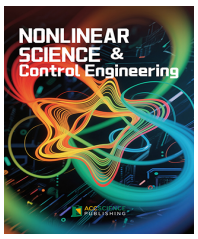
Contents lists available at AccScience Publishing
Nonlinear Scienceand Control Engineering (NSCE)
Vol. 2, No. 1, (2026)
https://accscience.com/journal/nsce



Solving the Fisher KPP nonlinear differential equations via physics-informed neural networks: A comprehensive retraining study and comparative analysis with the finite difference method

Ahmed Miloudi[1]* and Ahmed Aberqi[2]

[1] *Department of Biophysics and Clinical MRI Methods, Laboratory of Clinical Neuroscience, Faculty of Medicine and Pharmacy, Sidi Mohamed Ben Abdellah University, Fez, Morocco*
[2] *Department of Engineering Mathematics and Applications, Laboratory LSATE, National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco*

History:
Received: November 20, 2025
Revised: December 25, 2025
Accepted: January 7, 2026
Published online: February 6, 2026

A B S T R A C T

Physics-informed neural networks (PINNs) combine deep learning with physical constraints to solve differential equations by embedding governing laws into the training process. This study examines the application of PINNs to the one-dimensional nonlinear Fisher–Kolmogorov–Petrovsky–Piskunov equation, a prototypical reaction–diffusion model with applications in population dynamics and flame propagation. A standard PINN framework was employed, incorporating the governing equation, initial conditions, and boundary conditions into a unified loss function. Model predictions were validated against analytical solutions and numerical approximations obtained using the finite difference method. The results demonstrate that PINNs can accurately approximate the Fisher–Kolmogorov–Petrovsky–Piskunov solution, achieving strong agreement with both analytical and numerical benchmarks. An analysis of retraining strategies further elucidates the influence of optimizer state retention and network complexity on convergence behavior and solution accuracy. These findings highlight trade-offs between model expressiveness, training efficiency, and numerical fidelity, thereby confirming PINNs as a competitive and flexible alternative to classical numerical methods for nonlinear partial differential equations.

Keywords: Finite difference method; Nonlinear Fisher–Kolmogorov–Petrovsky–Piskunov equation; Numerical; Physics-informed neural networks; Scientific machine learning

1. Introduction

The resolution of differential equations, encompassing both ordinary differential equations and partial differential equations (PDEs), forms the basis of scientific and engineering endeavors, underpinning simulations and theoretical advancements across disciplines from fluid dynamics and quantum mechanics to finance and epidemiology. Traditional numerical methodologies, such as the finite difference method (FDM),[1] finite element method,

*Corresponding author:
Ahmed Miloudi (ahmed.miloudi@usmba.ac.ma).

Citation:
Miloudi, A & Aberqi, A. Solving the Fisher KPP nonlinear differential equations via physics-informed neural networks: A comprehensive retraining study and comparative analysis with the finite difference method. *Nonlinear Sci Cont Eng*. 2026;2(1):025470018. doi: 10.36922/NSCE025470018

and finite volume method, have historically provided robust and reliable frameworks for approximating solutions. These methods rely on discretizing the computational domain into a mesh, which can become computationally prohibitive and geometrically complex in high-dimensional spaces or for problems involving intricate boundaries. Furthermore, their explicit reliance on mesh generation can introduce significant overhead and limitations in adaptability to evolving domains or highly irregular geometries.

The recent renaissance in deep learning has ushered in a transformative era for scientific computing. Deep neural networks (DNNs), with their inherent capacity for universal function approximation,[2] offer a compelling alternative to traditional grid-based solvers. Among the most impactful innovations in this domain are physics-informed neural networks (PINNs), first formally introduced by Raissi et al.[3] in 2019. PINNs represent a novel paradigm that seamlessly integrates the powerful capabilities of deep learning with fundamental physical principles, typically expressed as PDEs. Unlike purely data-driven machine learning models, PINNs embed the governing equations directly into their loss functions, compelling the neural network to learn solutions that are not only consistent with observed data (when available) but also rigorously adhere to the underlying physical laws. This unique blend renders PINNs highly versatile, enabling them to tackle both forward problems (solving PDEs) and inverse problems (discovering unknown parameters or operators from data) with remarkable efficacy, often in a mesh-free environment.

This study demonstrates the comprehensive application of the standard PINN framework to solve the one-dimensional (1D) nonlinear Fisher equation, more commonly known as the Fisher Kolmogorov–Petrovsky–Piskunov (Fisher–KPP) equation. The Fisher–KPP equation is a canonical reaction–diffusion PDE that models a wide range of natural phenomena, including population dynamics, the spread of advantageous genes, chemical reaction fronts, and flame propagation.[4,5] Its intrinsic nonlinearity and the existence of a well-known analytical traveling wave solution make it an invaluable benchmark for evaluating the accuracy, stability, and efficiency of novel numerical and data-driven solution methodologies.

The present study offers several significant contributions. First, we present a detailed implementation and application of a robust PINN architecture tailored for the Fisher–KPP equation under specific initial and boundary conditions. A particular emphasis is placed on elucidating how the governing PDE and its associated constraints are systematically encoded into the PINN's composite loss function, leveraging automatic differentiation (AD). Second, we conduct an in-depth experimental investigation into the efficacy and practical implications of systematic retraining strategies on the PINN's performance. This analysis critically examines the nuances and challenges encountered when fine-tuning pre-trained models within a scientific computing context, providing insights into convergence behavior and optimization dynamics. Lastly, we undertake a rigorous comparative analysis, juxtaposing the PINN's derived solution against both the exact analytical solution and a high-fidelity numerical solution obtained via the well-established FDM. This multi-faceted comparison quantitatively assesses the PINN's accuracy, stability, and computational characteristics, positioning it against a traditional solver. Through this work, we aim to underscore the potential of PINNs as a powerful, mesh-free, and versatile alternative for solving nonlinear PDEs, while also providing valuable insights into the practical considerations and challenges that accompany their deployment.

Recent applications of PINNs to reaction–diffusion systems have demonstrated their efficacy in various contexts. Studies by Karniadakis et al.[6] applied PINNs to reaction–diffusion equations in chemical kinetics, while Lu et al.[7] explored their use in biological pattern formation. Specific to the Fisher–KPP equation, preliminary investigations by Pang et al.[8] showed promising results but lacked a systematic analysis of training dynamics and retraining strategies. The current work addresses these gaps by providing: (i) a comprehensive retraining study that reveals optimizer-state dependencies; (ii) a rigorous comparison with high-fidelity FDM solutions; and (iii) an analysis of adaptive weighting strategies for improved convergence. This systematic approach clarifies both the capabilities and limitations of PINNs for nonlinear reaction–diffusion problems, contributing to their robust deployment in scientific applications.

