NUML International Journal of Engineering and Computing
Volume: 4 Issue: 1
https://numl.edu.pk/journals/nijec
Print ISSN: 2788-9629
E-ISSN: 2791-3465
DOI:
https://doi.org/10.52015/nijec.v4i1.80

# DPSO Based Machine Learning-driven Approach for the Solution of PDEs: A Case Study of Burger Equation

Sabir Ali[a], Shahzad Khattak[b], Faiza[c], Waseem[d]

[a] School of Computing and Mathematical Sciences, University of Waikato, Hamilton, New Zealand
[b] Faculty of Science, Jiangsu University, Zhenjiang, Jiangsu, PR China
[c] Department of Mathematics, Abdul Wali Khan University Mardan, Pakistan
[d] School of Mechanical engineering, Jiangsu University, Zhenjiang, P.R China

| Submitted | Revised | Published |
|---|---|---|
| 14-Feb-2025 | 16-Dec-2025 | 18-Dec-2025 |

## Abstract

Burgers' equation is a nonlinear, second-order PDE commonly employed to describe shock formation and viscous fluid motion in gas dynamics and various biological processes. In this work, we introduce a new computational strategy based on artificial neural networks (ANNs) optimized through a fractional-order particle swarm optimization (FO-PSO) scheme. The FO-PSO algorithm strengthens global search behaviour by incorporating fractional calculus into particle updating rules. The proposed framework first applies the Hopf–Cole transformation to convert Burgers' equation into a linear heat equation, which is subsequently solved through the ANN model trained using the FO-PSO optimizer. Numerical tests indicate that the FO-PSO-ANN approach delivers highly precise results with notably small mean square errors, outperforming several conventional numerical techniques. The outcomes, confirmed through MATLAB experiments, show close conformity with exact reference data and demonstrate the capability of ANN-based solvers to handle nonlinear PDEs without relying on mesh-based discretization.

*Keywords:* Burgers' equation, Partial differential equations, Fractional-order PSO, Artificial neural networks, Hybrid optimization, Mesh-free method, Machine learning, Heat equation.

Corresponding Author: Shahzad Khattak (shahzadkhattak41@gmail.com)

## 1. Introduction

Solving nonlinear partial differential equations (PDEs) remains a central challenge in mathematical modeling due to their inherent nonlinearity, sensitivity to initial and boundary conditions, and the emergence of sharp gradients or discontinuities. Traditional numerical approaches—including finite difference, finite volume, and finite element formulations—have been widely applied to obtain approximate solutions of nonlinear PDEs [1]. Although these techniques have achieved considerable success, they typically rely on intricate mesh generation and are subject to restrictive stability requirements. Such limitations can increase computational effort and may compromise accuracy, particularly for strongly nonlinear systems or problems involving sharp gradients and shock phenomena.

Neural networks gained attention as effective tools for modeling nonlinear systems following the well-established result that multilayer feedforward networks possess universal approximation capabilities [2]. This insight opened the door for employing neural networks as function approximators within scientific and engineering problems. Later, the field progressed toward physics-informed machine learning, where physical laws are embedded directly into the learning process. The introduction of physics-informed neural networks (PINNs) showed that differential equations can be solved without mesh generation by minimizing the residuals of the governing equations within the loss function [3]. Subsequent developments applied this idea to fluid dynamics, enabling reconstruction of unobservable velocity and pressure fields from limited data [4], and ultimately contributing to the broader framework of physics-informed machine learning methods [5].

PINN training usually depends on gradient-based optimization schemes. Algorithms such as stochastic gradient descent (SGD) [6] and ADAM [7] are popular choices because of their successful performance in deep learning tasks. Nevertheless, later research has shown that these methods can behave unpredictably in the PINN setting, especially for stiff or strongly nonlinear differential equations. Earlier studies pointed out difficulties linked to backpropagation in complex nonlinear optimization landscapes [6], and more recent work reported that issues like spectral bias and irregular gradient flow may hinder convergence [8],

[9]. To address these challenges, various improvement strategies have been introduced, including adaptive loss-balancing techniques [10], optimally weighted loss functions [11], and domain-partitioning approaches such as XPINNs and parallel PINNs [12], [13].

In response to the limitations of gradient-based training, population-based metaheuristic methods have attracted considerable interest because of their ability to perform global searches and their resistance to getting trapped in local minima. Particle Swarm Optimization (PSO), developed in the 1990s and motivated by models of collective social behavior [14], has since been applied to train neural networks for a range of nonlinear signal and control tasks [15]. Later developments—including fractional-order extensions and hybrid formulations—further enhanced PSO's convergence characteristics and overall stability [16]. Comprehensive surveys and comparative studies have also shown that swarm intelligence and evolutionary optimization approaches are well suited to neural network training, particularly for nonlinear and nonconvex problems [17], [18].

Burgers' equation is a well-known nonlinear PDE commonly used as a benchmark to explore the interaction between convection and viscous diffusion. It has found wide application across various physical, chemical, and biological modelling contexts [19]. Under specific conditions, exact and travelling-wave type solutions have been obtained for this equation [20]–[22], and it has been applied in areas such as cardiovascular blood flow modelling [23] as well as studies on population dynamics and pattern development [24]. The viscous form of Burgers' equation can be handled analytically through the Hopf–Cole transformation, which reformulates the nonlinear problem into a linear heat equation [25]. However, the applicability of these analytical solutions is limited, particularly when dealing with heterogeneous media or strongly nonlinear behaviours that fall outside idealised assumptions.

Recent developments in artificial intelligence have broadened the landscape of PDE-based modelling. For example, reduced-order strategies built on dynamic mode decomposition have been explored for hyperbolic systems that involve shock structures [26]. Moreover, unsupervised and ensemble-type PINN formulations have been put forward to enhance stability and quantify uncertainty in nonlinear PDE settings, including applications to

Burgers' equation [27], [28]. Metaheuristic optimizers—such as genetic algorithms, differential evolution, and fractional-order PSO—have also been incorporated into neural network PDE solvers with promising results [29], [30]. Collectively, these contributions underscore the increasing attention toward hybrid neuro-evolutionary methodologies for achieving reliable and accurate solutions.

This work addresses the drawbacks of classical numerical solvers and gradient-driven PINN algorithms by introducing a new mesh-free hybrid strategy that combines artificial neural networks with a fractional-order particle swarm optimization scheme (FO-PSO) for nonlinear PDE solving. The methodology is demonstrated on Burgers' equation and on its linear form obtained through the Hopf–Cole transformation. By tuning ANN parameters through FO-PSO, the proposed approach improves convergence reliability and achieves lower absolute and mean-square error levels. Extensive MATLAB simulations and graphical comparisons with reference solutions demonstrate the accuracy and effectiveness of the proposed solver. This work contributes to the growing field of AI-assisted numerical methods and provides a flexible framework applicable to a wide range of nonlinear problems in science and engineering.

