Chen, D. & Ma, Y. Breast cancer diagnosis based on guided water strider algorithm. *Proc. Inst. Mech. Eng. Part H J. Eng. Med.* **236**, 30–42 (2021).
41. Hu, L., Zhang, Y. & Yousefi, N. Nonlinear modeling of the polymer membrane fuel cells using deep belief networks and modified water strider algorithm. *Energy Rep.* **7**, 2460–2469. https://doi.org/10.1016/j.egyr.2021.04.050 (2021).
42. Kaveh, A., Ilchi Ghazaan, M. & Asadi, A. An improved water strider algorithm for optimal design of skeletal structures. *Periodica Polytech. Civil Eng.* **64**, 1284–1305. https://doi.org/10.3311/PPci.16872 (2020).
43. Liu, B. & Pouramini, S. Multi-objective optimization for thermal comfort enhancement and greenhouse gas emission reduction in residential buildings applying retrofitting measures by an enhanced water strider optimization algorithm: A case study. *Energy Rep.* **7**, 1915–1929. https://doi.org/10.1016/j.egyr.2021.03.044 (2021).
44. Syah, R. et al. Optimal parameters estimation of the PEMFC using a balanced version of water strider algorithm. *Energy Rep.* **7**, 6876–6886. https://doi.org/10.1016/j.egyr.2021.10.057 (2021).
45. Holland, J. *Adaptation in Natural and Artificial Systems* (University of Michigan Press, 1975).
46. Babolian, E. & Saeidian, J. Analytic approximate solutions to Burgers, Fisher, Huxley equations and two combined forms of these equations. *Commun. Nonlinear Sci.*[SPACE]https://doi.org/10.1016/j.cnsns.2008.07.019 (2009).
47. Alkhadhr, S. & Almekkawy, M. A combination of deep neural networks and physics to solve the inverse problem of burger's equation. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 4465–4468, https://doi.org/10.1109/EMBC46164.2021.9630259 (2021).
48. Pakravan, S. A., Mistani, P., Aragon-Calvo, M. A. & Gibou, F. Solving inverse-PDE problems with physics-aware neural networks. *J. Comput. Phys.* **440**, 110414. https://doi.org/10.1016/j.jcp.2021.110414 (2021).
49. Aliyari Boroujeni, A., Pourgholi, R. & Tabasi, S. A new improved teaching-learning-based optimization (ITLBO) algorithm for solving nonlinear inverse partial differential equation problems. *Comput. Appl. Math.*[SPACE]https://doi.org/10.1007/s40314-023-02247-4 (2023).

Acknowledgements

The result was created through solving the student project "Wireless networks security focusion on IoT" using objective oriented support for specific university research from the University of Finance and Administration, Prague, Czech Republic. Authors thank Zdenek Truhlar, Manuel Rosinus, Mykola Hrynevyc, Patrick Mini, Jenna Aurelie Huppertz for their help with the research connected with the topic of the article.

Author contributions

H.D.M conceived the main idea of this research and implemented the algorithms. K.P. analyzed the results and contributed to writing the manuscript. M.H. conducted the experiments and analyzed the results, while J.L. and V.N. also participated in the analysis of the results. All authors reviewed the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to K.P.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.