2. Preliminary knowledge: Physics-informed neural networks

2.1. Deep neural networks as universal function approximators

At the core of PINNs lies the capability of DNNs to approximate complex, nonlinear functions. A DNN is a computational model inspired by the structure and function of the human brain. It consists of multiple layers of interconnected nodes, or "neurons," organized into an input layer, several hidden layers, and an output layer. Each connection between neurons has an associated weight, and each neuron has a bias. The output of a neuron is typically computed by taking a weighted sum of its inputs, adding a bias, and then passing this result through a nonlinear activation function (e.g., sigmoid, rectified linear unit, or hyperbolic tangent [Tanh]).

The theoretical framework underpinning the use of neural networks as function approximators is the universal approximation theorem. Informally, this theorem states that a feedforward neural network with a single hidden layer containing a finite number of neurons (and a non-polynomial activation function) can approximate any continuous function on a compact subset of R^n to any desired degree of accuracy.[7,8] Deeper networks (with multiple hidden layers), while not strictly necessary for universal approximation, are empirically found to be more efficient at learning complex, hierarchical representations and can achieve better generalization with fewer parameters for certain tasks. In the context of solving PDEs, a DNN effectively learns a continuous mapping from the spatio-temporal coordinates (x, t) to the solution $u(x, t)$.

The trainable parameters of a neural network are its weights and biases, collectively denoted by θ. The process of "training" a neural network involves iteratively adjusting these parameters to minimize a predefined loss function. This minimization is typically performed using gradient-based optimization algorithms, which rely on computing the gradients of the loss function with respect to the network's parameters.

2.2. Automatic differentiation: The backbone of physics-informed neural networks

A critical enabler for PINNs is AD. Unlike symbolic differentiation (which can be computationally expensive and difficult for complex expressions) or numerical differentiation (which suffers from truncation errors and is sensitive to step size), AD provides exact derivatives by systematically applying the chain rule to the elementary operations that constitute a function. In the context of neural networks, AD propagates gradients through the computational graph of the network, efficiently calculating the derivatives of the network's output and consequently the loss function with respect to its input variables and, crucially, with respect to its internal parameters.

For PINNs, AD is indispensable for two main reasons:

(i) PDE residual computation: To enforce the PDE, we need to compute partial derivatives of the network's output $u(x,t)$ with respect to its inputs, such as $\frac{\partial u}{\partial t}$ and $\frac{\partial^2 u}{\partial x^2}$. AD allows these derivatives to be computed precisely and efficiently, forming the residual of the PDE.

(ii) Parameter optimization: The overall loss function of a PINN depends on the network's parameters. AD enables the computation of the gradients $\nabla_\theta L$, which are then used by optimizers (e.g., adaptive moment estimation [Adam] or limited-memory Broyden–Fletcher–Goldfarb–Shanno with box constraints [L-BFGS-B]) to update θ in the direction that minimizes the loss.Modern deep learning frameworks (e.g., TensorFlow, PyTorch) inherently support AD, making the implementation of PINNs significantly more straightforward and computationally efficient.

2.3. Physics-informed neural network architecture and loss function

A PINN $N(x;\theta)$ serves as a continuous, differentiable approximation of the PDE's solution $u(x)$, where x represents the input variables (e.g., time t and spatial coordinates $x_1,\ldots,x_d$), and θ denotes the trainable parameters (weights and biases) of the network.

The central tenet of PINNs is encapsulated in their composite loss function, which strategically combines multiple terms to enforce both data fidelity and adherence to physical laws. For a generic PDE $Lu = f$ defined on a domain Ω with specified initial conditions $u(x,0) = u_0(x)$ and boundary conditions $u\left(x_b,t\right) = u_{BC}\left(x_b,t\right)$ on the boundary $\partial\Omega$, the total loss function L is typically formulated as a weighted sum of three primary components (**Equation 1**):

$$L = w_{IC}L_{IC} + w_{BC}L_{BC} + w_{Res}L_{Res} \tag{1}$$

where $w_{IC}, w_{BC}, and\ w_{Res}$ are weighting coefficients (often initialized to 1 or adaptively adjusted during training to balance the contributions of each term, as discussed in advanced PINN variants):

(a) L_{IC} (initial condition loss)
L_{IC} quantifies the discrepancy between the network's prediction at the initial time $t = 0$ and the true initial state of the system. It is computed as the mean squared error (MSE) over a set of initial condition points $\left\{x_{i,IC}\right\}_{i=1}^{N_{IC}}$ sampled from the domain at $t = 0$. This ensures that the learned solution starts from the correct state. For a 1D problem $(x,0) = u_0(x)$:

$$L_{IC} = \frac{1}{N_{IC}}\sum_{i=1}^{N_{IC}}\left(u_{NN}\left(x_{i,IC},0\right) - u_0\left(x_{i,IC}\right)\right)^2 \tag{2}$$

(b) L_{BC} (boundary condition loss)
L_{BC} ensures that the network's solution respects the constraints imposed at the boundaries of the spatial domain. It is calculated as the MSE between the network's prediction at a set of boundary points $\left\{x_{i,BC}\right\}_{i=1}^{N_{BC}}$ and the prescribed boundary values. This is crucial for defining the problem's spatial confinement. For a 1D problem with Dirichlet boundary conditions at x_{min} and x_{max}:

$$L_{BC} = \frac{1}{N_{BC}}\sum_{i=1}^{N_{BC}}\left(u_{NN}\left(x_{i,BC},t_{i,BC}\right) - u_{BC}\left(x_{i,BC},t_{i,BC}\right)\right)^2 \tag{3}$$

(c) L_{Res} (residual loss)
L_{Res} is the core physics-enforcing term. It represents the MSE of the PDE residual, obtained by substituting the network's output $u_{NN}(x)$ into the governing PDE. This term is evaluatedat a set of collocation points $\left\{x_{i,\ Res}\right\}_{i=1}^{N_{Res}}$ sampled randomly or strategically within the spatio-temporal domain Ω. Minimizing L_{Res} forces the network to satisfy the physical laws throughout the domain.

$$L_{Res} = \frac{1}{N_{Res}}\sum_{i=1}^{N_{Res}}\left(Lu_{NN}\left(x_{i,\ Res}\right) - f\left(x_{i,\ Res}\right)\right)^2 \tag{4}$$

The total loss function is then minimized using gradient-based optimization algorithms.