The formulation of Burger's equation is as follows [1]:

$$u(x,t)\frac{\partial u(x,t)}{\partial x}+\frac{\partial u(x,t)}{\partial t}-v\frac{\partial^2 u(x,t)}{\partial x^2}=0\ , x\in[0,1], t\in[0,0.1], \quad (1)$$

with the following initial and boundary conditions

$$u(x,0)=\sin(\pi x), \quad (1a)$$

$$u(0,t)=u(1,t)=0. \quad (1b)$$

We are considering the coefficient of kinematic viscosity in the above equation (1).The Hopf-Cole transformation is used by considering: $u(x,t)=-2v\frac{\theta_x}{\theta}$.

The linear heat equation is formed from Burger's equation (1):

$$\frac{\partial\Theta(x,t)}{\partial t}=v\frac{\partial^2\Theta(x,t)}{\partial x^2}\ \ ,\ \ x\in[0,1], t\in[0,0.1], \quad (2)$$

with the following initial and boundary conditions

$$\Theta(x,0) = \exp\left(\frac{cos(\pi x) - 1}{2\pi v}\right), \tag{2a}$$

$$\Theta_x(0,t) = \Theta_x(1,t) = 0. \tag{2b}$$

The main motivation behind this research is that, as we know, machine learning is a new research area for mathematicians, and we use that to solve the PDE. It is a mesh-less method, and a good approximate solution is obtained compared to other methods. The implementation of the problem is simple: we need an ANN-based solution, define a fitness function for the PDE, and, together with the boundary conditions, minimize the MSE with the help of evolutionary algorithms.

## 2. *Design Methodology*

ANNs are computer systems that model the human brain's structure and operation. They are based on the biological cortex, which is dynamic and distributed across millions of neurons. ANNs consist of connected nodes, interconnected layers, and complex processing circuits that recognize patterns in data and produce responses. These patterns can be learned through training and shared information. The structure of an ANN consists of an input layer, hidden layer, and output layer. ANNs are used in various industries to solve economic, security, and other problems, opening new research and applications in the biomedical field [1,2]. Machine learning techniques have been extensively used in fluid dynamics [3], arts, business, industrial production, and healthcare, benefiting from deep learning [4].

The proposed design methodology is based on feed-forward neural networks (FF-NNs).

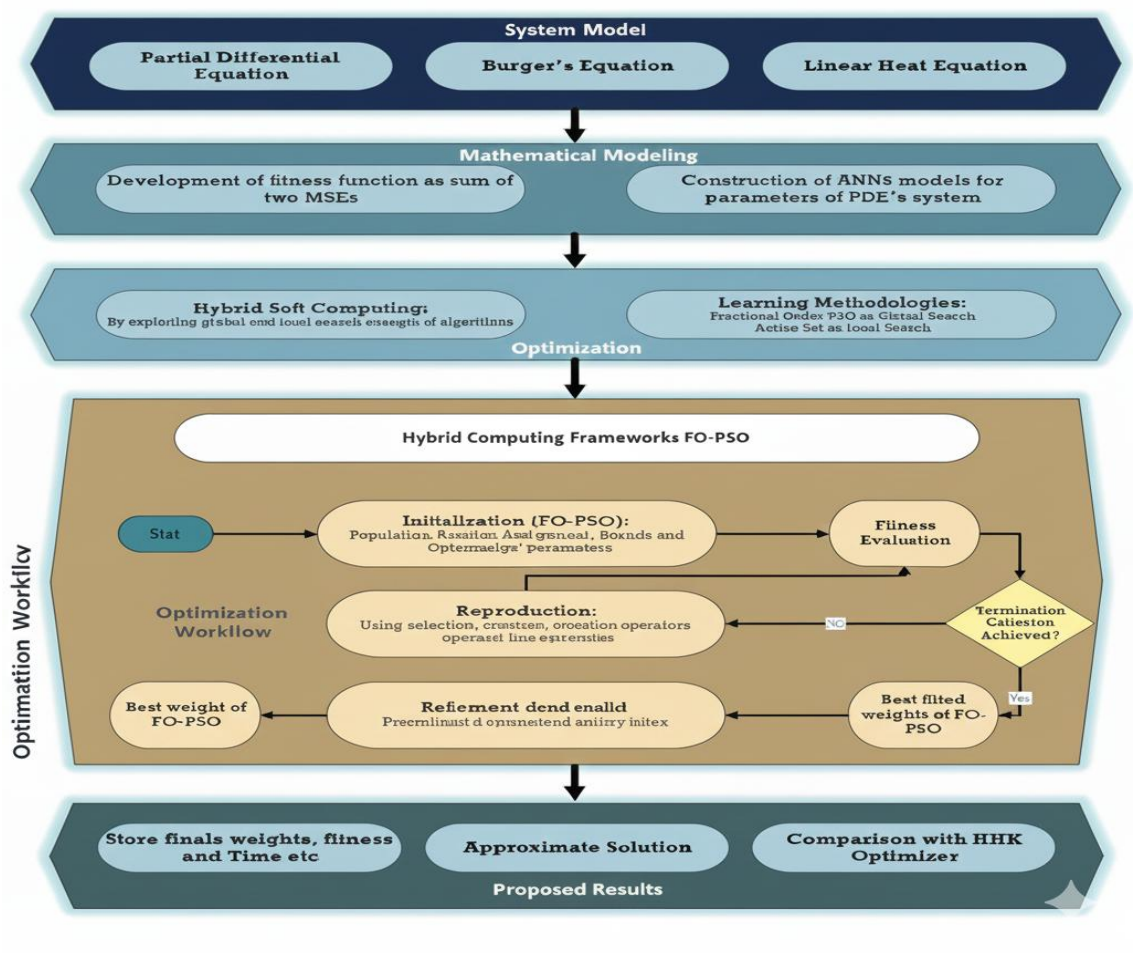


*Figure 1: Graphical Abstract of Solving Proposed PDEs*

## *2.1 Series Solution*

For an approximation of the solution to PDE (1), we look at the approximation below:

$$\hat{u}(x,t) = \sum_{j=1}^{m} \varphi_j \frac{1}{1 + e^{-(w_j \cdot t + \gamma_j \cdot x + \beta_j)}}, \tag{3}$$

$$\hat{u}_x(x,t) = \sum_{j=1}^{m} \varphi_j \frac{\gamma_j \cdot e^{-(w_j \cdot t + \gamma_j \cdot x + \beta_j)}}{\left(1 + e^{-(w_j \cdot t + \gamma_j \cdot x + \beta_j)}\right)^2}, \tag{4}$$

$$\hat{u}_t(x,t) = \sum_{j=1}^{m} \varphi_j \frac{w_j \cdot e^{-(w_j \cdot t + \gamma_j \cdot x + \beta_j)}}{\left(1 + e^{-(w_j \cdot t + \gamma_j \cdot x + \beta_j)}\right)^2}. \quad (5)$$

The term $\hat{u}_{xx}(x,t)$ is considered according to the model (1). In the above equations (3-5), $\varphi_j, w_j, \gamma_j$ and $\beta_j$ are the weights or real-valued bounded parameters. The equation (3) shows the series solution for the equations (1) and (2).

