Big Data and Information Analytics

https://www.aimspress.com/journal/BDIA

BDIA, 10: 391–408.
DOI:10.3934/bdia.2026020
Received: 02 July 2026
Revised: 01 November 2026
Accepted: 06 November 2026
Published: 19 November 2026

Research article

Physics-informed neural networks for reaction-diffusion systems: a comparative study with numerical integration of the FitzHugh–Nagumo equations

Chao Li[1,2], **Xiyuan Xiao**[1,3], **Yan Du**[1,*] **and Lingxi Xia**[1]

[1] School of Mathematics, Southwest Jiaotong University, Chengdu 611756, China

[2] Rizhao Central Vitality Zone Development Service Center, Rizhao 276800, China

[3] Rizhao Sanlian Water Transfer Co., Ltd., Rizhao 276800, China

* **Correspondence:** Email: dy20021220@my.swjtu.edu.cn; Tel: +86-156-8690-2656.

Abstract: This study employs physics-informed neural networks (PINNs) and the explicit Euler, second-order Runge–Kutta (RK2) and fourth-order Runge–Kutta (RK4) methods to solve the two-dimensional FitzHugh–Nagumo (FHN) reaction-diffusion system with periodic boundary conditions. The PINN combines the governing equations, the initial condition, and sparse samples of the RK4 solution in its loss function, and all methods are evaluated against a high-accuracy pseudo-spectral reference solution on test points that are not used for training. The results show that the PINN reproduces the spatiotemporal evolution of the system at grid and off-grid points. Owing to the physics loss, the PINN achieves lower errors than RK4 for the excitation variable and maintains its accuracy with very sparse data, although it requires much longer training and is less accurate for the recovery variable. It is therefore a promising complement to classical solvers, particularly when data are sparse.

Keywords: physics-informed neural networks; reaction-diffusion equations; FitzHugh–Nagumo system; Runge–Kutta methods; error analysis

1. Introduction

Reaction-diffusion equations serve as fundamental mathematical frameworks for modeling a wide spectrum of ubiquitous spatiotemporal dynamical phenomena in nature, with extensive applications spanning biology, chemistry, ecology, physics, and other interrelated disciplines [1]. In particular, the FitzHugh–Nagumo (FHN) system, as a prototypical reaction-diffusion model, is frequently employed to describe excitation–recovery dynamics, including nerve impulse propagation, cardiac electrophysiological activity, and spontaneous pattern formation. Specifically, FitzHugh [2] originally

proposed this simplified model to characterize physiological excitation states in nerve membranes, and Nagumo et al. [3] subsequently constructed an active pulse transmission line circuit to simulate nerve axon conduction. The numerical solution and qualitative analysis of this system have long constituted active research frontiers in nonlinear dynamics and computational science [4].

Conventional numerical approaches for partial differential equations (PDEs)—such as the finite difference method [5] and the finite element method [6]—approximate continuous problems through spatiotemporal discretization, underpinned by well-established mathematical foundations and validated by a broad range of practical applications. Among these, time-stepping schemes including the Euler method, the second-order Runge–Kutta (RK2) method, and the fourth-order Runge–Kutta (RK4) method are routinely employed in the numerical simulation of reaction-diffusion systems, serving as workhorses in computational practice. Nevertheless, these methods are inherently contingent upon mesh generation, provide the solution only at the grid nodes, and become costly or inflexible for high-dimensional problems, irregular domains, or scattered observations [7].

With the rapid development of deep learning, PINNs, which combine physical laws with data-driven learning, have become a new approach for solving PDEs [8]. By embedding the governing equations and the initial and boundary conditions into the loss function of a neural network, PINNs impose the physical laws as soft constraints and represent the solution as a continuous, mesh-free function of space and time. They have shown great potential in fluid mechanics [9], solid mechanics [10], biomedicine [11], and other fields. Since the loss function can combine physical residuals with sparse data, PINNs are also suitable for reconstructing continuous fields from discretely sampled data [12].

