

BRIDGING THE GAP BETWEEN ARTIFICIAL NEURAL NETWORKS (ANNs) AND PARTIAL DIFFERENTIAL EQUATIONS (PDEs)

Nek Muhammad Katbar^{1✉}, Ghulam Fatima², Khuda Bux Amur³,
Emad A. A. Ismail⁴, Fuad A. Awwad⁴, Mosab Alqurashi⁴,
Abdul Hakeem¹

Received on October 28, 2024

Presented by Ch. Roumenin, Member of BAS, on November 26, 2024

Abstract

The current research presents novel and hybrid method to tackle complex nonlinear Partial Differential Equations (PDEs) by combining Radial Basis Function Neural Networks (RBFNNs) with a powerful hybrid optimization technique. RBFNN is used to generate data from the PDEs. A hybrid Chaotic Enriched Crayfish Optimization (HCE_CO) algorithm is utilized to optimize the RBFNN. The algorithm combines multiple optimization strategies, including Chimp Optimization Algorithms and Crayfish Optimization Algorithms. The RBFNN is trained to minimize the difference between its predictions and the actual numerical solutions by loss function. The HCE_CO algorithm adjusts the RBFNN's parameters, such as weights, biases, batch size, epochs, and number of hidden layers, to improve its accuracy. By comparing the RBFNN's predictions to numerical solutions, we found that our model achieved a Mean Absolute Error (MAE) of 7.464%, which is lower than that of previous methods.

Key words: PDEs systems, Radial Basis Function (RBF), Artificial Neural Networks, Chaotic Enriched Crayfish Optimization, optimization algorithm, evaluation metrics

Researchers Supporting Project No. RSPD2025R576, King Saud University, Riyadh, Saudi Arabia.

<https://doi.org/10.7546/CRABS.2025.04.08>

1. Introduction. Partial Differential Equations (PDEs) have gained significant attention in nonlinear science and engineering. These equations are used across various mathematical and scientific disciplines to model a broad spectrum of phenomena. Applications are numerous and include fields like chemical physics, hydrodynamics, solid-state physics, fluid mechanics, computational biology, image processing, quantum mechanics, chemical kinematics, fluid dynamics, ocean engineering, and hydrodynamics [1–3]. Diffusion equations are a distinguished class of nonlinear PDEs that provide information about the collective motion of small-scale particles inside materials [4, 5], which is caused by the random motions of individual particles. The important real-world scenarios that this modelling skill extends to include flood routing, oil pollution, biological systems, chemical reactions, water flow, and environmental contamination.

Even with their effectiveness, solving a variety of PDEs analytically is still very difficult. Many examples, especially in the engineering and applied sciences, provide situations in which analytical methods are either non-applicable or extremely complicated. Numerical and approximate methods have been essential tools in addressing the challenges of solving complex problems [6–8]. Rational function approximations, a core concept in applied mathematics, have found applications in fields like chemistry, engineering, and physics. Radial basis function networks (RBFN) are a powerful technique for function approximation and PDE resolution [9–11]. By adjusting weights and centres of radial basis functions, RBFN can accurately approximate functions and PDE solutions. This method offers efficient and cost-effective solutions for complex nonlinear PDEs. RBFNs are valuable tools in mathematical modelling and problem-solving, contributing to progress in various scientific and technical fields [12, 13].

Heuristic methodologies developed to solve the most complex problems within the past decades include Artificial Bee Colony, Simulated Annealing (SA), Genetic Algorithm (GA) [14], Evolution Approaches with Covariance Matrix Adaptation, Particle Swarm Optimization (PSO) [15], and VIM based fractional gradient approach. Different kinds of differential equations have been effectively solved by heuristics. OSMAN et al. [16] have discussed the PDE Fuzzy Method for Linear-Nonlinear Systems. Two techniques – fuzzy modified Laplace decomposition method (MLDM) and fuzzy Adomian decomposition technique are presented to address fuzzy fractional Navier–Stokes problems in a tube under fuzzy partial derivatives. CHAKRAVERTY and MALL [17] have introduced the regression-based weighted single-layer Chebyshev neural network (ChANN) model for nonlinear ordinary differential equation solutions. They presented the construction of a novel single-layer functional link neural network model, known as the ChANN, for managing normal differential equations utilizing regression-based weights. ZHANG et al. [18] have utilized the Bilinear Neural Network Method, to solve n-PDEs involving breathers, fractals, and exact periodic waves.