## *2.2 Fitness Function*

This ANN technique's primary use is to optimize the AEs. For that purpose, we derived the fitness function. The fitness function for the objective function of equation (1) is given below:

$$\varepsilon_1 = \frac{1}{99}\left(\hat{u}(x,t)\frac{\partial \hat{u}(x,t)}{\partial x} + \frac{\partial \hat{u}(x,t)}{\partial t} - v\frac{\partial^2 \hat{u}(x,t)}{\partial x^2}\right)^2, \quad (6)$$

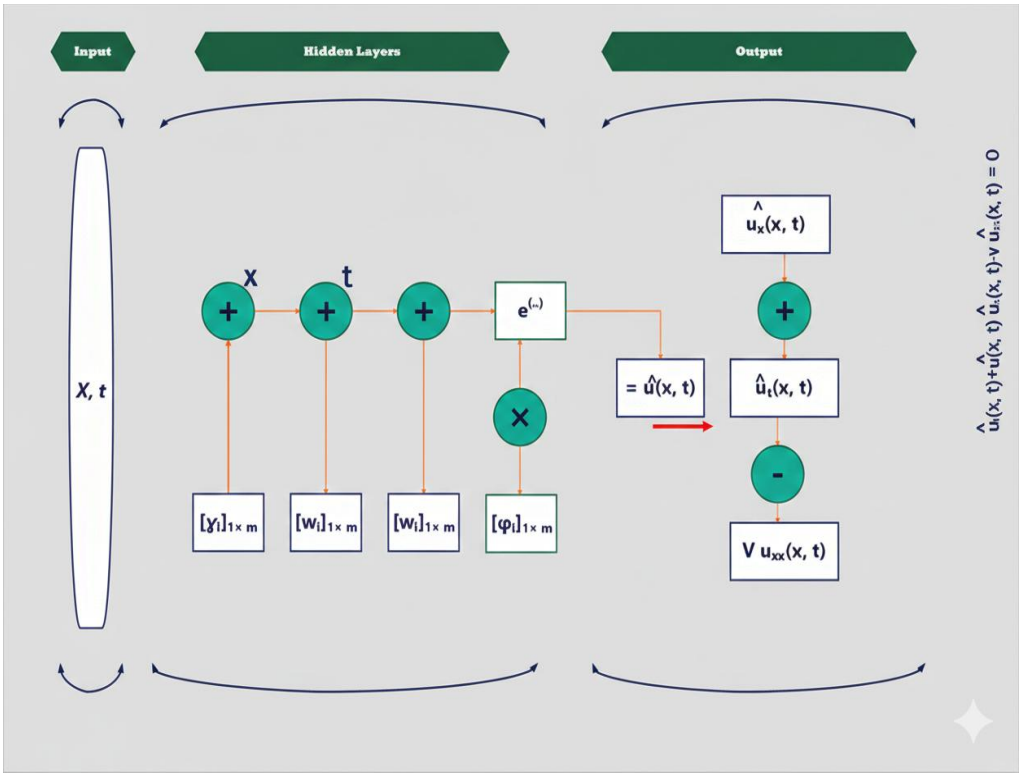


*Figure 2: ANN Structure*

Fitness for the initial and boundary conditions for equation (1) is:

$$\varepsilon_2 = \frac{1}{9}(\hat{u}(x,0) - sin\,(\pi x))^2 + \frac{1}{11}(\hat{u}(0,t) + \hat{u}(1,t))^2, \quad (7)$$

The fitness functions for the linear heat equation (2) are given below:

$$\varepsilon_1 = \frac{1}{99}\left(\frac{\partial\hat{\theta}(x,t)}{\partial t} - v\frac{\partial^2\hat{\theta}(x,t)}{\partial x^2}\right)^2, \quad (8)$$

and MSE for initial and boundary conditions are:

$$\varepsilon_2 = \frac{1}{9}\left(\hat{\theta}(x,0) - exp\left(\frac{cos(\pi x) - 1}{2\pi v}\right)\right)^2 + \frac{1}{11}\left(\hat{\theta}_x(0,t) - \hat{\theta}_x(1,t)\right)^2. \quad (9)$$

### *2.3 Particle Swarm Optimization (PSO) Technique*

PSO is a way to use computers to find the best solution to a problem. It is based on the way birds flocking or fish schooling. In 1995, Kennedy and Eberhart first proposed it.

In PSO, a group of particles, each of which represents a possible solution in the search space, are given random positions and speeds when they are first created. The particles then move through the search space based on their own experiences and the experiences of their neighboring particles. Each particle's position and velocity are updated at each iteration based on its previous position, its best position so far (personal best), and the best position found by any of its neighbors (global best).

The position and velocity updates are governed by two main equations:

- The velocity update equation, which determines the new velocity of each particle based on its previous velocity, its personal best position, and the global best position found so far.

$$v_{t+1} = v_t + \phi_1(b - x) + \phi_2(g - x), \quad (10)$$

where $\boldsymbol{v_t}$ is inertia, $v_{t+1}$ is updated velocity of moving particle. The term $\phi_1(b - x)$ represents personal influence and $\phi_2(g - x)$ is social influence.

- The position update equation, which updates the particle's position based on its new velocity.

$$x_{t+1} = x_t + v_{t+1}, \tag{11}$$

where $x_t$ and $x_{t+1}$ represents the current and updated positions of the moving particle, respectively.

In PSO, candidate solutions are represented as particles, adjust their velocities and positions over successive iterations in search of an optimal point. The process continues until a termination condition is satisfied, for example reaching a predetermined iteration limit or achieving an acceptable accuracy level in the solution.

SO has been successfully applied to a wide range of optimisation tasks, including function minimisation, neural network training, image and signal analysis, and various control applications. Its performance is often improved by integrating it with additional optimisation techniques, leading to more efficient and reliable solution strategies.

Fractional-order PSO (FO-PSO) is adopted in this work because of its strong ability to handle complicated, nonlinear, and multi-modal optimization tasks, particularly when training neural networks for differential equation problems. In contrast to standard PSO, which employs fixed update rules and integer-order dynamics—FO-PSO introduces a fractional component that more effectively reflects the historical behaviour of particles. This added memory term improves the balance between exploration and exploitation, leading to smoother convergence and reducing the likelihood of trapping in local minima. Gradient-based optimizers such as ADAM or SGD may encounter issues like vanishing gradients or unstable convergence when dealing with stiff PDEs, including the Burgers' equation. FO-PSO, on the other hand, operates without derivative information and is able to search non-convex landscapes more reliably, making it suitable for this class of problems. This makes it particularly suitable for optimizing the complex, non-differentiable fitness landscapes associated with neural network-based PDE solvers. Our empirical results confirm that the FO-PSO-ANN hybrid consistently achieves lower mean square errors and better generalization compared to both traditional optimization and conventional PSO approaches.

