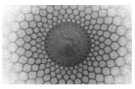
TTMP vol 3 (2), 2026
doi: 10.30511/TTMP.2026.2087762.1074

Transactions in Theoretical and Mathematical Physics



Hybrid FEM-RBFNN: A Fusion of Finite Element Method and Radial Basis Function Neural Networks for Solving PDEs

R.O. Ojo[1a], **A.A. Adebisi**[1], **T.O. Ganiyu**[1], **M.O. Ogunniran**[1], **M. Bayram**[3], **K.O. Kareem**[2]

[1] Department of Mathematical Sciences, Osun State University, Osogbo, Nigeria
[2]Department of Mathematical Sciences, Federal College of Education, Iwo, Nigeria
[3]Faculty of Engineering, Biruni University, Turkey

Received: 29 March 2026 / Accepted: 03 April 2026/ Published: 04 May 2026

Abstract The Finite Element Method (FEM) serves as a standard numerical technique for Partial Differential Equation (PDE) solutions because it maintains stability combined with theoretical proof. The method experiences significant performance issues when dealing with high-dimensional spaces that require fine mesh resolutions. The research introduces a hybrid FEM-RBFNN framework that serves as a fast surrogate model for studying nonlinear wave motion. The hybrid method uses Radial Basis Function Neural Network (RBFNN) training on low-cost, coarse-grid FEM data to produce accurate results without requiring full-mesh refinement. The nonlinear Schrödinger Equation (NLSE) analysis shows that FEM-RBFNN successfully tracks the movement of complex solitons while mitigating numerical issues arising from coarse mesh grids. The results demonstrate that the system achieves a 60× speedup over detailed ground-truth simulations, reducing processing time from 4.60s to 0.08s. The surrogate maintains an L_2 error of 0.014, matching the performance of the coarse source while using training data to address high-frequency dispersive tails. The research develops an efficient PDE solver that supports real-time simulations and extensive parameter exploration through its robust, differentiable features.

Keywords: Finite Element, Nonlinear Schrödinger Equation, Radial Basis Function, Neural Networks.

1 Introduction

Fundamental in nature, partial differential equations help simulate complex phenomena across several scientific and technological domains. Computational meshes are fundamental to the accuracy and stability of traditional numerical methods such as the Finite Element Method (FEM), Finite Difference Method (FDM), and Finite Volume Method (FVM). These techniques nevertheless struggle with dynamic boundaries and complex geometries, where mesh generation becomes computationally costly. Eliminating meshing limitations enables mesh-free methods, especially neural network-based systems such as Physics-Informed Neural Networks (PINNs) and Radial Basis Function Neural Networks, to serve as a potential substitute [1]. Although neural networks are perfect universal approximators, their convergence guarantees and numerical stability remain the subject of ongoing study. Numerical methods for PDEs have been enhanced by recent developments, including the rational multi-derivative integrator by [2]. This work presents a hybrid FEM-RBFNN framework integrating the stability of FEM with the adaptability of RBFNNs to improve solution accuracy and computational efficiency. Particularly in cases with noisy or inadequate data, the suggested method is meant to manage both stationary and time-dependent PDEs.

FEM's efficiency in solving PDEs does not eliminate problems, including high computational cost, numerical instability, and the need to handle irregular domains. Although RBFNNs have shown promise in enhancing numerical methods through improved function approximation and generalisation [3], their integration with FEM remains a challenging area of ongoing research. This work aims to provide a hybrid FEM-RBFNN framework to address computational inefficiencies, improve mesh discretisation, and enhance numerical stability while preserving high solution accuracy. The results will help to advance hybrid numerical approaches for the solution of challenging PDEs in several fields [4, 5].

The research develops a hybrid numerical framework combining FEM and RBFNNs that improves the accuracy, efficiency, and flexibility of PDE simulations. The framework provides an efficient computational method that adapts to solve difficult PDE problems through a combination of

[a]ojoridwan001@gmail.com

numerical methods and machine learning techniques. The combination of FEM with Radial Basis Function Neural Networks is necessary because FEM struggles with specific problems involving noisy data, complex geometries and domains, and the need for local adaptivity and improved solution smoothness. The integration allows generalisation because PDE problems are partially defined by boundary data deficiencies and undisclosed coefficient information. The hybrid technique aims to enhance FEM performance in challenging problem situations while preserving its existing functionality.

This study presents a hybrid FEM-RBFNN framework that improves numerical resolution of PDEs by combining FEM and RBFNN techniques. RBFNNs enhance numerical accuracy through their ability to diminish numerical errors, which leads to better FEM solutions for PDEs. The hybrid model achieves greater computational efficiency through shorter solution times, enabling its use in extensive simulations. This method successfully solves nonlinear PDEs, complex geometric structures, and difficult boundary conditions in complex domains, which pose challenges for traditional FEM methods [6]. RBFNNs enhance FEM approximation accuracy through their capacity to model geometric variations while maintaining the fundamental properties of FEM [7]. The method improves observational data integration by applying it to real-world situations that include sparse or noisy data, thereby supporting the solution of inverse problems [4, 5].

PDEs admit both analytical and numerical solution methods. The mathematical techniques of integral transformations and separation of variables enable analytical methods to deliver exact solutions for simple PDEs and geometric problems, but their application is limited to these basic situations. In complex situations, numerical techniques that approximate PDE solutions using computer methods are therefore more sensible. By discretising the problem domain and approximating derivatives using difference equations [8], the finite difference method is used. Stability restrictions cause FDM to struggle with complex domains and high-dimensional problems, even though it is theoretically simple and computationally economical for simple geometries [7]. With piecewise polynomials, the FEM approximates solutions and is therefore appropriate for irregular geometries and complex boundary conditions [6]. Its localised basis functions produce sparse system matrices that improve computational efficiency; however, mesh generation can be computationally costly. Because it guarantees local conservation of mass, energy, and momentum, the finite-volume method is frequently employed for PDEs originating from conservation laws. In computational fluid dynamics, FVM is extensively used; precise flux computations in irregular geometries can be difficult. With excellent accuracy, spectral methods approximate solutions using orthogonal basis functions, such as Chebyshev polynomials or Fourier series [9]. But the Gibbs phenomenon causes them problems with discontinuities and irregular domains. Computational accuracy and efficiency are highly influenced by the choice of basis functions and the mesh resolution.