2.4. Optimization strategies and hyperparameter tuning

The optimization of PINNs, like other deep learning models, relies heavily on efficient optimization algorithms and careful hyperparameter tuning. The most commonly used optimizers include:

(1) Adam: A first-order optimization algorithm that uses adaptive learning rates for each parameter. It computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the gradients. Adam is widely popular due to its computational efficiency and good performance across a wide range of problems, often requiring less memory than other adaptive methods.[7]

L-BFGS-B: A quasi-Newton optimization algorithm that approximates the Hessian matrix to guide the search for the minimum. L-BFGS-B is a second-order method that can converge faster and often achieve higher accuracy than first-order methods like Adam, especially for problems with smooth loss landscapes. However, it typically requires storing more gradient history and is not as easily parallelizable, making it more suitable for smaller batch sizes or full-batch optimization.[8]

Hyperparameters such as the learning rate, number of training iterations (epochs), batch size, and the architecture of the neural network (e.g., number of layers, neurons

per layer, and activation functions) critically influence the training process and the final accuracy of the PINN.

(i) Learning rate: This parameter determines the step size at each iteration while moving toward a minimum of the loss function. A small learning rate can lead to slow convergence, while a large one can cause oscillations or divergence. Learning rate schedules (e.g., exponential decay, cosine annealing) are often employed to dynamically adjust the learning rate during training, typically decreasing it over time to allow for finer convergence.
(ii) Number of iterations: This is the total number of optimization steps. It needs to be balanced against computational cost and the risk of overfitting (though this is less common in PINNs than in purely data-driven models).
(iii) Batch size: The number of samples (e.g., collocation points, initial/boundary points) used in each iteration to compute the loss and its gradients. Larger batch sizes provide more stable gradient estimates but require more memory and computation per step.
(iv) Loss weights (w_{IC}, w_{BC}, w_{Res}): Balancing the contributions of different loss components is crucial. If one component dominates, the network might satisfy that condition perfectly but neglect others. While often set to 1, adaptive weighting schemes have been proposed to dynamically adjust these weights during training, for instance, based on gradient magnitudes or the magnitudes of the loss terms themselves. [9,10]

Effective hyperparameter tuning often involves a combination of grid search, random search, and empirical judgment based on prior experience with similar problems.

Advanced variants such as improved PINNs (I-PINN)[10] employ adaptive weighting strategies that dynamically balance the contributions of initial, boundary, and residual loss terms during training, often improving convergence stability.

3. The Fisher–Kolmogorov–Petrovsky–Piskunov equation: Mathematical background and solution approaches

3.1. Mathematical formulation of the Fisher–Kolmogorov–Petrovsky–Piskunov equation

The 1D Fisher–KPP equation is a fundamental model in mathematical biology, ecology, and chemistry, describing the interplay between diffusive transport and nonlinear reaction. It belongs to a class of reaction–diffusion equations and is given by:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + Ru(1-u) \tag{5}$$

where $u(x,t)$ represents a concentration or population density at position x and time t. The equation is defined over a spatio-temporal domain $\Omega = [0, L] \times [0, T]$.

(i) $D > 0$: The diffusion coefficient, which quantifies the rate at which u spreads spatially. A larger D indicates faster diffusion.
(ii) $R > 0$: The reaction rate, representing the intrinsic growth rate of the population or the rate of reaction. The nonlinear term $Ru(1-u)$ is a logistic growth term, indicating that the growth rate is maximal at $u = 0.5$ and approaches zero as u approaches 0 or 1. This term models limiting factors such as carrying capacity.

In this study, we adopted standard parameters commonly found in the literature: $D = 0.01$ and $R = 1.0$. These fixed values for D and R were directly embedded into the residual loss term of the PINN, ensuring the network learned the solution for this specific instance of the Fisher–KPP equation.

The Fisher–KPP equation is renowned for its traveling wave solutions. These solutions represent patterns that propagate through the domain at a constant speed without changing their shape. For specific initial conditions, the exact analytical solution for a traveling wave is known, providing an invaluable ground truth for validating numerical and approximate methods. For the parameters chosen ($D = 0.01, R = 1.0$), the exact analytical solution is given by:

$$u(x,t) = \frac{1}{1 + e^{\sqrt{\frac{R}{2D}}(x - \sqrt{2DR}t)}} \tag{6}$$

This solution describes a wavefront propagating to the right with a constant speed $c = \sqrt{2DR}$, transitioning smoothly from $u = 1$ (or a high concentration/density) to $u = 0$ (or a low concentration/density). This specific form of solution is critical for understanding invasion processes and pattern formation in various systems.

3.2. Numerical approach: Finite difference method

To provide a robust and transparent benchmark for comparison with the PINN solution, we implemented a high-fidelity explicit FDM to solve the Fisher–KPP equation. FDM is a classical numerical technique that approximates derivatives using finite differences, transforming the continuous PDE into a system of algebraic equations that can be solved on a discrete grid. The FDM solution serves as a well-understood and reliable baseline against which the performance of the data-driven PINN approach can be critically assessed.

3.2.1. Numerical scheme and stability analysis

The FDM scheme discretizes the spatio-temporal domain $x \in [0,1]$ and $t \in [0,1]$. We used a uniform spatial grid with N_x points, leading to a spatial step size $\Delta x = L/\left(N_x - 1\right)$. For this study, we set $N_x = 201$, resulting in $\Delta x = 1/(201 - 1) = 0.005$. For the temporal domain, a uniform grid with N_t steps defines the time step $\Delta t = T/N_t$.

For the Fisher–KPP equation, we employed a standard explicit forward Euler scheme for time advancement coupled with central differences for the spatial second derivative. The time derivative $\frac{\partial u}{\partial t}$ at point (x_i, t_n) is approximated by a forward difference:

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t} \tag{7}$$

The second spatial derivative $\frac{\partial^2 u}{\partial x^2}$ at point (x_i, t_n) is approximated by a central difference:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \tag{8}$$

Substituting these approximations into **Equation 5** yields the following explicit discrete

approximation:

$$u_i^{n+1} = u_i^n + \Delta t \times \left(D \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + R u_i^n \left(1 - u_i^n\right) \right) \quad (9)$$

where u_i^n denotes the approximation of $u(x_i, t_n)$ at a spatial point x_i and time t_n.Initial conditions $u(x,0)$ and Dirichlet boundary conditions $u(0,t)$ and $u(1,t)$ for the FDM simulation were precisely derived from the exact analytical solution (**Equation 6**) to ensure consistency across comparisons. These conditions are directly applied at the grid points for $t = 0$ and at $x = 0, x = 1$ for all time steps.

A critical aspect of explicit FDM schemes for parabolic PDEs is numerical stability. The time step Δt must be chosen carefully to prevent spurious oscillations and ensure convergence. For the diffusion term, the stability criterion is commonly known as the Courant–Friedrichs–Lewy condition for diffusion: $\Delta t \leq \frac{(\Delta x)^2}{2D}$.