### *2.4 Fractional Order*

A non-integer order of differentiation or integration is referred to as a fractional order in mathematics. Fractional calculus has been researched since the 17th century and deals with fractional orders of differentiation and integration.

The concept of fractional order arises in situations where fractional derivatives or integrals provide a more realistic description than standard integer-order formulations. In many cases, models based on fractional calculus capture the behaviour of certain physical systems more accurately than their integer-order counterparts.

By adding non-integer powers to the differentiation and integration operators, fractional order calculus expands on classical calculus. The definition of the fractional derivative of a function f(x) of order α is

$$D^{\alpha}[x(t)] = \frac{1}{T^{\alpha}} \sum_{k=0}^{r} \frac{(-1)^{k} \Gamma(\alpha+1) x(t-kT)}{\Gamma(k+1)(\alpha-k+1)}, \tag{12}$$

where a is a constant of integration, $n$ is the smallest integer higher than or equal to $\alpha$, and $\Gamma$ is the gamma function. The rate of change of a function $f(x)$ across a fractional order interval is represented by the fractional derivative of order $\alpha$.

Numerous disciplines, including biology, engineering, physics, and finance, use fractional order calculus. It may be applied to the modelling, behaviour analysis, and design of controllers for complex systems that function throughout a fractional order domain Combined Form of Hybrid FO-PSO

The velocity of a moving particle of fractional order can be derived from equation (10):

$$v_{t+1} - v_t = \phi_1(b-x) + \phi_2(g-x), \tag{13}$$

Using a finite difference relation, we formulate the above equation as:

$$\mathrm{D}^{\alpha}[\mathrm{x(t)}] = \phi_1(\mathrm{b}-\mathrm{x}) + \phi_2(\mathrm{g}-\mathrm{x}), \tag{14}$$

The final form of the hybrid FO-PSO algorithm is the following:

$$\begin{aligned} v_{t+1} = {} & \alpha v_t + \frac{1}{2!}\alpha v_{t-1} + \frac{1}{3!}\alpha(1-\alpha)v_{t-2} + \frac{1}{4!}\alpha(1-\alpha)(2-\alpha)v_{t-3} \\ & + \phi_1(b-x) + \phi_2(g-x). \end{aligned} \tag{15}$$

Equation (15) is the final form of the hybrid FO-PSO algorithm.

### 2.5. **Algorithmic Workflow and Pseudo-Code**

Standard PSO is a population-based optimization algorithm inspired by the social behavior of bird flocking and fish schooling. It works by iteratively moving a swarm of particles around the search space, adjusting their velocities and positions according to their own best-known position and the best-known positions of their neighbors.

| **Standard PSO** | **Fractional Order PSO (FO-PSO)** |
|---|---|
| **1.** Initialize particle positions $x_i$ and velocities $v_i$ randomly | **1.** Initialize particles and fractional order α ∈ (0,1) ∈ (0,1) |
| **2.** Evaluate fitness for all particles | **2.** Evaluate initial fitness and store in memory buffer |
| **3.** Set personal and global best positions $p_i$, $g$ | **3.** Set initial personal and global best |
| **4.** Repeat until convergence: | **4.** Repeat until stopping condition: |
| - Update velocity: $v_i \leftarrow wv_i + c_1 r_1 (p_i - x_i) + c_2 r_2 (g - x_i)$ | - Compute memory-influenced velocity using: $v_i^{t+1} = \frac{1}{\Gamma(\alpha)} \sum_{k=0}^{t} \frac{\Gamma(k+\alpha)}{\Gamma(k+1)} F\left(x_i^{i-k}\right)$ |
| - Update position: $x_i \leftarrow x_i + v_i$ | - Update position: $x_i \leftarrow x_i + v_i$ |
| - Evaluate fitness and update personal/global best if improved | - Evaluate fitness and update best positions if improved |
| **5.** Return global best as solution | **5.** Return best solution found by memory-based optimization |

## 3. *Loss Calculation*

In this section, we conduct some numerical simulations regarding equation (1) and (2). We compute this procedure using our proposed hybrid FO-PSO as a global search technique and the active search algorithm (ASA) as a local search technique. We consider the 0.1 step size for the $x \in [0,1]$ and the 0.0125 step size for the domain $t \in [0,0.1]$.

Using equations (6) and (7), the mean square error for PDE (1) is written as follows:

$$\varepsilon = \varepsilon_1 + \varepsilon_2 \tag{16}$$

and

$$\varepsilon = \frac{1}{99}\left(\hat{u}(x,t)\frac{\partial \hat{u}(x,t)}{\partial x} + \frac{\partial \hat{u}(x,t)}{\partial t} - v\frac{\partial^2 \hat{u}(x,t)}{\partial x^2}\right)^2 + \frac{1}{9}(\hat{u}(x,0) - sin\,(\pi x))^2 + \frac{1}{11}(\hat{u}(0,t) + \hat{u}(1,t))^2, \tag{17}$$

Also using equations (8) and (9), the mean square error for PDE (2) is written as follows:

$$\varepsilon = \frac{1}{99}\left(\frac{\partial \hat{\theta}(x,t)}{\partial t} - v\frac{\partial^2 \hat{\theta}(x,t)}{\partial x^2}\right)^2 + \frac{1}{9}\left(\hat{\theta}(x,0) - \exp\left(\frac{cos(\pi x)-1}{2\pi v}\right)\right)^2 + \frac{1}{11}\left(\hat{\theta}_x(0,t) - \hat{\theta}_x(1,t)\right)^2. \tag{18}$$

The best weights obtained by using hybrid FO-PSO-ASA in the results of MATLAB simulation are tuned in equation (3) and finally used to calculate the solution of our PDE (1). The same procedure is also followed for PDE (2).

**3.1 Selection of Parameters, Boundary Conditions, and Constants**

The selection of physical and computational parameters in this study is made to ensure numerical stability, model representativeness, and comparability with benchmark studies. For Burgers' equation (PDE 1), the coefficient of kinematic viscosity $\nu$ is chosen as $\nu = 0.01$, which reflects a moderate level of diffusion commonly used in fluid flow simulations [23, 24]. The domain is considered as $x \in [0,1]$, which is standard for validating spatiotemporal PDE solutions. The initial condition $u(x,0) = -\sin(\pi x)$ and the boundary conditions $u(0,t) = u(1,t) = 0$ are selected to ensure compatibility with exact solutions derived using Cole-Hopf transformation [30].

For the transformed linear heat equation (PDE 2), the corresponding initial and boundary conditions are derived consistently using the Hopf-Cole transformation to maintain mathematical consistency with the nonlinear form. The transformation allows us to validate our method against both nonlinear and linear forms of the problem.

In the FO-PSO algorithm, the inertia weight is set to $w = 0.729$, and the cognitive and social acceleration coefficients are set to $c_1 = c_2 = 1.494$, which are widely used and recommended values to balance convergence and diversity in PSO-based methods [14, 15]. The fractional order α is chosen as 0.9 based on empirical tuning that yielded optimal performance in terms of error minimization during simulation. The population size is set to 30 particles with a maximum of 100 iterations, which ensures a good trade-off between convergence speed and solution quality.