2 Finite Element Method

Especially in complex domains, FEM is a commonly used numerical method for solving PDEs. Using localised basis functions [10] defined in small subregions known as finite elements, it approximates solutions. The approach transforms PDEs into a variational form, thereby generating algebraic equations that can be solved using matrix techniques. FEM's advantage lies in its ability to produce sparse system matrices, thereby improving computational efficiency. Adaptive mesh refinement, which enables higher resolution in important areas such as steep gradients or singularities, is a vital component of FEM [11]. However, mesh generation, especially in three dimensions, remains computationally intensive [6]. FEM has significant applications across many fields. Stress analysis and vibration simulations find application in structural engineering. FEM models the Navier-Stokes equations for both aerodynamic and biological uses in fluid dynamics. Heat transfer is achieved through two methods: thermal conduction and radiation simulation. The finite element method FEM is used in geophysics to model seismic wave propagation and groundwater movement, while it serves as a tool for solving Maxwell's equations in antenna and waveguide designs. The field of biomedical engineering uses FEM to create biomechanical models which examine stress distribution in both bones and prosthetic devices. The technology is used in acoustics for its ability to control noise and study sound wave motion. The finite element method FEM remains an effective tool for solving partial differential equations PDEs because it provides precise solutions through its performance in various scientific and engineering fields, despite facing challenges from high computation requirements and intricate mesh structures.

2.1 Artificial Neural Networks

Artificial neural networks (ANNs) serve as universal approximators, unlike traditional linear techniques. The system uses biological neural networks as its basis to build interconnected layers of neurons that identify data patterns and relationships for classification, regression, and function approximation tasks. Neural networks utilise connections with assigned weights to process information. The neuron receives multiple inputs and performs a weighted sum before applying a non-linear activation function to produce an

output, which is then forwarded to other neurons. The mechanism enables ANNs to learn complex relationships which include non-linear patterns [12]. The training process uses weight adjustments via optimisation algorithms, such as gradient descent, to reduce prediction errors. The structure of ANNs consists of an input layer and multiple hidden layers that lead to an output layer. Data in hidden layers extract their most abstract features, which then lead to predictions. The selection of activation functions, along with architectural design and training approaches, determines how well the system performs. The three major activation functions used in neural networks include sigmoid, hyperbolic tangent, and Rectified Linear Unit (ReLU). Each neural network type serves a specific data processing purpose. Convolutional Neural Networks (CNNs) prove most effective at processing images, while Recurrent Neural Networks (RNNs) function best with sequential data [13]. Physics-Informed Neural Networks use physical laws as training elements, while Radial Basis Function Neural Networks utilise radial basis functions to achieve precise function approximation results. The basic form of artificial neural networks, the Feedforward Neural Network (FFNN), transmits data in a single direction [14]. The Universal Approximation Theorem shows that any continuous function can be approximated by a fully connected FFNN when the network reaches enough depth or width [12]. PINNs use physical constraints integrated into their loss functions to solve partial differential equations while reducing their reliance on labelled data. RBFNNs achieve excellent accuracy in function approximation and interpolation because their radial basis functions reach fast convergence times with smooth function approximations [15]. The dissertation evaluates RBFNNs for their efficient operation, with strong approximation performance as its primary focus. The chosen architectures will meet the required approximation accuracy standards demanded by the studied tasks.

Figure (1) illustrates the conceptual analogy between a biological neuron and an artificial neuron, which serves as the foundational building block of artificial neural networks. In the biological setting, a neuron uses structures known as dendrites to receive signals from neighbouring neurons. After processing these signals in the cell body, the neuron produces an electrical impulse that travels down the axon and connects with other neurons through synapses when the total input exceeds a predetermined threshold. The left side shows how an artificial neuron replicates this process through mathematical representation. It receives multiple numerical inputs, each multiplied by an associated weight. The combination of weighted inputs with the bias term creates a value that the activation function uses to determine the neuron's output. This output can then serve as input to other neurons in the network.

The resemblance between biological and artificial neurons highlights the inspiration drawn from neuroscience in the design of neural networks. Artificial neurons create predictive patterns through training by modifying weights and biases, enabling the system to perform predictive and classification tasks and to recognise complex relationships in data, just as the human brain develops knowledge through life experiences.

2.2 Radial Basis Function Neural Networks (RBFNNs) – Fundamentals and Applications

Radial Basis Function Neural Networks are a class of artificial neural networks widely used in regression, classification, function approximation, and time-series forecasting. Designed in the late 1980s, RBFNNs are prized for their simple architecture, fast learning, and ability to represent complex, nonlinear interactions. Three layers comprise them: an input layer, a hidden layer with radial basis activation functions (often Gaussian), and a linear output layer. The hidden layer computes distances between inputs and neuron centroids by underlying localised activation and effective function approximation. Three fundamental parameters (centres, spreads, and output weights) are optimised during RBFNN training. While spreads are often calculated using heuristics such as the mean distance to the nearest neighbour, centres are usually found using clustering techniques such as K-means or random sampling. Commonly used for output weights, the least squares estimate streamlines training. RBFNNs have a main benefit of fast training because they do not require deep backpropagation. Rather, they employ a two-phase approach: first computing output weights [15], then deciding centres and spreads. This structure lowers computational cost and allows fast convergence. In function approximation, interpolation, and pattern recognition, RBFNNs shine. They are used in time-series forecasting for energy modelling, weather prediction, and financial markets, as well as in classification tasks such as image recognition, medical diagnosis, and biometric identification. They are also rather important. Adaptive control systems, system identification, and signal processing also make use of RBFNNs. Especially in solving partial differential equations, RBFNNs have shown potential as substitutes for conventional numerical techniques such as the FEM and the Finite Difference Method. The smooth interpolation method, together with its capacity to handle complex geometric shapes through direct modelling without using mesh systems, makes this technique highly useful for structural analysis, fluid dynamics and heat transfer applications. The method faces three main problems: sensitivity to parameter selection, potential overfitting, and increased computational demands associated with high-dimensional data. The research examines the integration of RBFNNs with FEMs