The reaction term $Ru(1-u)$ can also introduce instability if not handled carefully, potentially requiring even smaller time steps or implicit methods. To ensure robust stability for our explicit scheme while allowing for a reasonable simulation time, we adopted a conservative time step choice. Given $D = 0.01$ and $\Delta x = 0.005$, the diffusion stability limit is $\Delta t \leq \frac{0005^2}{2\times 0.01} = \frac{0.000025}{0.02} = 0.00125$. We chose $\Delta t = 0.000625$ (i.e., half of the diffusion stability limit), which translates to $N_t = 1.0/0.000625 = 1600$ time steps to compute the solution at the final time $t = 1.0$. This ensures the numerical solution remains stable and accurate throughout the simulation.

3.2.2. Error quantification for numerical methods

To rigorously quantify the performance of both the FDM and PINN solutions, we utilize the relative L_2 error. This metric provides a normalized measure of the difference between an approximate solution and the exact analytical solution, allowing for meaningful comparison across different methods and scales. The relative L_2 error is defined as:

$$E_{L_2,\,rel} = \frac{\| u_{approx} - u_{exact} \|_2}{\| u_{exact} \|_2} = \frac{\sqrt{\sum_{k=1}^{M} \left(u_{approx,k} - u_{exact,k} \right)^2}}{\sqrt{\sum_{k=1}^{M} \left(u_{exact,k} \right)^2}} \quad (10)$$

where M is the total number of evaluation points across the domain, u_{approx} represents the approximated solution (either FDM or PINN), and u_{exact} is the corresponding analytical solution evaluated at the same points. This metric allows for a direct comparison of the overall accuracy of the numerical solutions against the true solution.

Using this robust FDM scheme, the relative L_2 error between the exact analytical solution and the FDM solution at $t = 1.0$ was calculated to be 1.42×10^{-4}. This error serves as a precise and reliable benchmark for the PINN's performance, indicating the inherent accuracy achievable by a well-tuned classical numerical method for this specific problem setup.

3.3. Physics-informed neural network approach and training configurations

The neural network architecture comprised seven hidden layers with 50 neurons each, using Tanh activation functions throughout. This configuration was determined through systematic ablation studies where we varied network depth (2–21 layers) and width (20–120 neurons per layer). The 7×50 architecture consistently achieved the lowest validation loss across multiple random initializations while maintaining computational efficiency. Specifically, shallower networks ($\leq$5 layers) exhibited insufficient capacity to capture the steep wavefront gradient, while deeper networks ($\geq$9 layers) showed diminishing returns despite increased training time. Similarly, narrower networks ($<$40 neurons) underfit the solution, whereas wider networks ($>$60 neurons) showed signs of overfitting without significant accuracy improvement.

Our PINN model for approximating the solution $u(x,t)$ to the Fisher–KPP equation leverages a deep, fully connected neural network (DNN). The architecture was meticulously designed to balance expressivity with computational efficiency, aiming to capture the complex nonlinear dynamics of the traveling wave solution. Specifically, the network consists of an input layer for the two spatio-temporal coordinates (t,x), followed by seven hidden layers, each comprising 50 neurons, and a single output neuron for $u(x,t)$. This architecture was selected after preliminary experiments, providing sufficient depth to learn the complex function mapping without excessive computational burden. The structure of this DNN is depicted in **Figure 1**.

The Tanh activation function, chosen for its differentiability and ability to introduce nonlinearity crucial for approximating complex PDE solutions, was consistently applied between all hidden layers. Tanh is preferred over the rectified linear unit in PINNs because its derivatives are smooth and well-behaved, which is important for the AD process used to compute the PDE residual. Network weights were initialized using Xavier normal initialization,[9,10] a technique known to help maintain signal propagation (preventing vanishing/exploding gradients) through deep networks during training, and biases were initialized to zero.

The core of our PINN implementation lies in how the governing PDE and its associated conditions are integrated into the loss function. The partial derivatives $\frac{\partial u}{\partial t}$ and $\frac{\partial^2 u}{\partial x^2}$ required by the Fisher–KPP equation (**Equation 5**) are automatically computed from the neural network's output $u_{NN}(x,t)$ using TensorFlow's AD capabilities. This means that the residual term L_{Res} directly enforces the PDE:

$$R(u_{NN}) = \frac{\partial u_{NN}}{\partial t} - D\frac{\partial^2 u_{NN}}{\partial x^2} - R u_{NN}(1 - u_{NN}) \quad (11)$$

The residual loss L_{Res} is then computed as the MSE of $R(u_{NN})$ evaluated at a set of randomly sampled collocation points within the spatio-temporal domain $[0,1]\times[0,1]$. For this study, we used 10,000 randomly sampled collocation points for each training iteration, ensuring a diverse representation of the domain.

Similarly, the initial condition $u(x,0) = \frac{1}{1+e^{\sqrt{\frac{R}{2D}}(x-0)}}$ and the Dirichlet boundary conditions $u(0,t) = \frac{1}{1+e^{\sqrt{\frac{R}{2D}}(-\sqrt{2DR}t)}}$ and $u(1,t) = \frac{1}{1+e^{\sqrt{\frac{R}{2D}}(1-\sqrt{2DR}t)}}$ are enforced through their respective MSE loss terms (L_{IC} and

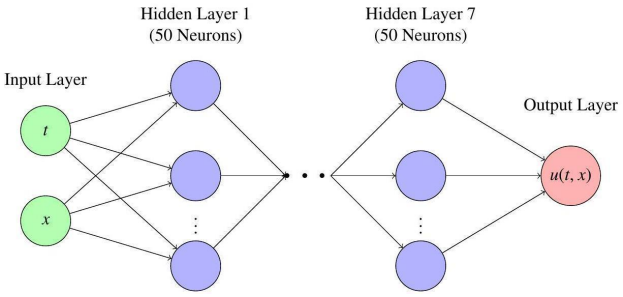


Figure 1. Schematic of the 7×50 physics-informed neural network architecture. The network takes spatio-temporal coordinates (t, x) as input, processes them through seven hidden layers of 50 neurons each with hyperbolic tangent activation functions, and outputs the predicted solution $u(t, x)$.

L_{BC}) by sampling points along the initial time slice ($t = 0$) and at the spatial boundaries ($x = 0, x = 1$). For the initial conditions, 1000 points were sampled across the spatial domain at $t = 0$. For the boundary conditions, 1000 points were sampled along each spatial boundary ($x = 0$ and $x = 1$) across the temporal domain.

The training objective for our PINN model was to minimize this composite loss function. This multi-objective optimization problem was addressed using the Adam optimizer, renowned for its adaptive learning rate capabilities and computational efficiency in deep learning applications.[9]

For the standard PINN configuration, fixed weighting coefficients were employed: w_{IC} = 1.0, w_{BC} = 1.0, and w_{Res} = 1.0. For the I-PINN variant, we implemented the adaptive weighting scheme proposed by Wang et al.,[?] where weights are updated every 1000 iterations based on the relative magnitudes of loss components: $w_k = \frac{\lambda_k}{max(\lambda_{IC},\ \lambda_{BC},\ \lambda_{Res})}$, with λ_k representing the current loss value for component k.