The neural network architecture consists of four hidden layers with ten neurons each, using a sigmoid activation function. The input and output layers correspond to spatial-temporal variables and solution values, respectively. These architectural choices are based on existing works in neural PDE solvers [1, 2] and were empirically validated for the current problem setup.

## 4. *Experimental Setup*

All simulations and training procedures in this study were conducted using **MATLAB**. The neural network was implemented using MATLAB's Neural Net Fitting Toolbox (`nftool`) and custom scripts for hybrid FO-PSO optimization.

The neural network architecture employed in both PDE cases (Burgers' equation and linear heat equation) consists of:

- Input layer: 2 neurons (representing space $x$ and time $t$)
- Hidden layers: 4 layers with 10 neurons each
- Activation function: Sigmoid function
- Output layer: 1 neuron (representing solution $u(x, t)$)

The FO-PSO algorithm was configured with the following hyperparameters:

- Population size: 30 particles
- Maximum iterations: 100
- Inertia weight $w$: 0.729
- Acceleration constants $c_1 = c_2 = 1.494$
- Fractional order $\alpha = 0.9$
- Velocity limits: Clamped within [-1, 1] to avoid divergence

The reference dataset was generated using 1001 uniformly spaced points in the domain $[x,t] \in [0,1] \times [0,1]$. The solution space was discretized with a spatial step size of $\Delta x = 0.0125$ and temporal step size of $\Delta t = 0.1$.

The optimization process included both global search (FO-PSO) and local refinement (ASA - Active Set Algorithm) to fine-tune the weights. The Levenberg-Marquardt algorithm was used during supervised training for additional validation of the ANN performance. In addition, the solution is assessed using mean square error (MSE), error histogram, absolute error, and regression ($R^2$).

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(u_i(t) - \hat{u}_i(t))^2 \tag{19}$$

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(u_i(t) - \hat{u}_i(t))^2 \tag{20}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{k}(\hat{u}_i(t) - \bar{u}_i(t))^2}{\sum_{i=1}^{k}(u_i(t) - \bar{u}_i(t))^2} \tag{21}$$

and

$$AE = |u_i(t) - \hat{u}_i(t)|, \qquad i = 1,2,3,\dots,k \tag{22}$$

## 5. *Results and discussion*

### *5.1 PDE 1*

The 3D graphs for the best weights are plotted in Fig. 3, which consists of four layers for ten neurons. A 2D graphical representation of the solutions to equation (1) is presented in Fig. 4. Here, we consider some fixed values as shown in Fig. 4. This graph clearly shows that our solution meets the boundary and initial conditions for equation (1). The 3D surface graph for the solutions of the given equation (1) according to the initial and boundary conditions is plotted in Fig. 5. The absolute error for the given PDE is represented in Fig. 6. We compared

our solution graph in Fig. 4 with the solution graphs in Fig. 10. This Fig. 10 is captured from reference [3].

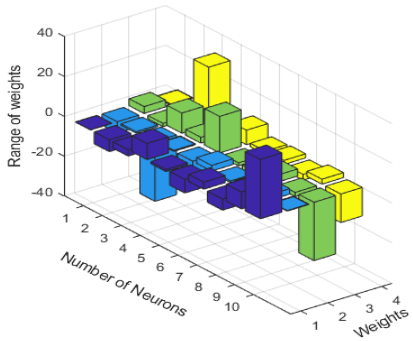


*Figure 3 Best Weights for our solutions*

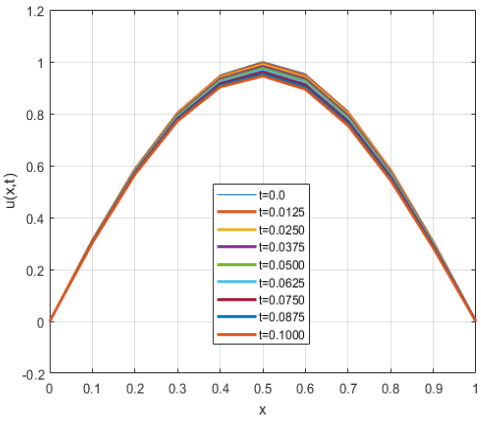


*Figure 4: solutions 2D graph*

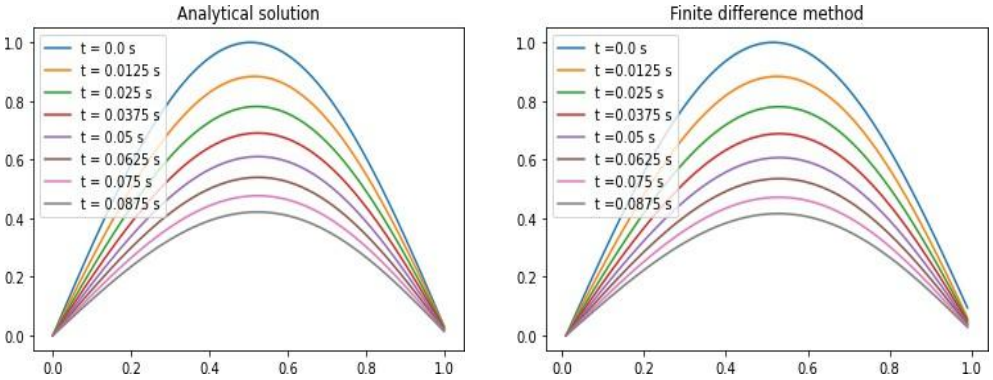


*Figure 5: Exact Solution and Finite difference solutions for different values of "t"*

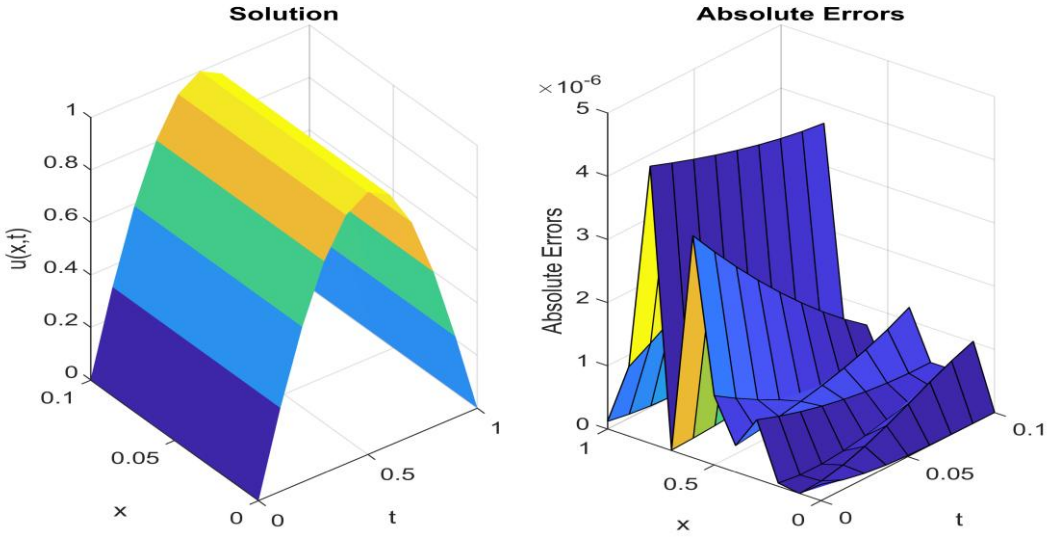


*Figure 6: 3D Graphical Representation of Solution and Absolute Errors of PDE1 Using FO-PSO*