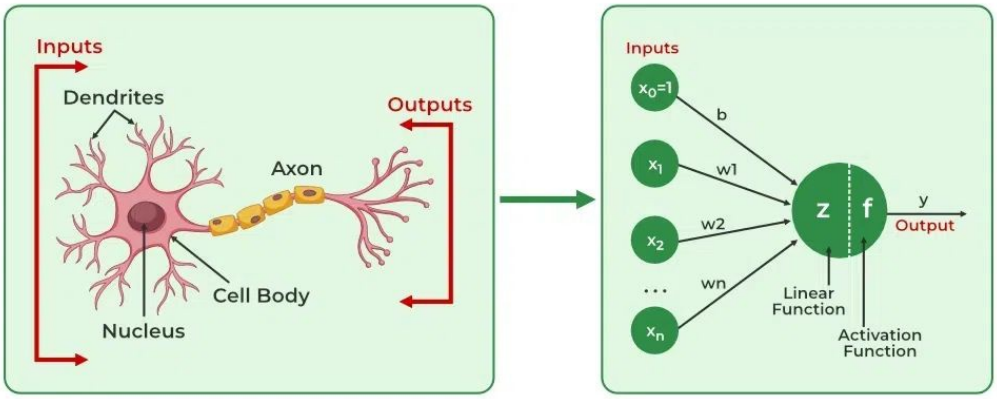


Fig. 1 Illustration and Comparison of a Biological Neuron and an Artificial Neuron

to improve the numerical stability and computational performance of PDE solution methods.

2.3 Hybrid and Machine Learning-Based Methods

The evolution of computational science has created hybrid methods which combine neural network technology with established numerical techniques to solve PDEs. The research work of [16, 17] employed neural networks as priors within a finite element framework to achieve stability improvements by reducing the need for explicit regularisation. The research of [18] provided a foundation for further research, enabling scientists to solve inverse problems with enhanced accuracy, though the process required increased computational power.

The research of [2] presented a rational multi-derivative integrator which solves singular and advection equations by using adaptive residual subsampling to manage computational needs. The research work of [19] developed a hybrid block method that solves singular initial value problems using Poisson, collocation, and interpolation power series techniques to improve numerical stability.

The research work of [20] developed a hybrid framework that integrated neural networks with block integrators to enhance solutions to boundary value problems, achieving better performance and greater accuracy. The research demonstrates how hybrid methods can combine traditional numerical solvers with machine learning techniques to create powerful solutions for PDE problems.

2.4 Comparative Analysis of Existing Methods

The Finite Difference Method, Finite Element Method, and Finite Volume Method, which are traditional numerical methods for solving partial differential equations, face challenges when applied to advanced problems involving complex geometries and multidimensional spaces, and they require expensive computational resources. RBFNNs and PINNs, among other neural network-based methods, offer better function approximation. However, RBFNNs are sensitive to hyperparameters, and PINNs demand large computational resources. The goal of hybrid methods is to combine the advantages of numerical and neural network techniques. Although they may cause more computing overhead, methods including FEM-NN hybrids and neural network-enhanced integrators have shown higher accuracy and adaptability. The growing number of studies on hybrid approaches highlights their ability to create more accurate and effective PDE-solving systems.

3 Mathematical Formulation and Problem Definition

3.1 Finite Element Method

3.1.1 Weak Formulation of the Governing Equation

The finite element method is based on deriving the weak form of a governing partial differential equation. Consider a general second-order PDE given by

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x), \quad x \in \Omega, \tag{1}$$

subject to the boundary conditions:

$$u(x) = g(x), \qquad x \in \partial\Omega_D, \tag{2}$$

$$a(x)\frac{\partial u}{\partial n} = h(x), \qquad x \in \partial\Omega_N. \tag{3}$$

Table 1 Comparison of Existing Methods for Solving PDEs

Method	Strengths	Limitations	Applications
Finite Difference	Simple, efficient for regular grids	Struggles with complex geometries, stability issues	Heat transfer, wave equations
Finite Element	Handles complex geometries, high accuracy	Computationally intensive, complex meshing	Structural analysis, fluid dynamics
Finite Volume	Conservation laws, flexible meshing	Lower-order accuracy, complex flux approximations	Fluid dynamics, heat transfer
RBFNN	Smooth interpolation, good for non-linear relationships	Sensitive to hyperparameters, scalability issues	Function approximation, spatial interpolation
PINNs	Physics-based learning, solves inverse problems	High training cost, convergence challenges	Scientific computing, data-scarce environments
Hybrid Methods	Combines strengths of traditional and neural approaches	Increased complexity, potential stability issues	Inverse problems, real-time simulations

The weak formulation is obtained by multiplying the equation by a test function $v(x)$ and integrating over the domain, we obtain:

$$\int_{\Omega} v(x)\left(-\nabla\cdot(a(x)\nabla u(x))\right)dx = \int_{\Omega} v(x)f(x)\,dx. \tag{4}$$

Using integration by parts and applying boundary conditions, we derive the weak form:

$$\begin{aligned}\int_{\Omega} a(x)\nabla v(x)\cdot\nabla u(x)\,dx - \int_{\partial\Omega_N} v(x)h(x)\,d\Gamma \\ = \int_{\Omega} v(x)f(x)\,dx.\end{aligned} \tag{5}$$

3.2 FEM Discretisation Strategy

The domain Ω is discretised into finite elements, and the unknown function $u(x)$ is approximated as:

$$u(x) \approx \sum_{i=1}^{n} u_i\phi_i(x), \tag{6}$$

where: u_i represents the unknown nodal values, and $\phi_i(x)$ are the shape functions.