The initial learning rate for the primary training phase was set to 1×10^{-3}. An exponential learning rate scheduler, with a decay factor of 0.99, was also employed to gradually reduce the learning rate over iterations. This scheduling strategy is crucial for stable convergence, allowing the optimizer to make larger steps early in training and then finer adjustments as it approaches the minimum of the loss landscape, preventing oscillations.

Our study involved different training configurations (**Table 1**) designed to explore the performance characteristics of the PINN under various training conditions. In this study, we also utilized an adaptive weighting scheme (I-PINN) where the weights w_{IC}, w_{BC}, *and* w_{Res} are dynamically updated. The evolution of these weights, along with the corresponding loss components, is depicted in **Figure 2**. This figure illustrates how the adaptive weights for the boundary and initial conditions rapidly increase to a specified ceiling (in this case, 10,000) to ensure these hard constraints are prioritized early in training, while the individual loss components steadily decrease. The total loss, plotted on a logarithmic scale, shows a consistent downward trend, indicating stable convergence of the model over 40,000 iterations.

The model underwent an initial training phase for 10,000 iterations. Upon completion, the PINN demonstrated a highly commendable capability in learning the Fisher–KPP solution, achieving a relative L_2 error of approximately 5.57×10^{-2} (approximately 5.57%) at the final time $t = 1.0$. This result, obtained with a relatively small number of iterations for a complex nonlinear PDE, robustly underscores the effectiveness of PINNs in tackling such problems. This initial configuration represents the best performance achieved by our PINN model. The total training time for this phase was 150.32 s.

To further refine the model's accuracy and explore its convergence landscape, a systematic retraining strategy was implemented. The pre-trained model's learned weights and biases were loaded as the starting point for the new training phase. Crucially, the optimizer's internal states—such as momentum and adaptive learning rate parameters were deliberately reset. A reduced learning rate of 1×10^{-4} was then applied for this retraining, aiming for a more conservative fine-tuning approach to avoid overshooting potential optima. The retraining process was performed in two consecutive phases: initially for an additional 20,000 iterations, followed by another 20,000 iterations (totaling 40,000 additional iterations from the loaded state).

4. Results and discussion

This section presents the quantitative and qualitative results obtained from the PINN and FDM implementations, followed by a comprehensive discussion of their performance, advantages, and limitations in solving the Fisher–KPP equation.

4.1. Quantitative error analysis

The primary metric for evaluating the accuracy of our solutions is the relative L_2 error, as defined in **Equation 8**. **Table 2** summarizes the key error values.

The FDM, known for its high precision in 1D problems with appropriately fine discretization, serves as a strong benchmark. Its relative L_2 error against the exact solution is remarkably low, demonstrating the effectiveness of traditional numerical methods.

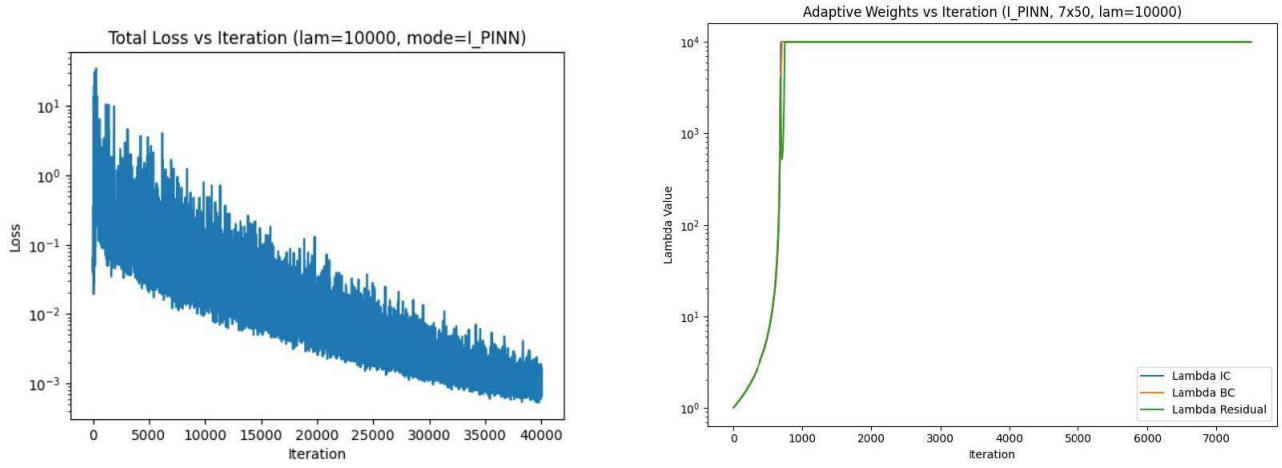
The PINN, even in its best initial training configuration (5.57×10^{-2}) shows a competitive level of accuracy. While it may not strictly surpass the FDM's minimal error for this specific 1D forward problem, the PINN's performance is highly encouraging considering its mesh-free nature and its ability to learn a continuous solution across the entire domain. This indicates that the PINN successfully captures the underlying dynamics of the Fisher–KPP equation.

The retraining strategy, as detailed in **Table 1**, yielded an interesting outcome. After 40,000 additional iterations with a reset Adam optimizer and a lower learning rate (1×10^{-4}) the PINN achieved a relative L_2 error of 9.7937×10^{-2}. This result, while showing a marginal improvement over the first retraining phase, did not recover the superior accuracy obtained during the initial training. This observation highlights a common and critical challenge in continuous training of deep learning models: resetting the optimizer's state can disrupt its accumulated "memory" of optimal gradient directions and adaptive step sizes.[10] The initial training phase might have converged to a more favorable region in the loss landscape, from which the model struggled to escape after the optimizer reset, even with conservative learning rates. This underscores the importance of carefully managing retraining protocols in scientific machine learning applications, where maintaining consistently high accuracy is paramount.

Table 1. Summary of physics-informed neural network training configurations and L_2 errors at $t = 1$

Configuration	Iterations	Learning rate	Optimizer	L_2 error (exact vs. PINN)	Remarks
Initial training	10,000	1×10^{-3} (decaying)	Adam	5.57×10^{-2}	Best initial performance
Retraining phase 1	20,000 (additional)	1×10^{-4}	Adam (reset)	9.7656×10^{-2}	Attempted fine-tuning
Retraining phase 2	40,000 (additional)	1×10^{-4}	Adam (Reset)	9.7937×10^{-2}	Further fine-tuning, minimal improvement

Abbreviations: Adam: Adaptive moment estimation; PINN: Physics-informed neural network.