The authors in [19] have suggested a technique for determining the inverse mapping of an n-PDE and systems of n-PDEs. The authors utilized extraordinary learning devices for n-PDE computational solutions and bifurcation study. In the estimation of steady-state case of non-linear PDEs, they introduced an Extreme Learning Machines (ELMs) and collocation-based machine learning numerical approach. By utilizing the capacity of ELMs to approximate functions universally, the numerical scheme circumvents the need for computationally intensive and frequently unreliable training phase processes associated with other ANN models [20]. Our main goal is to introduce a pioneering framework using Radial Basis Function Neural Networks for solving nonlinear partial differential equations like the Allen–Cahn, Cahn–Hilliard.

2. Materials and methods. Nonlinear partial differential equations (PDEs) are fundamental to many fields, but finding exact solutions is often difficult or requires oversimplifying assumptions. As a result, numerical methods are commonly used to approximate solutions. While traditional numerical methods are useful, they can struggle with complex systems like those with dynamic microstructures or intricate fluid behaviour. To overcome these challenges, this research proposes a novel framework using radial basis function neural networks (RBFNN). RBFNN, a type of artificial neural network, provides a unique approach to solving PDEs. The proposed single-layer RBFNN method aims to solve Allen–Cahn and Cahn–Hilliard equations with accuracy and efficiency, offering a promising alternative to traditional numerical methods.

2.1. Problem formulation. In this study we consider two partial differential equations:

2.1.1. Allen–Cahn (AC) equation. In materials science, the Allen–Cahn equation serves as a valuable tool for investigating microstructure evolution. This mathematical model, classified as a diffusion nonlinear partial differential equation, simulates phase separation processes in materials. It characterizes the temporal and spatial progression of a scalar order parameter $u(x, t)$ offering insights into the dynamics of material transformations. AC equation can be expressed as

$$(1) \quad \frac{\partial u}{\partial t} = \varepsilon^2 \nabla^2 u - f(u),$$

where ∇^2 is the Laplacian operator, ε is a tiny positive constant interface width, and $u(x, t)$ is the order parameter.

2.1.2. Cahn–Hilliard (CH) equation. The Cahn–Hilliard equation is another model used to study phase separation. Unlike other diffusion models, it considers interfacial energy between different phases. This equation is particularly useful in materials science and engineering to understand phase separation in binary fluid systems. It allows researchers to simulate the complex behaviour of materials like polymers and alloys. The evolution in time and space of a scalar

order parameter $u(x, t)$ is designated by the Cahn–Hilliard equation

$$(2) \quad \frac{\partial u}{\partial t} = \varepsilon^2 \nabla^2 u - \nabla u - u^3 + u.$$

2.2. Radial basis function neural network (RBFNN). This section introduces a single-layer RBFNN to solve complex nonlinear PDEs. RBFNNs employ Gaussian basis functions to directly map input vectors into a hidden space, bypassing the need for weight connections. The network linearly transforms the hidden layer space into the output layer space, where parameters control the weighted summation of hidden layer outputs. The RBFNN is extensively used in a wide range of areas, including pattern classification and function approximation. In addition to their simple construction, they are highly regarded for their exceptional resistance to input noise, as well as their simplicity and ease of training. The RBFNN architecture is depicted in Fig. 1.