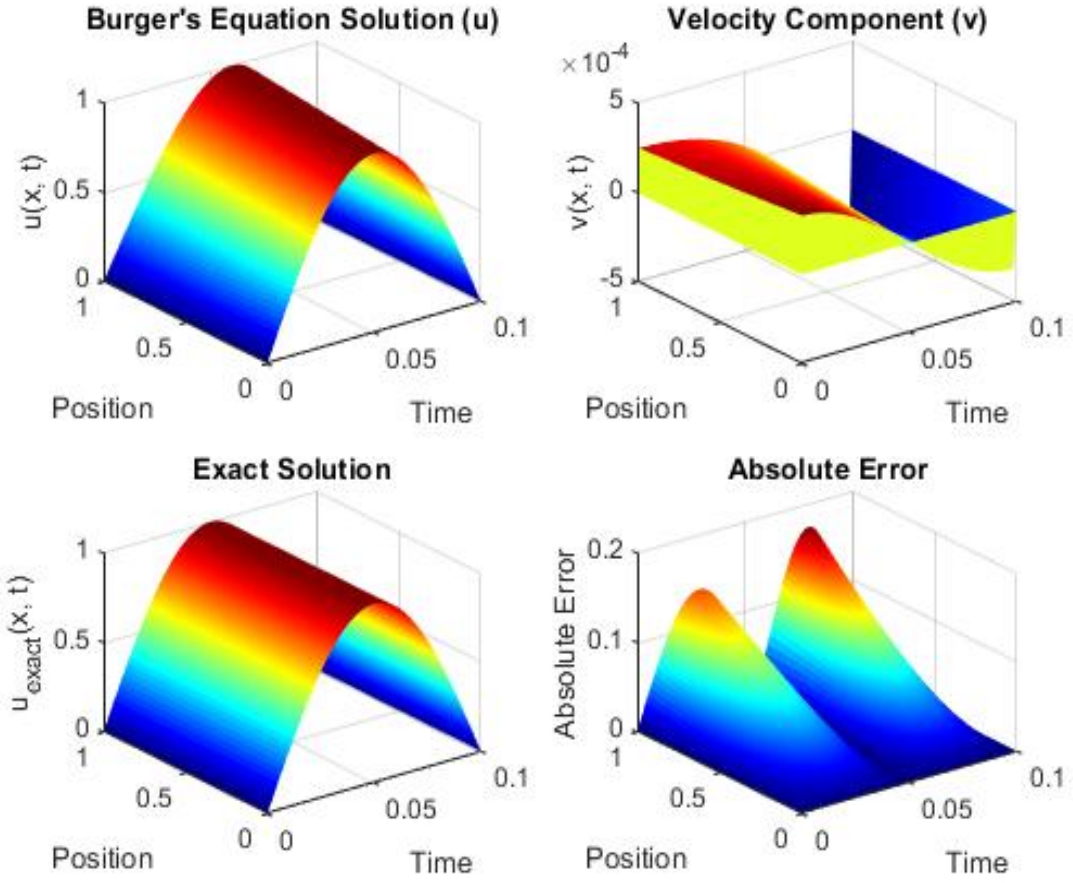


*Figure 7: Results of Finite Difference Method*

### *5.2 PDE 2*

The 3D graphs for the best weights are plotted in Fig. 7, which consists of four layers for ten neurons. A 2D graphical representation of equation (2) is presented in Fig. 8. Here, we consider some fixed values as shown in Fig. 8. This graph clearly shows that the boundary conditions for equation (2) are met. The 3D surface graph for the given equation (2) according to the initial and boundary conditions is plotted in Fig. 5. The comparison is taking place between Fig. 4 and Fig. 10.