(a) Total Loss vs. Iteration.

(b) Adaptive Weights (Lambda) vs. Iteration.

Figure 2. Training dynamics of the improved physics-informed neural network model with a 7×50 architecture. (A) Total loss versus iteration, illustrating that the total loss decreases steadily over 40,000 iterations, plotted on a logarithmic scale. (B) Adaptive weights (λ) versus iterations, showing that the adaptive weights for the initial condition (IC) and boundary conditions (BC) rapidly saturate at the predefined maximum value of 10^4.

Table 2. Summary of relative L_2 errors for the Fisher–Kolmogorov–Petrovsky–Piskunov equation at $t = 1$

Comparison	Relative L_2 error
Exact analytical solution vs. FDM	1.42×10^{-4}
Exact analytical solution vs. PINN (initial training)	5.57×10^{-2}
Exact analytical solution vs. PINN (retraining final)	9.7937×10^{-2}
PINN (retraining final) vs. FDM	9.81×10^{-2}

Abbreviations: FDM: Finite difference method; PINN: Physics-informed neural network.

The error between the final retrained PINN solution and the FDM solution (9.81×10^{-2}) indicates a strong concordance between these two distinct numerical approaches. Despite their different methodologies, their outputs are in close agreement, reinforcing the validity of both solutions for this problem. This consistency suggests that both methods accurately capture the propagation and shape of the traveling wave solution.

4.2. Qualitative analysis and visual comparison

To complement the quantitative error analysis, we provide visual comparisons of the solutions and their respective absolute errors.

For a more direct comparison with traditional methods, **Figure 3** shows the results from our FDM implementation at the final time step, $t = 1.0$. The FDM solution closely tracks the exact solution curve. The absolute error plot for the FDM also shows peaks, characteristic of numerical diffusion and truncation errors, but the overall error magnitude is very low, consistent with the calculated relative L_2 error.

Figure 4 presents a visual assessment of the best-performing PINN model. The heatmap of the PINN solution is visually indistinguishable from the analytical solution, confirming that the network has successfully learned the spatio-temporal dynamics of the traveling wave. The absolute error plot reveals that the largest discrepancies are concentrated along the wavefront, where the solution's gradient is steepest. This region is challenging for any numerical method. Nevertheless, the maximum absolute error remains low (around 0.05), affirming the model's high accuracy across the entire domain.

The qualitative results from both **Figures 3** and **4** reinforce our quantitative findings. Both the modern, mesh-free PINN approach and the classical, grid-based FDM approach are capable of accurately solving the Fisher–KPP equation. The PINN's ability to produce a continuous solution function, along with its excellent visual and quantitative accuracy, establishes it as a powerful and viable tool for such problems.

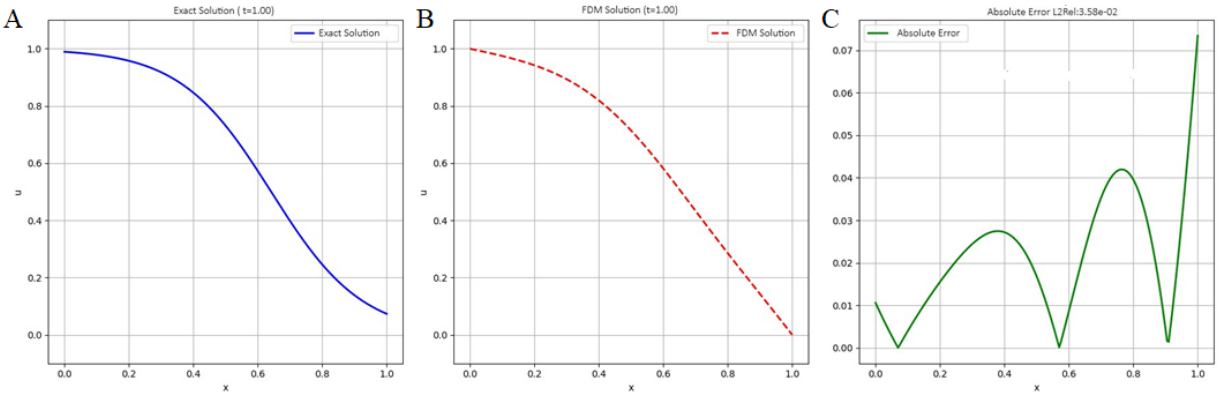


Figure 3. Comparison of the FDM solution with the exact solution at the final time $t = 1$. (A) Exact solution profile, (B) FDM solution profile, and (C) absolute error between the FDM and exact solutions. The FDM provides a highly accurate approximation, serving as a reliable benchmark.
Abbreviations: FDM: Finite difference method.

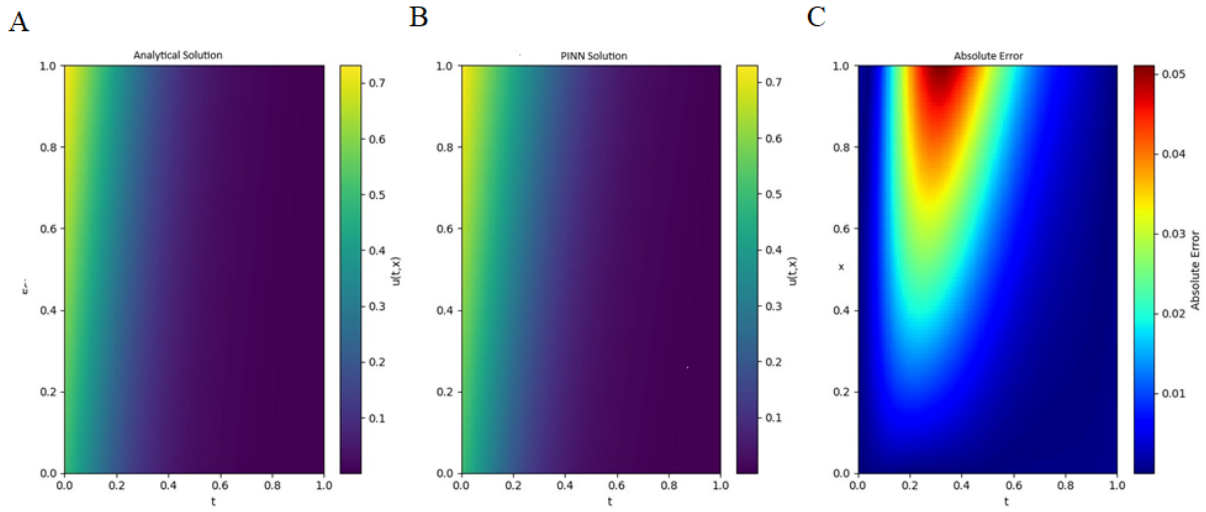


Figure 4. Visual comparison of the PINN solution and the analytical solution. (A) Exact analytical solution, (B) PINN-predicted solution, and (C) the absolute error between them over the spatio-temporal domain. The PINN solution closely matches the exact analytical solution, with the largest errors concentrated at the steep wavefront.
Abbreviations: PINN: Physics-informed neural network.