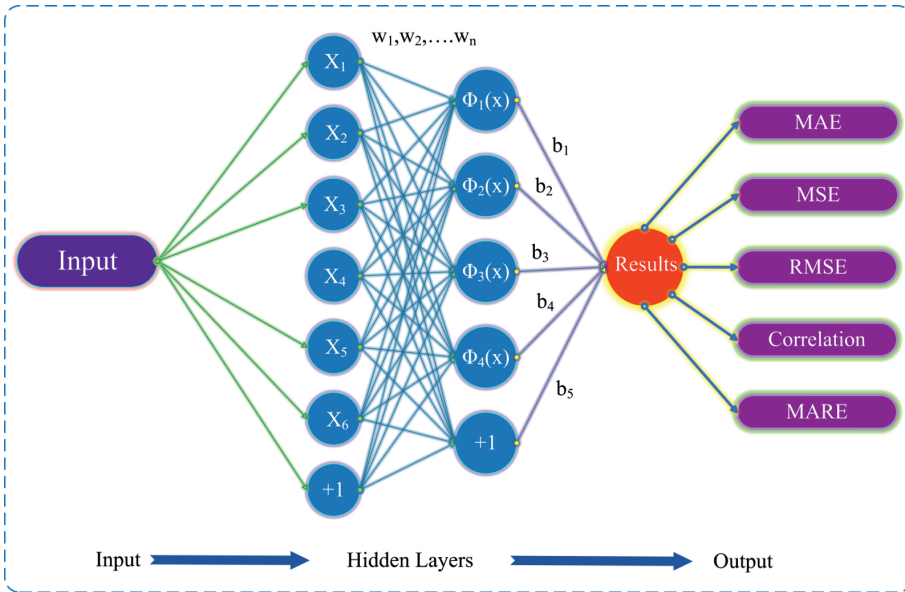


Fig. 1. Layered architecture of the RBFNN model

An input-output data set is assumed in order to design the RBF network's training issue

$$(3) \quad (x_i, y_i), \quad i = 1, 2, \dots, K,$$

where $x_i \in \mathbb{R}^N$, $y_i \in \mathbb{R}$, and K denotes the total number of samples. The purpose of the training procedure is to produce an approximation function $f: \mathbb{R}^N \rightarrow \mathbb{R}$, so that every input generates an equivalent target output. With reference to Fig. 1, the hidden units implement the RBF function φ_j , $j = 1, 2, \dots, L$. The thin plate spline function is utilized by $\varphi(v) = v^2 \log(v)$.

The activity of the l^{th} unit is denoted by $v_1(x(k))$, which is defined as

$$(4) \quad v_1(x(k)) = \|x(k) - c_l\|_2 = \sqrt{\sum_{i=1}^N (x_i(k) - c_{i,l})^2}, \quad k = 1, 2, \dots, K,$$

where the k input vector is represented by $x(k)$ and c_l represents the centre vector of the l^{th} RBF unit. The i^{th} input receives the output of the j^{th} hidden unit, which is

$$(5) \quad h_{j,i}(x_i) = \varphi_j(\|x_i - c_j\|_2).$$

The output of RBFN for each case of input, $i = 1, 2, \dots, K$ is defined as

$$(6) \quad \hat{y}_i(x_i) = \sum_{j=1}^L w_j \varphi_j(x_i, c_j).$$

The synaptic weights are determined by addressing a linear least squares problem when the RBF centres have been identified

$$(7) \quad w = (\varphi^T \varphi)^{-1} \varphi^T y.$$

Here y is a vector holding all of the target values and φ is a matrix holding the actions of each hidden layer to all input vectors. The hyper parameters like batch size, epoch and number of hidden layers are optimized using the HCE_CO. HCE_CO then uses this information to search the space of hyperparameters and find the configuration that maximizes the fitness function.

2.2.1. Hybrid Chaotic Enriched Crayfish optimization (HCE_CO) algorithm. The algorithm starts with the definition of the crayfish colony X to represent the features of swarm intelligence optimization. The position of the crayfish, denoted by X_i , suggests a solution known as the fitness value. The investigation and exploitation of COA are governed by temperature, a random variable that represents the local temperature, where the subject is situated. For COA, the competition stage is beginning when the temperature rises too much. Eq. (8) illustrates how the COA algorithm is initialized.