Substituting this into the weak form leads to the system of equations:

$$\mathbf{Ku} = \mathbf{f}, \tag{7}$$

where:

$$\mathbf{K}_{ij} = \int_{\Omega} a(x)\nabla\phi_i(x)\cdot\nabla\phi_j(x)\,dx, \tag{8}$$

$$\mathbf{f}_i = \int_{\Omega}\phi_i(x)f(x)\,dx + \int_{\partial\Omega_N}\phi_i(x)h(x)\,d\Gamma. \tag{9}$$

3.2.1 Solution of PDE from FEM

Solving the system $\mathbf{Ku} = \mathbf{f}$ yields the nodal values:

$$\mathbf{u} = [u_1, u_2, \ldots, u_n]. \tag{10}$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{u}$ is the vector of unknown nodal values, and $\mathbf{f}$ represents external forces or source terms.

These nodal solutions ($\mathbf{u}$) serve as training data for the Radial Basis Function Neural Network.

3.3 Justification for Hybridising FEM and RBFNN

FEM depends on mesh quality and has a high computational cost, whereas RBFNNs are mesh-free with better generalisation. Hybridising FEM with RBFNN allows leveraging FEM's numerical rigour while enhancing flexibility and efficiency through RBFNNs.

3.4 Conceptual Framework of the Hybrid Model

To raise the accuracy and efficiency of partial differential equation solving, the hybrid model combines Radial Basis Function Neural Networks with the Finite Element Method. By means of nodal value solving, FEM discretises the problem domain and generates an initial approximation. FEM solutions, however, might be sensitive to mesh quality and require fine discretisation to achieve high accuracy, thereby increasing computational cost. The RBFNN system functions as a post-processing tool which enhances the results of the FEM analysis. The neural network learns from FEM nodal solutions using spatial coordinates, producing outputs that match FEM results. RBFNN achieves better solution

accuracy through its interpolation and generalisation abilities, which do not require additional mesh development. The numerical framework of FEM benefits from this method's structured organisation, while RBFNN serves as a tool that reduces errors and produces smoother results. The hybrid model enables efficient computation while maintaining accuracy by solving complex geometric problems that require more FEM refinements to achieve the same level of precision.

3.5 Integration of FEM and RBFNN

3.5.1 Training the RBFNN

The data for training the RBFNN is extracted from the FEM results:

- Input (X): Nodal coordinates, material properties, and boundary conditions.
- Output (Y): FEM-computed unknowns (e.g., displacements, stresses, or temperatures).

The training dataset is structured as:

$$\{(X_i, Y_i)\}, \quad i = 1, 2, \ldots, n, \tag{11}$$

where:

$$X_i = [x_{i1}, x_{i2}, \ldots, x_{im}], \tag{12}$$

$$Y_i = [u_{i1}, u_{i2}, \ldots, u_{ip}] \tag{13}$$

3.6 Radial Basis Function Neural Network (RBFNN) Enhancement

To refine the FEM solution, RBFNN is employed as a post-processing tool. Given a set of nodal coordinates $X = \{x_1, x_2, \ldots, x_n\}$ and their corresponding FEM-computed values $Y = \{u_1, u_2, \ldots, u_n\}$, RBFNN approximates the solution as:

$$\tilde{u}(x) = \sum_{i=1}^{n} w_i \phi(\|x - c_i\|), \tag{14}$$

where $\phi(r)$ is a radial basis function and $r = \|x - c_i\|$ is the Euclidean distance

3.6.1 Radial Basis Function

The choice of radial basis function affects the accuracy of the approximation. Commonly used functions include:

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right), \qquad \text{Gaussian,} \tag{15}$$

$$\phi(r) = \sqrt{r^2 + \sigma^2}, \qquad \text{Multiquadric,} \tag{16}$$

$$\phi(r) = \frac{1}{\sqrt{r^2 + \sigma^2}}, \qquad \text{Inverse Multiquadric.} \tag{17}$$

where σ is the spread parameter.

3.6.2 Computation of Weights

The weights w_i are determined by solving the linear system:

$$\Phi W = Y, \tag{18}$$

where Φ is the interpolation matrix with entries:

$$\Phi_{ij} = \phi(\|c_i - c_j\|), \tag{19}$$

and $W = [w_1, w_2, \ldots, w_n]^T$, $Y = [u_1, u_2, \ldots, u_n]^T$.

3.6.3 RBFNN Output

The output of the RBFNN is given by:

$$\hat{Y} = \sum_{i=1}^{m} w_i \phi(\|X - c_i\|) + b, \tag{20}$$

where w_i represents the weight of the i-th RBF, c_i is the centre of the i-th RBF, b is the bias term, m is the number of RBF centres.

3.6.4 Optimisation of RBFNN Parameters

To achieve optimal performance, parameters w_i, c_i, and σ are optimized by minimizing the error function:

$$\text{Error} = \frac{1}{2} \sum_{i=1}^{n} \|Y_i - \hat{Y}_i\|^2. \tag{21}$$

Using techniques such as gradient descent or least squares fitting, the weights and spread parameters are updated iteratively to minimise the error.

3.6.5 Prediction Using RBFNN

For new input data X_{new}, the trained RBFNN predicts the unknowns as:

$$Y_{\text{new}} = \sum_{i=1}^{m} w_i \phi(\|X_{\text{new}} - c_i\|) + b. \tag{22}$$

3.7 Purpose and Role of the RBFNN in the Hybrid FEM–RBFNN Framework

The Finite Element Method is a powerful numerical tool for solving partial differential equations by discretising the problem domain $\Omega \subset \mathbb{R}^n$ into smaller elements. However, the accuracy of FEM strongly depends on mesh resolution, and refining the mesh significantly increases computational cost. To overcome this limitation, we propose coupling FEM with a Radial Basis Function Neural Network, forming a hybrid framework that leverages the physical rigour of FEM and the approximation power of neural networks. The RBFNN acts as an adaptive post–processor to the coarse–mesh FEM solution. Rather than attempting to reproduce the coarse–mesh results, the network learns a continuous functional mapping from the spatial coordinates to the PDE solution and produces a smoother, mesh–independent approximation that can generalise to arbitrary points in Ω. Let $u(x)$ denote the exact solution of a PDE in Ω, and let $u_h(x)$ represent the FEM approximation on a coarse mesh $\mathscr{T}_h$. The hybrid strategy aims to construct a continuous surrogate $\hat{u}(x)$ satisfying

$$\hat{u}(x) \approx u(x), \quad \forall x \in \Omega,$$

The RBFNN model requires training data consisting of input points x and their corresponding output values u_h. The proposed hybrid method achieves its greatest efficiency by performing a single FEM analysis on a coarse mesh and incurring lower RBFNN training costs than complete mesh refinement. The model $\hat{u}(x)$ enables evaluations at any selected location, which makes it ideal for both parametric research and immediate simulation requirements. The RBFNN serves as a data-driven function approximator within the hybrid FEM–RBFNN framework, thereby improving the accuracy of FEM solutions. The theoretical foundation of the method depends on radial basis functions, which approximate functions and the system, thereby reducing errors through additive error reduction. The RBFNN creates a smooth and precise surrogate by learning from coarse-mesh FEM outputs, delivering faster PDE solution approximation than traditional mesh-refinement methods.