However, standard PINNs have clear limitations for strongly nonlinear, multiscale reaction-diffusion systems such as the FHN model, mainly for two reasons. First, they are usually built on fully connected networks, which have difficulty representing steep gradients and fine local structures: Krishnapriyan et al. [13] demonstrated that standard PINNs can fail to learn complex evolutionary PDE dynamics, while Wang et al. [14] showed that gradient flow pathologies can make training ill-conditioned for stiff or multi-scale problems. Second, the initial and boundary conditions are imposed only as soft penalty terms in the loss function, so the accuracy depends on the choice of the loss weights, and these conditions are in general satisfied only approximately. For the FHN system, Ren et al. [15] reported that standard PINNs fail to capture most of the dynamical patterns and yield much larger errors than their convolutional-recurrent physics-informed network, and hard-coded physical structures [16] have since been proposed to mitigate these issues. Moreover, the training loss is often reported as evidence of accuracy, although a small weighted loss does not guarantee a small error.

A comparison of PINNs with the Euler and Runge–Kutta methods for an epidemic reaction-diffusion model was recently reported in [17]. Against this background, this study compares a PINN with the Euler, RK2, and RK4 methods for the two-dimensional FHN system with periodic boundary conditions. The PINN imposes periodicity exactly through a Fourier feature embedding introduced by Dong and Ni [18], selects collocation points via a temporal curriculum inspired by the time-marching curriculum training of Krishnapriyan et al. [13] and the temporal causality principle of Wang et al. [19], together with the residual-based adaptive sampling of Wu et al. [20], and is trained on sparse samples of a numerical solution alongside the physical residual. All methods are evaluated against a high-accuracy reference solution on independent test points, and each loss component is reported separately from the actual error. The main contributions are a rigorous evaluation of the PINN against classical time integration, an analysis of where the physics loss improves on the training data and where it does not,

and ablation and data-sparsity studies that clarify the roles of the PINN components and of the data.

The remainder of this paper is organized as follows. Section 2 details the reaction-diffusion system and the numerical techniques employed—specifically, the finite difference method, Euler's method, the second- and fourth-order Runge–Kutta schemes, and the PINN, together with the reference solution, data sets, and error metrics. Numerical experiments conducted on the FHN system are reported in Section 3. Finally, Section 4 summarizes the principal conclusions and outlines potential directions for future research.

2. Methods

This section is concerned with the following dimensionless reaction-diffusion system:

$$\begin{cases} u_t(t,\mathbf{x}) = D_u \Delta u(t,\mathbf{x}) + \mathcal{F}(u,v), \quad \mathbf{x} \in \Omega, \quad t \in [0,T], \\ v_t(t,\mathbf{x}) = D_v \Delta v(t,\mathbf{x}) + \mathcal{G}(u,v), \quad \mathbf{x} \in \Omega, \quad t \in [0,T], \\ \mathcal{B}(u) = b_1(t,\mathbf{x}), \quad \mathcal{B}(v) = b_2(t,\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \quad t \in [0,T], \\ \mathcal{I}(u) = i_1(\mathbf{x}), \quad \mathcal{I}(v) = i_2(\mathbf{x}), \quad \mathbf{x} \in \bar{\Omega}, \quad t = 0, \end{cases} \tag{2.1}$$