$$(8) \quad X = [X_1, X_2, \dots, X_N] = \begin{bmatrix} X_{1,1} & \dots & X_{1,j} & \dots & X_{1,\text{dim}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ X_{i,1} & \dots & X_{i,j} & \dots & X_{i,\text{dim}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ X_{N,1} & \dots & X_{N,j} & \dots & X_{N,\text{dim}} \end{bmatrix}_{N \times m},$$

where $X_{i,j}$ is the location of the crayfish i in the j dimension, N is the total population, dim is its dimension, and the value of $X_{i,j}$ is derived from Eq. (11)

$$(9) \quad X_{i,j} = lb_j + (ub_j - lb_j) \times \text{rand},$$

where rand is a random integer, lb_j denotes the lower bound of the j^{th} dimension, and ub_j denotes the upper bound. The crayfish is going to experience distinct stages due to temperature changes, which may affect their behaviour. Eq. (10) is the definition of temperature

$$(10) \quad \text{temp} = \text{rand} \times 15 + 20.$$

The mathematical model of the crayfish is given by

$$(11) \quad p = C_1 \times \left(\frac{1}{\sqrt{2} \times \pi \times \sigma} \times \exp \left(-\frac{(\text{temp} - \mu)^2}{2\sigma^2} \right) \right).$$

The presence of $\text{rand} \geq 0.5$ and $\text{temp} > 30$ indicates the interest of other crayfish in the cave. They will now engage in combat to take the cave. The crayfish uses Eq. (12) to compete for the cave.

$$(12) \quad X_{i,j}^{t+1} = X_{i,j}^t - X_{z,j}^t + X_{\text{shade}}.$$

The random individual of crayfish

$$(13) \quad z = \text{round}(\text{rand} \times (N - 1)) + 1.$$

Crayfish compete with one another at the Competition stage, and crayfish X_i modifies their position in response to another crayfish's position (X_z). Position adjustments increase the search area of COA and improve the algorithm's exploration capability.

The concept of a food establishment can be mathematically modelled using the driving and chasing the prey phase of the Chimp optimization algorithm (ChOA) which is a metheuristic algorithm.

$$(14) \quad X_{\text{food}} = X_G - a \cdot d,$$

$$(15) \quad d = |c \cdot X_{\text{food}} - m \cdot X_{i,j}^t|.$$

In this case, t denotes the number of iterations remaining, a , m , and c are the coefficient vectors, X_{food} is the position of the prey, and $X_{i,j}^t$ is the crayfish location vector. We compute the a , m , and c vectors by $a = 2 \cdot \text{fitness}_{\text{food}} \cdot \text{rand}_1 - \text{fitness}_{\text{food}}$, $c = 2 \cdot \text{rand}_2$, $m = \text{Chaotic value}$.

This involves the iterative reduction of $\text{fitness}_{\text{food}}$ from 2.5 to 0 in a non-linear fashion (during both phases). The random vectors in the interval $[0, 1]$ are rand_1 and rand_2 . Lastly, m is a chaotic vector that was computed using a variety of chaotic maps; it illustrates the impact of chimpanzees' sexual desire on the hunting process. The food size Q is defined as:

$$(16) \quad Q = C_3 \times \text{rand} \times \left(\frac{\text{fitness}_i}{\text{fitness}_{\text{food}}} \right),$$

where the value of C_3 , the food factor that represents the more food, is always equal to 3. The fitness value of the i^{th} crayfish is represented by fitness_i , and the fitness value of the food location is represented by $\text{fitness}_{\text{food}}$. The largest food serves as the basis for the crayfish's estimation of food size. $Q > (C_3 + 1)/2$ indicates an excessively large serving of food. Now, the crayfish may rip the food with its first claw foot. Mathematically

$$(17) \quad X_{\text{food}} = \exp\left(-\frac{1}{Q}\right) \times X_{\text{food}}.$$

There is a connection between the amount of food consumed and the food that crayfish obtain, thus the formula for foraging is as follows:

$$(18) \quad X_{i,j}^{t+1} = X_{i,j}^t + X_{\text{food}} \times p \times (\cos(2 \times \pi \times \text{rand}) - \sin(2 \times \pi \times \text{rand})).$$

The crayfish only needs to approach and consume the food when $Q \leq (C_3 + 1)/2$; the equation is as follows:

$$(19) \quad X_{i,j}^{t+1} = (X_{i,j}^t - X_{\text{food}}) \times p + p \times \text{rand} \times X_{i,j}^t.$$

The algorithm's exploitation and convergence skills enhance as COA gets closer to the optimal solution during the foraging phase.