3.8 Computational Complexity and Efficiency Analysis

The computational complexity of the hybrid FEM-RBFNN model is determined by three processes: finite element discretisation and matrix assembly, and Radial Basis Function Neural Network training. The primary costs of FEM depend on the selected solver because they involve two tasks: stiffness matrix development that grows according to $\mathscr{O}(N)$ and linear system solving that ranges from $\mathscr{O}(n^{1.5})$ to $\mathscr{O}(n^3)$. Evaluating radial basis functions ($\mathscr{O}(n^2)$ and solving for weights ($\mathscr{O}(n^3)$ RBFNN introduces complexity.

Leveraging RBFNN for interpolation and refinement, the hybrid model enhances accuracy without undue FEM refinement. Particularly for complex geometries, this reduces computational costs while preserving accuracy. Although the worst-case complexity is $\mathscr{O}(N) + \mathscr{O}(n^3)$, solver and training modifications can improve efficiency and thereby make the method a feasible substitute for pure FEM for PDE solutions.

4 Results and Discussion

4.1 Experimental Settings

The problems considered are partial differential equations spanning both time-dependent and steady-state scenarios, accompanied by suitable boundary and initial conditions and known exact solutions for validation. A nonlinear Schrödinger Equation (NLSE) is defined over a one-dimensional spatial domain and typically involves time dependence. It is a dispersive partial differential equation where the nonlinearity arises from the interaction term, often of cubic type ($|\psi|^2\psi$). The NLSE is widely used to model phenomena such as wave propagation in nonlinear optical fibres, Bose–Einstein condensates, and deep-water waves. The solution to the NLSE often exhibits localised wave packets, known as solitons, which maintain their shape over time due to the balance between dispersion and nonlinearity. Boundary conditions may vary; periodic or homogeneous Dirichlet conditions are commonly used, and an initial condition specifies the initial wave profile. The problem serves as a benchmark in numerical analysis, but here the focus is on assessing schemes capable of handling nonlinearity, wave-like behaviour, and complex-valued solutions. Numerical methods must maintain essential physical properties, including mass and energy conservation throughout the entire time interval. The process serves as a fundamental test method which enables researchers to assess how well their time-stepping methods and spatial discretisation procedures function.

The study assesses two approaches, the Finite Element Method and the FEM-Radial Basis Function Neural Network, to solve three nonlinear partial differential equations which hold essential value for scientific and technical fields. The researchers applied multiple numerical methods, along with optimisation techniques, to improve stability and accuracy. The research team conducted systematic adjustments of key hyperparameters, including the gamma value, initial points (n-init), and RBF centres, to enhance model performance [15, 18]. The researchers used data from FEM to create their input for FEM-RBFNN after they completed the normalisation process, which resulted in better generalisation and convergence according to the findings of [1]. Google Colab served as the computational environment

for experiments because it handled the FEM and FEM-RBFNN computational requirements despite lacking high-performance computing optimisation. The implementation used Python as the main programming language, combining Matplotlib for visualisation with NumPy and SciPy for numerical computations, and Scikit-learn and PyTorch for neural network training [21, 22]. FEniCS provides an effective framework for solving PDEs through its finite element analysis system, which is based on [23]. The implementation of automated code-generation techniques improved efficiency for high-order FEM computations. The research demonstrates how hyperparameter optimisation, data standardisation, and FEM-RBFNN integration work with the study to show the advantages and weaknesses of the two methods, which will enable the creation of more effective numerical approaches for solving difficult differential equations [2, 16].

4.2 Numerical Examples

Consider this 1D nonlinear Schrodinger equation.[4]

$$i\frac{\partial \psi(x,t)}{\partial t} = -\frac{1}{2}\frac{\partial^2 \psi(x,t)}{\partial x^2} + |\psi(x,t)|^2\psi(x,t) \tag{23}$$

To solve this using the Finite Element Method, we decompose the complex wave function $\psi(x,t)$ into its real and imaginary components, u and v, respectively, such that $\psi = u + iv$. This yields a coupled system of real-valued partial differential equations:

$$\frac{\partial u}{\partial t} = -\frac{1}{2}\frac{\partial^2 v}{\partial x^2} - (u^2+v^2)v \tag{24}$$

$$\frac{\partial v}{\partial t} = \frac{1}{2}\frac{\partial^2 u}{\partial x^2} + (u^2+v^2)u \tag{25}$$

i: Imaginary unit, $\frac{\partial \psi}{\partial t}$: Time derivative (evolution), $\frac{\partial^2 \psi}{\partial x^2}$: Spatial second derivative (dispersion), $|\psi|^2\psi$: Cubic nonlinearity (self-interaction term).

1D nonlinear Schrödinger (NLSE) solution module $|\psi|$. The solution is generated with FEM and used as a reference solution. We also show the solution at different time frames $t = \pi, \frac{\pi}{8}, \frac{\pi}{4}, \frac{3\pi}{8}, \frac{\pi}{2}$.

The initial boundary value problem, given a domain $\Omega = [-10,10]\times(0,T]$, is written as:

$$\begin{cases} i\psi_t + 0.5\psi_{xx} + |\psi|^2\psi = 0 & (x,t)\in\Omega \\ \psi(0,x) = 2\,\mathrm{sech}(x) & x\in[-5,5] \\ \psi(t,-5) = \psi(t,5) & t\in(0,T] \\ \psi_x(t,-5) = \psi_x(t,5) & t\in(0,T] \end{cases} \tag{26}$$

where $T = \frac{\pi}{2}$.