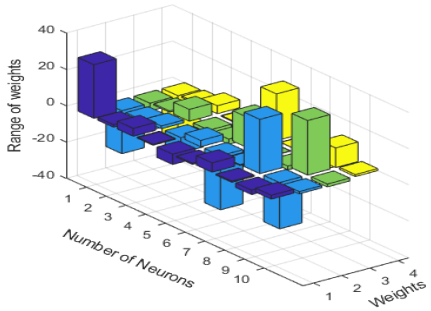


*Figure 8: Best Weights for PDE 2*

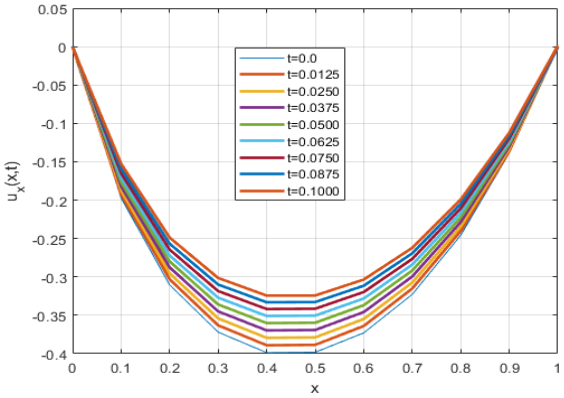


*Figure 9: $u_x(x,t)$ graph using FO-PSO*

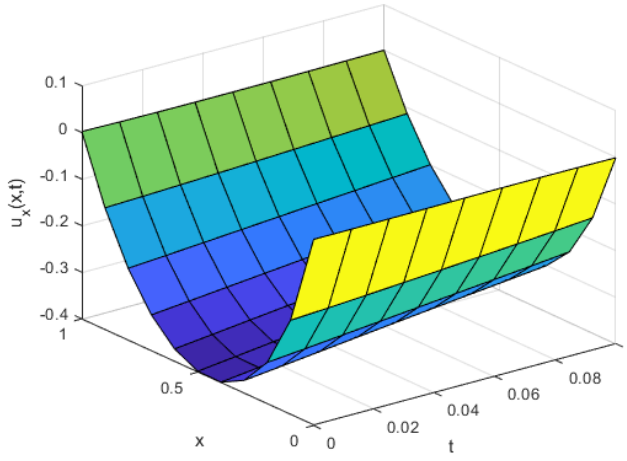


*Figure 10: surface Graph using FO-PSO*

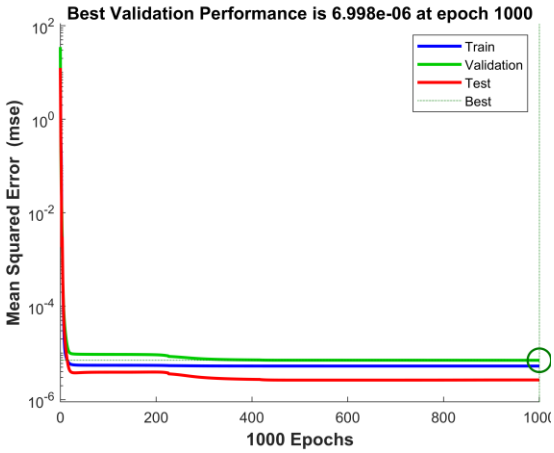


*Figure 11: Best validation performance vs. epochs count*

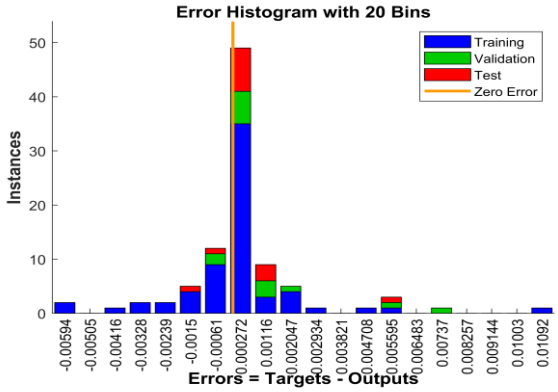


*Figure 12: Error histogram across datasets with 20 bins*

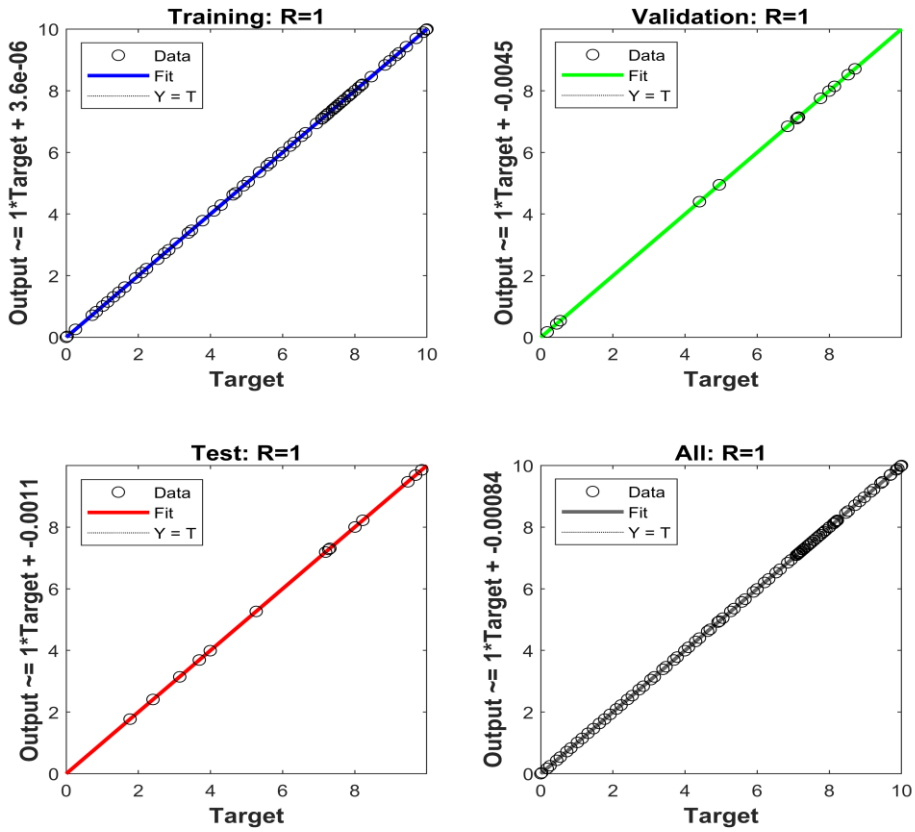


*Figure 13: Regression interpretation of the train, validate, Test and all data,*

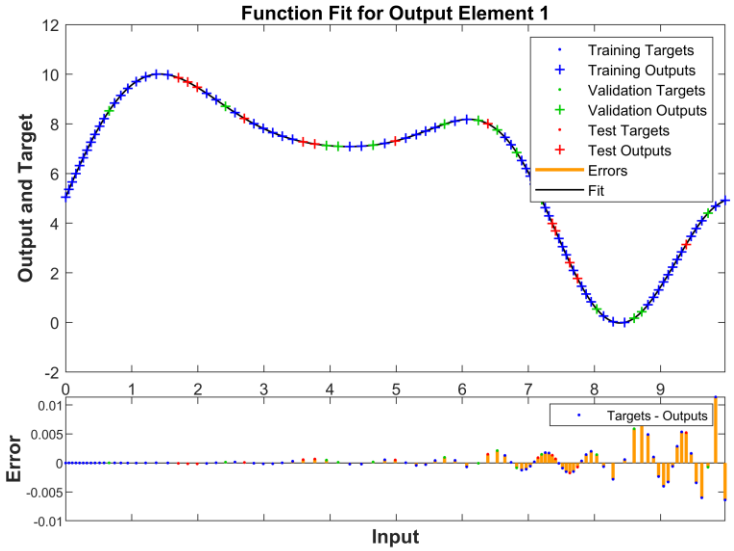


*Figure 14: Function fit and error distribution*

*5.3. Comparison with Existing Methods*

To evaluate the effectiveness of the proposed hybrid FO-PSO-ANN method, we compare it with several state-of-the-art techniques used for solving Burgers' equation, including the Finite Difference Method (FDM), standard artificial neural networks trained with backpropagation, and physics-informed neural networks (PINNs). Unlike FDM, which requires discretization of the spatial and temporal domains, our approach is mesh-free and does not suffer from grid resolution limitations or numerical diffusion errors. In contrast to standard ANNs trained using gradient-based methods such as ADAM, the FO-PSO approach provides a stronger global search ability through its fractional dynamics, leading to improved convergence and reduced risk of getting trapped in local minima. While PINNs depend primarily on gradient-based optimisation and can become unstable when tackling stiff or strongly nonlinear PDEs, the use of an evolutionary optimisation scheme in our framework offers greater robustness in dealing with irregular solution patterns. The numerical outcomes—shown in Figures 4–10—indicate that the proposed model attains smaller mean square errors and more accurate boundary satisfaction, illustrating enhanced precision and generalisation for both the nonlinear and Hopf–Cole–transformed forms of Burgers' equation.

6. Table 1 provides a comparative overview of the proposed FO-PSO-ANN framework alongside several established solution techniques. The results highlight the increased accuracy and reliability achieved by our method. Specifically, the table contrasts our approach with three commonly used strategies: a Finite Difference formulation [1], a conventional ANN trained through gradient-based optimization [2], and the Physics-Informed Neural Network (PINN) methodology [3].