3. Result and discussion. This section compares the outcomes of the suggested model with the methods currently in use. The Python platform is used to carry out the implementation. Since the RBFNN approach is meshfree, there is no need to create a mesh throughout the PDE's domain. The metric values obtained for the proposed model are compared with the existing techniques like Long Short-Term Memory, ANN [20], BNNM [18], and ChANN [17]. The comparison is performed in terms of MAE, MSE, RMSE, correlation and MARE. The comparison is shown in Table 1.

Table 1 presents a comparison of Proposed RBFNN method, LSTM, ANN [20], BNNM [18], and ChANN [17], using various evaluation metrics. The RBFNN technique demonstrates superior performance with the lowest values for MAE, MSE, and RMSE at 7.464, 89.793, and 10.422, respectively. Additionally, it achieves a

T a b l e 1

Comparison of performance metrics for proposed model and existing techniques

Techniques	MAE	MSE	RMSE	Correlation	MARE
Proposed RBFNN	7.464	89.793	10.422	0.962	0.242
LSTM	8.389	105.812	11.483	0.936	0.264
ANN [20]	7.852	99.58	10.887	0.885	0.816
BNNM [18]	8.15	111.364	11	0.944	0.96
ChANN [17]	10.114	145.368	12.081	0.868	1.114

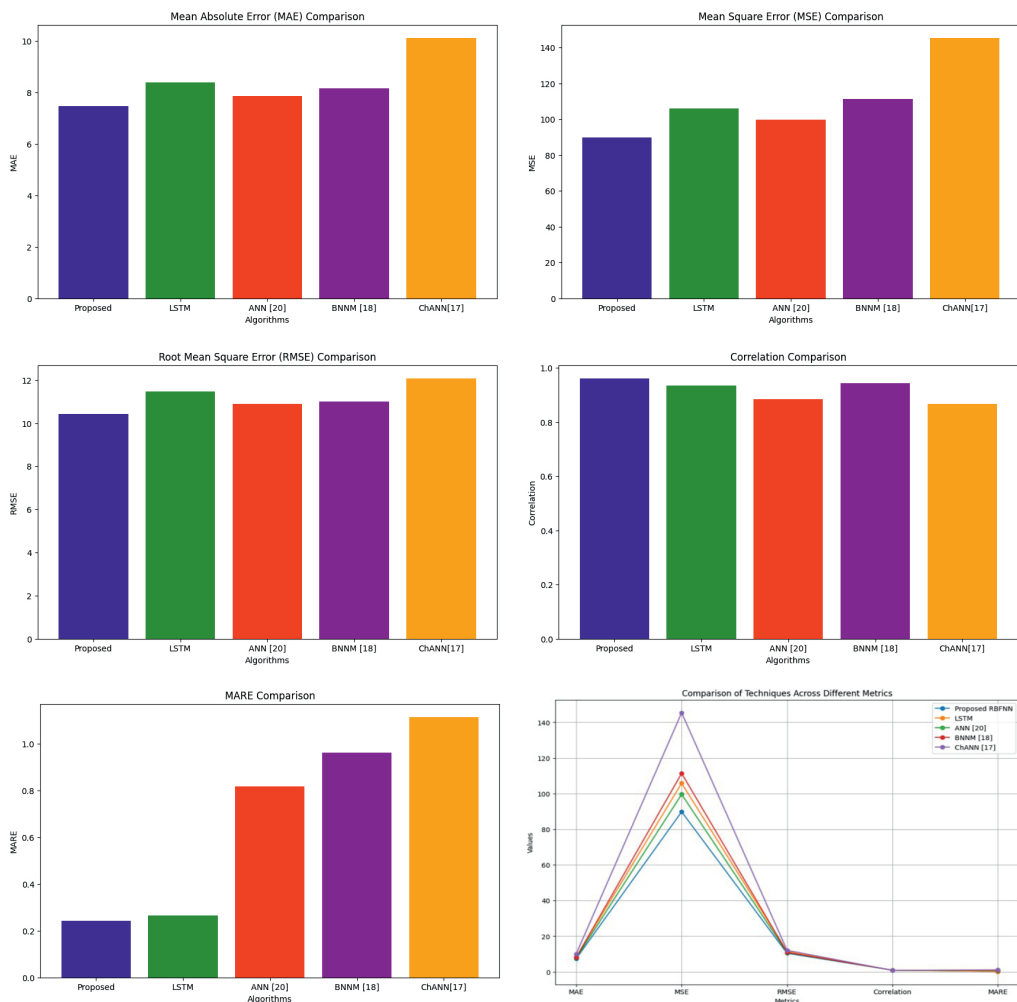


Fig. 2. Plots of metric performances

remarkably high correlation coefficient of 0.962, indicating a strong positive linear relationship between predicted and actual values. The MARE of RBFNN method is also the lowest among the listed techniques at 0.242, suggesting that the average % discrepancy between the expected and actual results is really small as seen in Fig. 2. The proposed RBFNN method emerges as the most effective based on the provided metrics, offering lower errors and a stronger correlation with actual values. Ultimately, the appropriate strategy to choose relies on the particular objectives and application requirements.

4. Conclusion. This study presents a novel approach for solving n-PDEs using RBFNNs coupled with a hybrid optimization technique. The RBFNN architecture, comprising an input layer, hidden layer with radial basis functions,

and output layer, effectively approximates solutions by converting the equation system into the network's output layer. The primary contribution of this study lies in the incorporation of the HCE_CO algorithm, which combines ChOA and COA, enhancing RBFNN training through weight and bias adjustments, as well as optimization of hyper parameters such as batch size, epochs, and hidden layers. The carefully crafted loss function quantifies the difference between the RBFNN predictions and actual solutions, ensuring