where $u(t,\mathbf{x})$ and $v(t,\mathbf{x})$ are the unknown solutions to be determined and $u_t(t,\mathbf{x})$, $v_t(t,\mathbf{x})$ denote the time derivatives. D_u and D_v are positive diffusion coefficients for u and v, respectively. Δ is the Laplace operator, i.e., $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2$. $D_u\Delta u(t,\mathbf{x})$ and $D_v\Delta v(t,\mathbf{x})$ are diffusion terms, and $\mathcal{F}(u,v)$ and $\mathcal{G}(u,v)$ are reaction terms, also known as source terms. $\mathcal{B}(\cdot)$ is the boundary operator, and $b_1(t,\mathbf{x})$, $b_2(t,\mathbf{x})$ are boundary conditions. $\mathcal{I}(\cdot)$ is the initial operator, $i_1(\mathbf{x})$, $i_2(\mathbf{x})$ are initial conditions, Ω is the computational domain, and $\partial\Omega$ is its boundary. In this work, $\Omega = [-L, L)^2$ and periodic boundary conditions are imposed, i.e., u, v and their derivatives take equal values on opposite sides of Ω.

We proceed to solve the above reaction-diffusion system in the subsequent sections, beginning with a description of several numerical schemes and followed by a detailed discussion of physics-informed neural networks.

2.1. Finite difference method

The finite difference method serves as a fundamental technique for approximating PDEs by discrete algebraic systems; for instance, Garvie [21] developed finite-difference schemes for reaction-diffusion predator-prey models, while Arif et al. [22] formulated a hybrid finite-difference approach for diffusive epidemic systems. The periodic domain $\Omega = [-L, L)^2$ is discretized by $N \times N$ uniformly spaced nodes

$$x_i = -L + i\Delta x, \qquad y_j = -L + j\Delta y, \qquad i, j = 0, 1, \ldots, N-1, \qquad \Delta x = \Delta y = h = \frac{2L}{N}. \tag{2.2}$$

Because of periodicity, the points $x = L$ and $y = L$ coincide with $x = -L$ and $y = -L$; hence the $N \times N$ nodes carry all unknowns, and node indices are understood modulo N (e.g., $u_{-1,j} = u_{N-1,j}$ and $u_{N,j} = u_{0,j}$). The Laplacian is approximated by the standard five-point stencil

$$(\mathbf{L}_h u)_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}, \tag{2.3}$$

which applies to every node, so that no special boundary or corner formulas are needed; Δv is discretized in the same way. This yields the semi-discrete system

$$\frac{\mathrm{d}u_{i,j}}{\mathrm{d}t} = D_u(\mathbf{L}_h u)_{i,j} + \mathcal{F}(u_{i,j}, v_{i,j}), \qquad \frac{\mathrm{d}v_{i,j}}{\mathrm{d}t} = D_v(\mathbf{L}_h v)_{i,j} + \mathcal{G}(u_{i,j}, v_{i,j}), \tag{2.4}$$

with $u_{i,j}(0) = i_1(x_i, y_j)$ and $v_{i,j}(0) = i_2(x_i, y_j)$. Writing $\mathbf{U} = (u_{i,j}, v_{i,j})$ and $\mathbf{R}(\mathbf{U})$ for the right-hand side of (2.4), the system is advanced with a uniform time step Δt by the three explicit schemes below.

2.2. Euler method

The (forward) Euler method advances the semi-discrete system (2.4) by

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t\, \mathbf{R}(\mathbf{U}^n), \tag{2.5}$$

i.e., $u_{i,j}^{n+1} = u_{i,j}^n + \Delta t\, [D_u(\mathbf{L}_h u^n)_{i,j} + \mathcal{F}(u_{i,j}^n, v_{i,j}^n)]$ and analogously for v. It is first-order accurate in time. In contrast to the explicit update used here, Madzvamuse and Chung [23] investigated fully implicit time-stepping schemes and nonlinear solvers for reaction-diffusion systems.