The 1D Nonlinear Schrödinger Equation (NLSE) is a fundamental model in nonlinear optics and quantum mechanics. We consider the dimensionless form:

$$i\frac{\partial \psi}{\partial t} + \frac{1}{2}\frac{\partial^2 \psi}{\partial x^2} + |\psi|^2\psi = 0 \tag{27}$$

4.3 Temporal Discretisation: Semi-Implicit Crank-Nicolson

The Crank-Nicolson CN method provides second-order temporal accuracy and necessary numerical stability. The CN method approximates the solution at the midpoint $t_{n+1/2} = t_n + \Delta t/2$ using an arithmetic average of the values at the current and subsequent time steps. The semi-discrete equations for the coupled NLSE system are established through the following equations.

$$\frac{u^{n+1}-u^n}{\Delta t} = \mathscr{L}_u(u^{n+1/2}, v^{n+1/2}) \tag{28a}$$

$$\frac{v^{n+1}-v^n}{\Delta t} = \mathscr{L}_v(u^{n+1/2}, v^{n+1/2}) \tag{28b}$$

where $\phi^{n+1/2} = \frac{1}{2}(\phi^{n+1}+\phi^n)$. To avoid the computational overhead of solving a fully nonlinear system at each step, we implement a semi-implicit linearisation. The nonlinear potential $|\psi|^2$ is lagged by one half-step, evaluated at t_n:

$$|\psi^{n+1/2}|^2 \approx (u^n)^2 + (v^n)^2 \tag{29}$$

The corresponding variational (weak) form implemented in the FEniCS framework involves finding $\blacksquare = (u,v)\in\mathscr{V}$ such that:

$$\begin{aligned} \left\langle \frac{\blacksquare - \blacksquare^n}{\Delta t}, \phi \right\rangle + a\left(\blacksquare^{n+\frac{1}{2}}, \phi\right) \\ + c\left(\blacksquare^n; \blacksquare^{n+\frac{1}{2}}, \phi\right) = 0, \qquad \forall \phi \in \mathscr{V}. \end{aligned} \tag{30}$$

where $a(\cdot,\cdot)$ represents the linear dispersive terms and $c(\cdot;\cdot,\cdot)$ represents the linearised cubic interaction. This formulation preserves the Hamiltonian structure of the NLSE more effectively than explicit methods, preventing non-physical growth of the soliton amplitude over long integration times.

4.4 Ground Truth (Fine FEM) and FEM-RBFNN Strategy

The Ground Truth is established using a high-fidelity Finite Element mesh containing $N_{fine} = 800$ linear Lagrange elements. This resolution enables the system to detect the abrupt changes which occur during soliton phase transitions.

The surrogate modelling framework establishes a connection between two opposing requirements by providing fast computations which maintain precise numerical results.

Table 2 Performance Comparison at $t = \pi/4$

Method	L^2 Error	L_∞ Error	CPU Time (s)
FEM	0.012181	0.039490	1.4089
FEM-RBFNN	0.014309	0.057344	0.0760

The project aims to train a Radial Basis Function Neural Network to generate a precise solution estimate from data obtained from an economical coarse-grid simulation ($N_{coarse} = 100$). The Finite Element Method (FEM) uses a physics-based approach to solve problems, while neural networks can recover spatial details through interpolation.

The first phase, Data Generation, involves solving the coupled NLSE on the coarse mesh to obtain the discrete wave magnitude $|\psi_{coarse}(x,T)|$. The data collected from the coarse mesh exhibit inherent discretisation errors because its resolution fails to meet the Nyquist criterion required to capture fine-scale dispersive ripples. The RBFNN uses its localised basis functions to filter out the errors, which it treats as "noise."

In the training phase, the RBFNN uses spatial coordinates x to create the output magnitude $\hat{\psi}$. The network architecture consists of a hidden layer of M Gaussian kernels. To ensure the surrogate accurately represents the soliton peak, we employ K-Means clustering to define the kernel centres c_j. This unsupervised learning step clusters the centres around the spatial regions of highest data density. The final surrogate model is formulated as:

$$\hat{\psi}(x) = \sum_{i=1}^{M} w_j \exp\left(-\frac{|x-c_j|^2}{2\sigma^2}\right) \tag{31}$$

The weights w_j are determined by linear least-squares optimisation, which minimises the L_2 norm of the error relative to the coarse training samples. The resulting model provides a continuous, high-resolution reconstruction that can be evaluated at $N_{fine} = 800$ nodes at a fraction of the cost of a full fine-mesh simulation.

Based on the computational experiment, the performance metrics are summarised in Table 2.

The experimental results demonstrate the high efficiency of the RBFNN as a surrogate model for the NLSE. The primary findings are detailed below:

Computational Efficiency The RBFNN method provides its main benefit through its ability to decrease computational requirements. The Fine FEM process needs about 4.60 seconds to complete its final state but the RBFNN evaluation process finishes in only 0.076 seconds. The system achieves a performance increase of approximately **60x** when compared to actual measurements and the system achieves a performance increase of **18x** when compared to the training coarse solver.

Accuracy and Generalisation The RBFNN achieves an L_2 error of 0.0143 which approaches the approximate error of the coarse training data that measures 0.0122. The surrogate shows high accuracy in representing the main soliton peak. The coarse grid exhibits a L_∞ error increase to 0.057 which occurs at the dispersive tails of the wave. The Gaussian kernels improve numerical noise reduction of the coarse mesh yet they function as a low-pass filter which causes slight reduction in the peak frequency oscillations of the dispersive wave components.