precise model training. The effectiveness of the method was demonstrated through evaluation, achieving an MAE of 7.464%, which surpasses existing techniques in terms of accuracy. This underscores its potential to advance n-PDE solution methodologies, offering a robust and efficient framework that outperforms conventional approaches. The combination of hybrid optimization and adaptable RBFNNs presents a promising strategy for tackling complex nonlinear problems across various scientific and engineering problems.

REFERENCES

- [1] KUMAR S., A. KUMAR, Z. ODIBAT et al. (2020) A comparison study of two modified analytical approach for the solution of nonlinear fractional shallow water equations in fluid flow, *AIMS Mathematics*, **5**(4), 3035–3055.
- [2] SINGH A., P. INCARDONA, I. F. SBALZARINI (2021) A C++ expression system for partial differential equations enables generic simulations of biological hydrodynamics, *Eur. Phys. J. Soft Matter*, **44**(9), 117.
- [3] TANG X. Y., C. J. CUI, Z. F. LIANG, W. DING (2021) Novel soliton molecules and wave interactions for a $(3 + 1)$ -dimensional nonlinear evolution equation, *Nonlinear Dynamics*, **105**, 2549–2557.
- [4] SHAH J., F. ALI, N. KHAN et al. (2022) MHD flow of time-fractional Casson nanofluid using generalized Fourier and Fick's laws over an inclined channel with applications of gold nanoparticles, *Sci. Rep.*, **12**, 17364.
- [5] CARTWRIGHT M. C. (2021) Collective coordinates approach for travelling waves in stochastic partial differential equations, Doctoral dissertation.
- [6] LIU G., Z. R. LU, L. WANG, J. K. LIU (2021) A new semi-analytical technique for nonlinear systems based on response sensitivity analysis, *Nonlinear Dynamics*, **103**, 1529–1551.
- [7] KATBAR N. M., K. B. AMUR, I. AHMED et al. (2025) Implicit numerical approach for biological Schnakenberg model employing finite element method (FEM), *Commun. Math. Biol. Neurosci.*, <https://doi.org/10.28919/cmbn/8924>.
- [8] NAWAZ S., A. HUSSAIN, S. A. KHASKHELI (2024) On the modeling and computation of molecular descriptors of a novel next-generation wonder material γ -graphyne and zig-zag Gamma graphyne nanoribbon, *Ain Shams Eng. J.*, **15**(10), <https://doi.org/10.1016/j.asej.2024.102868>.
- [9] KATBAR N. M., A. HAKEEM, I. AHMED, G. ABBASS (2024) Radial Basis Function Neural Networks (RBFNN) for secure intrusion detection in the Internet of Drones (IoD), *C. R. Acad. Bulg. Sci.*, **77**(8), 1185–1192.
- [10] HUSSAIN S. M., R. MAHAT, N. M. KATBAR et al. (2023) Artificial neural network modeling of mixed convection viscoelastic hybrid nanofluid across a circular cylin-