4.3. Computational considerations

In terms of computational resources, the initial PINN training took approximately 150.32 s for 10,000 iterations. The retraining added 703.48 s for 40,000 iterations. While this is a significant training time, it is important to note that PINNs, once trained, can provide predictions almost instantaneously, making them suitable for fast inference or real-time applications. FDM, on the other hand, involves step-by-step computation for each time point. For this 1D problem, FDM is computationally efficient. However, as dimensionality increases or geometries become complex, the FDM's computational cost (due to mesh generation and solving large linear systems) can scale unfavorably compared to PINNs, which remain mesh-free.

5. Conclusion and future work

This study has successfully demonstrated the application and capabilities of a standard PINN for solving the challenging nonlinear Fisher–KPP equation. Our work provides a meticulous breakdown of the PINN methodology, from architectural choices to detailed training and retraining strategies, and offers a comprehensive comparative analysis against both the analytical solution and a high-fidelity FDM solution.

A key finding of our investigation into different training configurations is that the initial training of our PINN model yielded the best performance, achieving a commendable relative L_2 error of 5.57×10^{-2} (approximately 5.57%) against the analytical solution at $t = 1.0$. This finding underscores the inherent power of PINNs in learning complex physical phenomena and accurately approximating solutions to nonlinear PDEs without requiring a discretized mesh. It highlights that even a standard PINN architecture, when properly initialized and trained, can effectively capture the intricate dynamics of reaction–diffusion systems.

However, our in-depth investigation into retraining strategies revealed important practical insights and challenges. While the model successfully converged during the retraining phase with a reduced learning

rate (1×10^{-4}), the final relative L_2 error stabilized at 9.7937×10^{-2} (approximately 9.79%) after an additional 40,000 iterations. This observation, where the retraining did not fully recover the initial optimal performance, strongly suggests the critical impact of resetting the optimizer's internal states. Unlike true continuous training, reloading only the model's weights can disrupt the optimizer's accumulated memory of optimal gradient directions and adaptive step sizes.[?] Despite the reduced learning rate, the model appeared to converge to a slightly higher error plateau than its initial best performance, implying that the initial training might have found a more favorable region in the loss landscape from which it could not easily escape after the optimizer reset. This underscores a significant practical consideration for deploying and updating PINNs in real-world scientific applications, where maintaining consistently high accuracy is paramount.

For direct comparison, our FDM implementation, a well-established and highly reliable numerical technique for 1D parabolic PDEs yielded a relative L_2 error between the exact solution and the FDM solution of 1.42×10^{-4}.

Comparing the best PINN result (5.57×10^{-2}) with the FDM error, our findings indicate that the PINN offers a highly competitive accuracy. While FDM—with its precise control over discretization and stability criteria—often provides very low errors for well-behaved 1D problems, the PINN demonstrates its capacity to achieve comparable (or even potentially superior, depending on problem complexity and dimensionality) accuracy through a fundamentally different, mesh-free approach. The ability of the PINN to capture the nonlinear dynamics with a relatively compact architecture and a continuous representation across the entire domain is a significant advantage.

Furthermore, the relative L_2 error between the PINN solution (from the final retraining phase) and the FDM solution was 9.81×10^{-2}. This metric confirms a strong consistency and agreement between these two distinct numerical approaches in approximating the Fisher–KPP solution. The proximity of their results, even if individually deviating from the exact solution by different margins, underscores the robustness of the underlying physical model and the validity of both solution methods. This consistency is a powerful testament that PINNs are not merely an academic exercise but a viable, competitive, and often advantageous alternative to classical numerical solvers for PDEs, particularly when considering their versatility.

The current retraining strategy employed a simple learning rate reduction with optimizer reset. More sophisticated approaches could enhance PINN adaptability, such as:

(i) Learning rate warm-up: Gradually increasing the learning rate during initial retraining iterations could stabilize gradient updates.
(ii) Cyclical learning rates: Periodic learning rate oscillations might help escape local minima in the loss landscape.
(iii) Gradient clipping: Limiting gradient magnitudes during retraining could prevent drastic parameter updates that degrade previously learned features.
(iv) Elastic weight consolidation: Incorporating elastic weight consolidation regularization could preserve important weights while allowing adaptation to new training phases.
(v) Optimizer state preservation: Consistent with our observations, saving and restoring optimizer internal states (Adam's m and v estimates) could enable true continuous training without performance degradation.

These strategies warrant systematic investigation to establish robust retraining protocols for scientific machine learning applications.

5.1. Applications of the Fisher–KPP equation

The Fisher–KPP equation, due to its ability to model reaction–diffusion processes, can be applied in various scientific and engineering disciplines:

(i) Biology and population dynamics: The Fisher–KPP equation is a cornerstone in ecological modeling. It is used to understand the spatial spread of species, the growth of bacterial colonies, and the propagation of epidemics or disease within a population.[12]
(ii) Neuroscience and central nervous system disease modeling: In neuroscience applications, the Fisher–KPP equation has been successfully used to model the spatiotemporal progression of neurodegenerative diseases, such as Alzheimer's, Parkinson's, and prion diseases. Here, $u(x, t)$ represents the concentration of misfolded proteins (e.g., amyloid-β, tau).[13] The reaction term models the autocatalytic conversion of healthy proteins into pathological forms, while the diffusion term captures their spread through neural tissue. Traveling wave solutions of the Fisher–KPP equation effectively represent the advancing front of disease pathology in the brain, providing a mathematical framework to understand disease propagation patterns.
(iii) Ecology and environmental science: Beyond simple population growth, the Fisher–KPP equation models the invasion of non-native species, the spread of pollutants in ecosystems, and the formation of spatial patterns in ecological communities.
(iv) Chemistry and combustion: In chemistry, the Fisher–KPP equation describes the propagation of chemical reaction fronts, such as in autocatalytic reactions. In combustion theory, the equation models the propagation of flame fronts, where the reaction term represents the combustion rate and the diffusion term represents heat transfer.
(v) Neuroscience: Variants of the Fisher–KPP equation are used to model the propagation of nerve impulses along axons, where the reaction term represents ion channel dynamics, and the diffusion term represents ion diffusion.
(vi) Epidemiology: The Fisher–KPP equation can be adapted to model the spatial spread of infectious diseases, considering both the rate of infection (reaction) and the movement of individuals (diffusion).
(vii) Finance: In mathematical finance, the Fisher–KPP equation appears in models of option pricing and risk management, particularly in contexts involving nonlinear feedback or spatial correlations.