4.5 Strengths and Limitations of the Proposed Method

By honing FEM solutions without requiring denser meshes, the FEM-RBFNN method improves accuracy and reduces computational costs. Its ability to interpolate helps reduce numerical oscillations and enhance generalisation, hence facilitating the handling of complicated geometries. Still, parameter selection is quite important, as poor choices can lead to instability or suboptimal approximations. Furthermore, even if FEM-RBFNN speeds up predictions, the initial RBFNN training adds significant computational overhead, especially when clustering with K-Means. As shown in situation 3, FEM alone was sufficient; thus, the usefulness of the hybrid model depends on the situation. FEM-RBFNN is most helpful when FEM suffers from accuracy issues. FEM-RBFNN provides, all things considered, a balanced and effective method for solving difficult PDEs, greatly enhancing numerical solutions while preserving computational feasibility.

5 Conclusion and Future Work

5.1 Summary of Findings

This study successfully implemented a hybrid FEM-RBFNN framework to optimise the numerical simulation of the Nonlinear Schrödinger Equation (NLSE). While the Finite Element Method (FEM) provides a robust mathematical foundation for managing complex wave dynamics, high-fidelity simulations on fine meshes are often computationally prohibitive. By integrating Radial Basis Function Neural Networks (RBFNNs), this work established an efficient surrogate model capable of reconstructing high-resolution wave profiles from computationally "cheap" coarse-grid data.

The benchmark tests showed that the soliton propagation testing demonstrated that the FEM-RBFNN method achieved a complete transformation of computational requirements for its calculations. The surrogate model achieved a speed improvement of 60 times, as indicated by the quantitative results, which showed it performed at

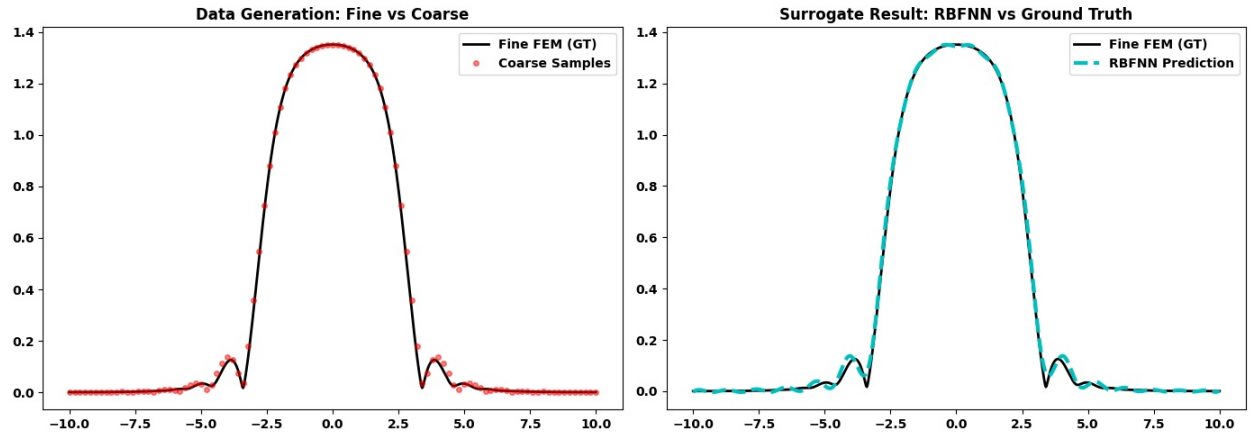


Fig. 2 Comparison of numerical solutions: (Left) Generation of training data via coarse sampling against the ground truth; (Right) RBFNN prediction vs. Fine FEM ground truth.

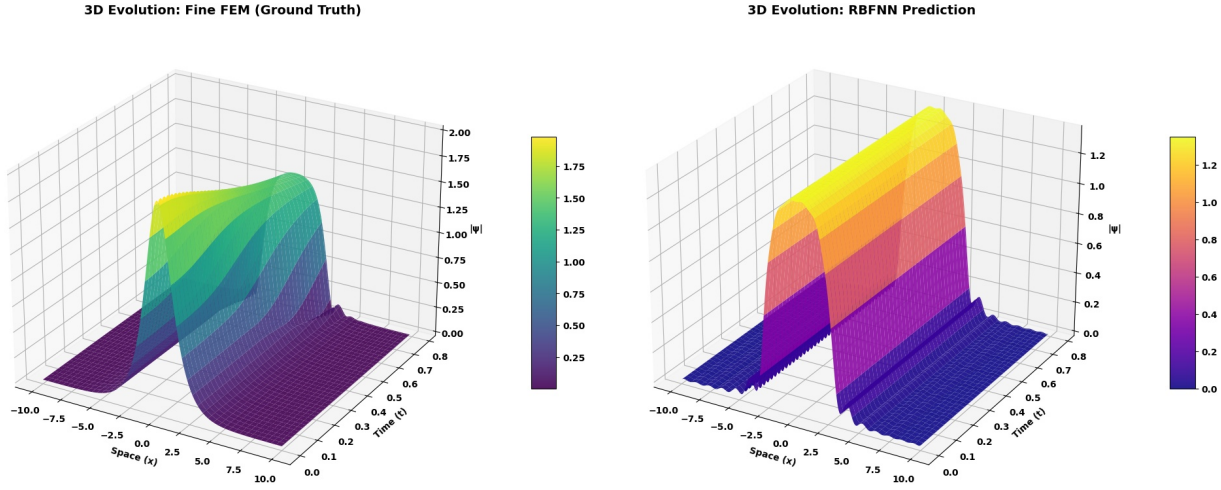


Fig. 3 Side-by-side 3D comparison of wave magnitude evolution. The left panel displays the Fine FEM (Ground Truth), while the right panel displays the RBFNN Prediction. The RBFNN successfully captures the qualitative features of the soliton's propagation. However, as seen in the right panel, the RBFNN tends to act as a low-pass filter, slightly smoothing the dispersive ripples found at the base of the soliton peak. This is an expected characteristic of Gaussian RBF kernels when the centre density M is lower than the Nyquist frequency of the underlying mesh.

0.08 seconds for the fine-mesh ground truth, compared to 4.60 seconds for the coarse-mesh ground truth. The RBFNN model produced an L_2 error of approximately 0.014, which closely matched the actual discretisation error of the coarse training data, 0.012. The results demonstrate that RBFNNs can use K-Means clustering to select their optimal centres, which serve as continuous interpolators that eliminate numerical artefacts arising from coarse FEM distributions.