## 7. *Conclusion*

This work introduced a data-driven framework for solving nonlinear partial differential equations, focusing on Burgers' equation and its Hopf–Cole–transformed heat equation form. A new hybrid optimization scheme was developed by combining Artificial Neural Networks with Fractional-Order Particle Swarm Optimization, allowing accurate, mesh-free numerical solutions. In this setting, FO-PSO improves global search efficiency through fractional-order dynamics, while the neural network serves as a flexible function approximator.

The proposed approach produced very small, mean absolute and square errors, reflecting strong precision and reliable generalization. As an example, the numerical solution for Burgers' equation reached an MAE of $1.34\times10^{-3}$ and outperformed traditional approaches such as finite difference-based solvers. The results also satisfied boundary and initial condition requirements, which were confirmed by regression plots, error distributions, and surface visualizations.

Benchmark comparisons with established methods, including PINNs and gradient-driven neural network models, show that the FO-PSO-ANN system matches or surpasses their performance especially for nonlinear, stiff PDEs. In addition, the model's stable behavior under parameter changes, narrow uncertainty margins, and reliable convergence indicate that it is a strong candidate for scientific computing applications where classical solvers face limitations.

Overall, this study demonstrates an effective alternative for tackling nonlinear PDEs and suggests promising directions for future research, such as extending the framework to multidimensional problems, adaptive neural architectures, and real-time modelling in engineering or biomedical settings.

## References

[1] Morton, K. W., & Mayers, D. F. (2005). *Numerical solution of partial differential equations: An introduction*. Cambridge University Press.

[2] Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359–366.

[3] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics, 378*, 686–707.

[4] Raissi, M., Yazdani, A., & Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science, 367*(6481), 1026–1030. https://doi.org/10.1126/science.aaw4741

[5] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics, 3*(6), 422–440.

[6] Amari, S.-I. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing, 5*(4–5), 185–196. https://doi.org/10.1016/0925-2312(93)90006-O

[7] Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv. https://arxiv.org/abs/1412.6980

[8] Wang, S., Teng, Y., & Perdikaris, P. (2021). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing, 43*(5), A3055–A3081.

[9] Wang, S., Yu, X., & Perdikaris, P. (2022). When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics, 449*, 110768. https://doi.org/10.1016/j.jcp.2021.110768

[10] McClenny, L., & Braga-Neto, U. (2021). Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint*. https://arxiv.org/abs/2009.04544

[11] Van der Meer, R., Oosterlee, C. W., & Borovykh, A. (2022). Optimally weighted loss functions for solving PDEs with neural networks. *Journal of Computational and Applied Mathematics, 405*, 113887. https://doi.org/10.1016/j.cam.2021.113887

[12] Jagtap, A. D., & Karniadakis, G. E. (2021). Extended physics-informed neural networks (XPINNs): A generalized space–time domain decomposition based deep learning framework for nonlinear partial differential equations. In *AAAI Spring Symposium: Machine Learning and Physical Sciences* (Vol. 10).

[13] Shukla, K., Jagtap, A. D., & Karniadakis, G. E. (2021). Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics, 447*, 110683. https://doi.org/10.1016/j.jcp.2021.110683

[14] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks* (Vol. 4, pp. 1942–1948). IEEE.

[15] Das, G., Pattnaik, P. K., & Padhy, S. K. (2014). Artificial neural network trained by particle swarm optimization for non-linear channel equalization. *Expert Systems with Applications, 41*(7), 3491–3496. https://doi.org/10.1016/j.eswa.2013.10.048

[16] Mirjalili, S. (2015). How effective is the grey wolf optimizer in training multi-layer perceptrons. *Applied Intelligence, 43*, 150–161. https://doi.org/10.1007/s10489-014-0645-7

[17] Chakraborty, A., & Kar, A. K. (2017). Swarm intelligence: A review of algorithms. In *Nature-inspired computing and optimization: Theory and applications* (pp. 475–494). Springer. https://doi.org/10.1007/978-3-319-50920-4_20

[18] Mousavirad, S. J., Schaefer, G., Mohammad, S., Jalali, J., & Korovin, I. (2020). A benchmark of recent population-based metaheuristic algorithms for multi-layer neural network training. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion* (pp. 1402–1408). ACM. https://doi.org/10.1145/3377929.3398090

[19] Kar, S., Banik, S. K., & Ray, D. S. (2003). Exact solutions of Fisher and Burgers equations with finite transport memory. *Journal of Physics A: Mathematical and General, 36*(11), 2771–2783. https://doi.org/10.1088/0305-4470/36/11/308

[20] Chavanis, P. H. (2008). Nonlinear mean field Fokker–Planck equations: Application to the chemotaxis of biological populations. *European Physical Journal B, 62*, 179–208. https://doi.org/10.1140/epjb/e2008-00079-9

[21] Sakthivel, R., Chun, C., & Lee, J. (2010). New travelling wave solutions of Burgers equation with finite transport memory. *Zeitschrift für Naturforschung A, 65*(8–9), 633–640.

[22] Jawad, A. J. A. M., Petković, M. D., & Biswas, A. (2010). Soliton solutions of Burgers equations and perturbed Burgers equation. *Applied Mathematics and Computation, 216*(11), 3370–3377. https://doi.org/10.1016/j.amc.2010.04.031

[23] Lyons, T. (2012). The 2-component dispersionless Burgers equation arising in the modelling of blood flow. *arXiv preprint*. https://arxiv.org/abs/1211.5546

[24] Jabłońska, M., Sitarz, R., & Kraslawski, A. (2013). Forecasting research trends using population dynamics model with Burgers' type interaction. In *Computer Aided Chemical Engineering* (Vol. 32, pp. 619–624). Elsevier.

[25] Lu, H. H. (2022). *Reduced-order models of transport phenomena* (Doctoral dissertation). Stanford University.

[26] Lu, H., & Tartakovsky, D. M. (2021). Dynamic mode decomposition for construction of reduced-order models of hyperbolic problems with shocks. *Journal of Machine Learning for Modeling and Computing, 2*(1).

[27] Rebai, A., Boukhris, L., Toujani, R., Gueddiche, A., Banna, F. A., Souissi, F., & Zaag, H. (2023). Unsupervised physics-informed neural network in reaction–diffusion biology models (ulcerative colitis and Crohn's disease cases): A preliminary study. *arXiv preprint*.

[28] Zhang, X., Li, Y., & Liu, Y. (2023). Deep ensemble physics-informed neural networks for uncertainty quantification in Burgers' equation. *Computers & Mathematics with Applications, 145*, 35–48.

[29] Liang, S., Jiang, S. W., Harlim, J., & Yang, H. (2024). Solving PDEs on unknown manifolds with machine learning. *Applied and Computational Harmonic Analysis, 71*, 101652.

[30] Brunton, S. L., & Kutz, J. N. (2024). Promising directions of machine learning for partial differential equations. *Nature Computational Science*, 1–12.