The versatility and fundamental nature of the Fisher–KPP equation make it a crucial benchmark for

new computational methods and a powerful tool for understanding diverse real-world phenomena.

5.2. Future work

Looking forward, several promising avenues for future research emerge from this study, aimed at overcoming current limitations and expanding the utility of PINNs, including:

(i) Optimizer state preservation for retraining: A crucial practical improvement would involve implementing mechanisms to rigorously save and load the optimizer's state (e.g., Adam's first and second moment estimates) and the model's parameters. This would enable seamless continuation of training and better leverage prior learning, potentially recovering or exceeding the initial optimal accuracy during retraining, especially for models deployed in dynamic or evolving environments where continuous learning is necessary.

(ii) Adaptive weighting schemes for loss components: Exploring more sophisticated adaptive weighting strategies for the loss function components (L_{IC}, L_{BC}, L_{Res}) could significantly enhance training stability and convergence speed. Inspired by recent advancements in I-PINN,1 [14] dynamically adjusting these weights based on the magnitude of each loss term or its gradients could help address gradient pathologies and ensure a balanced contribution from all physics constraints throughout training. This is particularly important for complex PDEs where different terms might have vastly different scales.

(iii) Advanced architectures and activation functions: Investigating the impact of different neural network architectures (e.g., deeper and wider networks, or incorporating concepts like Fourier features for better high frequency approximation [15]), and alternative activation functions (e.g., sigmoid linear unit, Gaussian error linear unit, or learnable activations) could lead to further improvements in accuracy and convergence for this class of nonlinear problems. Recurrent neural network components or convolutional layers might also be explored for spatio-temporal dependencies.

(iv) Inverse problems and parameter discovery: Beyond forward solving, PINNs are inherently capable of addressing inverse problems, such as discovering unknown physical parameters (e.g., D or R) directly from limited and potentially noisy observational data. Applying this framework to infer such parameters in the Fisher–KPP equation would represent a compelling and practical next step, demonstrating the full potential of physics-informed machine learning for scientific discovery.

(v) Uncertainty quantification: Integrating techniques for uncertainty quantification within the PINN framework could provide valuable insights into the reliability and confidence of the predicted solutions, especially in scenarios with noisy data, sparse observations, or inherent model uncertainties. This could involve Bayesian PINNs or ensemble methods.

(vi) High-dimensional and multi-physics problems: Extending the current 1D study to higher dimensions or to coupled multi-physics systems (e.g., two coupled reaction–diffusion equations) would be a natural progression. PINNs offer a significant advantage here due to their mesh-free nature, which mitigates the "curse of dimensionality" that plagues traditional grid-based methods.

(vii) Comparison with other data-driven methods: A comprehensive comparative study with other emerging data-driven methods for PDEs, such as DeepONets or Fourier neural operators, could further contextualize the strengths and weaknesses of PINNs for specific problem classes.

In conclusion, this work reinforces the position of PINNs as a powerful and flexible tool for scientific computing, capable of handling complex nonlinear PDEs with impressive accuracy. While challenges related to training dynamics, particularly retraining, persist, the continuous evolution of PINN methodologies holds immense promise for advancing our ability to model, predict, and understand intricate physical phenomena across various domains.

Acknowledgments

None.

Funding

None.

Conflict of interest

The authors declare no conflicts of interest.

Author contributions

Conceptualization: All authors
Formal analysis: Ahmed Aberqi
Investigation: Ahmed Miloudi
Methodology: All authors
Writing–original draft: Ahmed Miloudi
Writing–review & editing: Ahmed Miloudi

Availability of data

The data and code supporting the findings of this study are available from the corresponding author upon reasonable request.

AI Tools Statement

All authors confirm that no AI tools were used in the preparation of this manuscript.

References

1. LeVeque RJ. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM; 2007.
2. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Vol 9; May 13–15, 2010; Chia Laguna Resort, Sardinia, Italy; 2010:249-256.
3. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving

forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys*. 2019;378:686-707. https://www.doi.org/10.1016/j.jcp.2018.10.045.

4. Kolmogorov AN, Petrovsky IG, Piskunov NS. A study of the diffusion equation with increase in the amount of substance, and its application to a biological problem. *Mosc Univ Math Bull*. 1937;1(6)1-26.
5. Fisher RA. The wave of advance of advantageous genes. *Ann Eugen*. 1937;7(4):355-369. https://www.doi.org/10.1111/j.1469-1809.1937.tb02153.x.
6. Karniadakis GE, Kevrekidis IG, Lu L, et al. Physics-informed machine learning. *Nat Rev Phys*. 2021;3:422-440. https://www.doi.org/10.1038/s42254-021-00314-5.
7. Lu L, Meng X, Mao Z, Karniadakis GE. DeepXDE: a deep learning library for solving differential equations. *SIAM Rev*. 2021;63(1):208-228. https://www.doi.org/10.1137/19M1274067.
8. Pang G, Lu L, Karniadakis GE. fpinns: fractional physics-informed neural networks. *SIAM J Sci Comput*. 2019;41(4):A2603-A2626. https://www.doi.org/10.1137/18M1229845.
9. Byrd RH, Lu P, Nocedal J, Zhu C. A limited memory algorithm for bound-constrained optimization. *SIAM J Sci Comput*. 1995;16(5):1190-1208. https://www.doi.org/10.1137/0916069.
10. Wang S, Teng Y, Perdikaris P. Understanding and mitigating gradient pathologies in physics-informed neural networks. *SIAM J Sci Comput*. 2021;43(5):A3055-A3081. https://www.doi.org/10.1137/20M1318043.
11. Tancik M, Srinivasan P, Mildenhall B, et al. Fourier features let networks learn high frequency functions in low dimensional domains. In: *Advances in Neural Information Processing Systems*. 2020;33. arXiv preprint. https://arxiv.org/abs/2006.10739. Accessed March 22, 2025
12. Weickenmeier J, Kurt M, Ozkaya E, et al. Brain stiffens post mortem. *J Mech Behav Biomed Mater*. 2018;84:88-98. https://www.doi.org/10.1016/j.jmbbm.2018.04.009.
13. Fornari S, Schäfer A, Jucker M, Goriely A, Kuhl E. Prion-like spreading of Alzheimer's disease within the brain's connectome. *J R Soc Interface*. 2019;16(159):20190356. https://www.doi.org/10.1098/rsif.2019.0356.
14. Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv preprint. https://arxiv.org/abs/1412.6980. Accessed July 14, 2025.
15. Hornik K. Approximation capabilities of multilayer feedforward networks. *Neural Netw*. 1991;4(2):251-257. https://www.doi.org/10.1016/0893-6080(91)90009-T.