The RBFNN system preserves the coarse training set discretisation boundaries while providing users with a rapid differentiable model that displays wave amplitude. The hybrid method provides an effective and promising solution that enables researchers to conduct real-time wave analysis and study multiple parameters at large scales, whereas the standard FEM fine-mesh approach requires an unfeasible computational budget.

The FEM-RBFNN system requires more work on automated hyperparameter tuning for the RBF shape parameter ε. The proposed strategy provides a direct pathway to high-accuracy numerical results that require fewer computational resources.

The FEM-RBFNN surrogate system provides reliable speed and accuracy, making it suitable for real-time wave-propagation analysis and large parameter sweeps that exceed the limits of full FEM simulations.

5.2 Contributions to Knowledge

The research project demonstrates how numerical analysis advances through the successful application of the Finite Element Method (FEM) and Radial Basis Function Neural Networks (RBFNNs), which serve as fast surrogates

for modelling nonlinear wave behaviour. The study demonstrates that RBFNNs can serve as "super-resolution" interpolators, recreating the complete soliton profile using limited common grid information.

The work achieves its primary goal by reducing computational requirements. The semi-implicit Crank-Nicolson FEM method produced training data that enabled the hybrid model to achieve a 60x speedup while keeping the Nonlinear Schrödinger Equation (NLSE) solution unaffected. The study demonstrates that K-Means clustering improves RBF centre placement by enabling the neural network to correctly identify location-based events, including soliton peaks. This method provides executives with a fixed system that combines established deterministic solvers with machine learning systems to address the nonlinear physics requirements for both spatial and temporal resolution.

5.3 Future Research Directions

The research must advance by extending the existing framework to provide complete time-dependent surrogates that can use Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks to model the future evolution of $|\psi|$ over all future time periods without rerunning the FEM solver. The engineering value of the hybrid model should increase through its extension to higher-dimensional PDEs, including 2D and 3D NLSE systems for research on optical filamentation and multi-physics fluid dynamics.

The RBFNN evaluation should use GPU-accelerated systems, as this implementation will enhance the efficiency of real-time wavefront sensing during control processes. Research into automated hyperparameter optimisation must be conducted to test the RBF shape parameter ε and the number of centres M across various initial conditions. The FEM-RBFNN framework should be used for industrial-scale simulations in structural analysis, heat transfer, and electromagnetics to test its ability to serve as a general scientific computing tool in hybrid scientific computing.

References

1. A. Sacchetti, et al. (2011). Neural networks to solve partial differential equations: A comparison with finite elements.
2. M. O. Ogunniran, et al. (2024). Enhanced rational multi-derivative integrator for singular problems with applications to advection equations. *Ain Shams Engineering Journal*.
3. F. Mostajeran and S. M. Hosseini (2023). Radial basis function neural network (RBFNN) approximation of Cauchy inverse problems. *Computers & Mathematics with Applications*, 141, 129–144. DOI: 10.1016/j.camwa.2023.04.026
4. S. Cuomo, et al. (2022). Scientific machine learning through physics-informed neural networks: Where we are and what's next. *arXiv*. https://arxiv.org/abs/2201.05624
5. C. Uriarte (2023). Solving partial differential equations using artificial neural networks (Doctoral dissertation, University of the Basque Country). Retrieved from https://addi.ehu.es/handle/10810/68335
6. W. K. Liu, S. Li, and H. S. Park (2022). Eighty years of the finite element method: Birth, evolution, and future. *Archives of Computational Methods in Engineering*, 29, 4431–4453. DOI: 10.1007/s11831-022-09740-9
7. J. P. Narain (2021). Comparison of neural network and finite difference solutions. *American Journal of Computational and Applied Mathematics*, 11(3), 65–70.
8. P. B. Zhou (1993). Finite difference method. In *Numerical Analysis of Electromagnetic Fields*. Springer.
9. D. Komatitsch and J. Tromp (1999). Introduction to the spectral element method for three-dimensional seismic wave propagation. *Geophysical Journal International*, 139(3), 806–822. DOI: 10.1046/j.1365-246X.1999.00967.x
10. H. Chen, L. Kong, and W.-J. Leng (2010). Numerical solution of PDEs via integrated radial basis function networks with adaptive training algorithm.
11. I. Babuska and W. C. Rheinboldt (1978). Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15(4), 736–754. DOI: 10.1137/0715049
12. G. Cybenko (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4), 303–314. DOI: 10.1007/BF02551274
13. S. Hochreiter and J. Schmidhuber (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. DOI: 10.1162/neco.1997.9.8.1735
14. G. Bebis and M. Georgiopoulos (1994). Feed-forward neural networks. *IEEE Potentials*, 13(4), 27–31. DOI: 10.1109/45.329294
15. S. Chen, C. F. N. Cowan, and P. M. Grant (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2), 302–309. DOI: 10.1109/72.80341
16. J. Berg and K. Nyström (2018). Neural network augmented inverse problems for PDEs. Department of Mathematics, Uppsala University.
17. S. Emmanuel, S. Sathasivam, and M. O. Ogunniran (2024). Multi-derivative hybrid block methods for singular initial value problems. *Scientific African*, 24, e02141.

18. S. K. Mitusch, S. W. Funke, and M. Kuchta (2021). Hybrid FEM-NN models: Combining artificial neural networks with the finite element method. Simula Research Laboratory.
19. S. Emmanuel, S. Sathasivam, and M. O. Ogunniran (2024). Leveraging feed-forward neural networks to enhance the hybrid block derivative methods. *Journal of Computational Mathematics and Data Science*, 13, 100101.
20. M. O. Ogunniran, et al. (2025). Harnessing neural networks in hybrid block integrator for efficient solution of boundary value problems. *Thermal Advances*, 2, 100022.
21. M. Abadi (2015). TensorFlow.org. Retrieved from https://www.tensorflow.org
22. A. Paszke, et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of NeurIPS 2019* (pp. 1–12).
23. A. Logg (2016). Mesh generation in FEniCS. Retrieved from http://www.logg.org/anders/2016/generation-in-fenics