der with radiation effect: Case study, *Case Studies in Thermal Engineering*, **50**(4), 103487.

- [11] RAMABATHIRAN A. A., P. RAMACHANDRAN (2021) SPINN: sparse, physics-based, and partially interpretable neural networks for PDEs, *J. Comput. Phys.*, **445**, 110600.
- [12] LIU Z., Q. XU (2019) A multiscale RBF collocation method for the numerical solution of partial differential equations, *Mathematics*, **7**(10), 964.
- [13] DAVIS E., G. YAO, E. JAVOR et al. (2022) A test of backward stochastic differential equations solver for solving semilinear parabolic differential equations in 1D and 2D, *PDE Appl. Math.*, **6**, 100457.
- [14] EL-SHORBAGY M. A., A. ELREFAEY (2020) Hybridization of grasshopper optimization algorithm with genetic algorithm for solving system of non-linear equations, *IEEE Access*, **8**, 220944–220961.
- [15] VAHIDI-MOGHADDAM A., A. RAJAEI, M. AYATI et al. (2020) Adaptive prescribed-time disturbance observer using nonsingular terminal sliding mode control: Extended Kalman filter and particle swarm optimization, *IET Control Theory Appl.*, **14**(19), 3301–3311.
- [16] OSMAN M., Y. XIA, O. A. OMER, A. HAMOUD (2022) On the fuzzy solution of linear-nonlinear partial differential equations, *Mathematics*, **10**(13), 2295.
- [17] CHAKRAVERTY S., S. MALL (2020) Single layer Chebyshev neural network model with regression-based weights for solving nonlinear ordinary differential equations, *Evolutionary Intelligence*, **13**(4), 687–694.
- [18] ZHANG R., S. BILIGE, T. CHAOLU (2021) Fractal solitons, arbitrary function solutions, exact periodic wave and breathers for a nonlinear partial differential equation by using bilinear neural network method, *J. Syst. Sci. Complexity*, **34**, 122–139.
- [19] DEWASURENDRA M., K. VAJRAVELU, J. ABDEEN et al. (2021) A method of directly defining the inverse mapping for a nonlinear partial differential equation and for systems of nonlinear partial differential equations, *Comput. Appl. Math.*, **40**(6), 1–16.
- [20] FABIANI G., F. CALABRÒ, L. RUSSO, C. SIETTOS (2021) Numerical solution and bifurcation analysis of nonlinear partial differential equations with extreme learning machines, *J. Sci. Comput.*, **89**, <https://doi.org/10.1007/s10915-021-01650-5>.

¹*School of Mathematics and Statistics, Central South University Changsha, Hunan 410083, China*

e-mails: nekmhammad@csu.edu.cn, hakeem@csu.edu.cn

²*School of Electrical and Control Engineering, Xian University of Science and Technology, Xian, Shanxi, China*

e-mail: fatimaiccg1@gmail.com

³*Department of Mathematics & Statistics, Quaid-e-Awam University of Engineering, Science & Technology, Nawabshah, Pakistan*

e-mail: kbamur@quest.edu.pk

⁴*Department of Quantitative Analysis, College of Business Administration, King Saud University, P.O. Box 71115, Riyadh 11587, Saudi Arabia*

e-mails: emadali@ksu.edu.sa, fawwad@ksu.edu.sa, malqurashi@ksu.edu.sa