2.3. Second-order Runge–Kutta method

Semi-implicit alternatives have also been developed for stiff problems. Boscarino et al. [24] constructed semi-implicit Runge–Kutta schemes for time-dependent PDEs, while Nie et al. [25] treated linear diffusion exactly and nonlinear reactions implicitly. Here, we use the explicit RK2 method, also known as the improved Euler or Heun method. A predictor is obtained by an explicit Euler step, and the solution is then updated using the average of the slopes at the initial and predicted states:

$$\tilde{\mathbf{U}} = \mathbf{U}^n + \Delta t\, \mathbf{R}(\mathbf{U}^n), \qquad \mathbf{U}^{n+1} = \mathbf{U}^n + \frac{\Delta t}{2}\left[\mathbf{R}(\mathbf{U}^n) + \mathbf{R}(\tilde{\mathbf{U}})\right]. \tag{2.6}$$

2.4. Fourth-order Runge–Kutta method

Salah et al. [26] combined differential quadrature for spatial discretization with RK4 time integration to solve a (3 + 1)-dimensional unsteady gas-flow system. Other approaches to nonlinear dynamical systems include the homotopy perturbation method, which Kelleci and Yıldırım [27] applied to models of lactic acid fermentation and epidemic dynamics. In this study, we use the classical four-stage RK4 method to advance the semi-discrete system in time. The update formula is

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \frac{\Delta t}{6}\left(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4\right), \tag{2.7}$$

with the stage values

$$\mathbf{K}_1 = \mathbf{R}(\mathbf{U}^n), \quad \mathbf{K}_2 = \mathbf{R}\big(\mathbf{U}^n + \tfrac{\Delta t}{2}\mathbf{K}_1\big), \quad \mathbf{K}_3 = \mathbf{R}\big(\mathbf{U}^n + \tfrac{\Delta t}{2}\mathbf{K}_2\big), \quad \mathbf{K}_4 = \mathbf{R}\big(\mathbf{U}^n + \Delta t\, \mathbf{K}_3\big), \tag{2.8}$$

where each $\mathbf{K}_s$ contains both the diffusion and the reaction contributions of u and v.

2.5. PINNs

PINNs combine data-driven learning with physical constraints by incorporating the governing equations and the initial and boundary conditions into a loss function. Recent extensions include the conversion of PINNs to spiking neural networks by [28] and the use of Kolmogorov–Arnold networks within a physics-informed framework for incompressible flows by [29]. In this study, a fully connected PINN is trained using both the physical residuals and sparse samples of a numerical solution.

We use a fully connected neural network $NN(\mathbf{x}, t; \theta)$, which maps the space-time coordinates $(\mathbf{x}, t)$ to the predictions $\hat{u}$ and $\hat{v}$; θ denotes the trainable weights and biases. The structure of the PINN is shown in Figure 1. Before entering the first layer, the coordinates are transformed by the Fourier feature embedding, referred to below as the periodic embedding,

$$\boldsymbol{\gamma}(t,x,y) = \Big[\tilde{t},\ \cos(k_m x),\ \sin(k_m x),\ \cos(k_m y),\ \sin(k_m y)\Big]_{m=1}^{M_f}, \qquad \tilde{t} = \frac{2t}{T} - 1, \quad k_m = \frac{2\pi m}{P}, \tag{2.9}$$

where $P = 2L$ is the period of the domain. Every function of $\boldsymbol{\gamma}$ is exactly P-periodic in x and y together with all of its derivatives, so the periodic boundary conditions are satisfied by construction [18]. The forward propagation is defined as

$$\begin{cases} h_0 = \sigma_1(W_0\boldsymbol{\gamma} + b_0), \\ h_l' = \sigma_1(W_l h_{l-1} + b_l), \quad l = 1, 2, \ldots, n-2, \\ h_l = \alpha_1 h_l' + \alpha_2 h_{l-1}, \quad l = 1, 2, \ldots, n-2, \ \alpha_1 + \alpha_2 = 1, \end{cases} \tag{2.10}$$

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = W_{n-1} h_{n-2} + b_{n-1}, \tag{2.11}$$

where $\boldsymbol{\gamma}$ is the feature vector (2.9), W_l and b_l are the weight matrices and bias vectors, σ_1 is the activation function, l is the hidden-layer index, n is the total number of layers, and α_1, α_2 are the weights of the soft residual connections, which stabilize the training of the deep network.

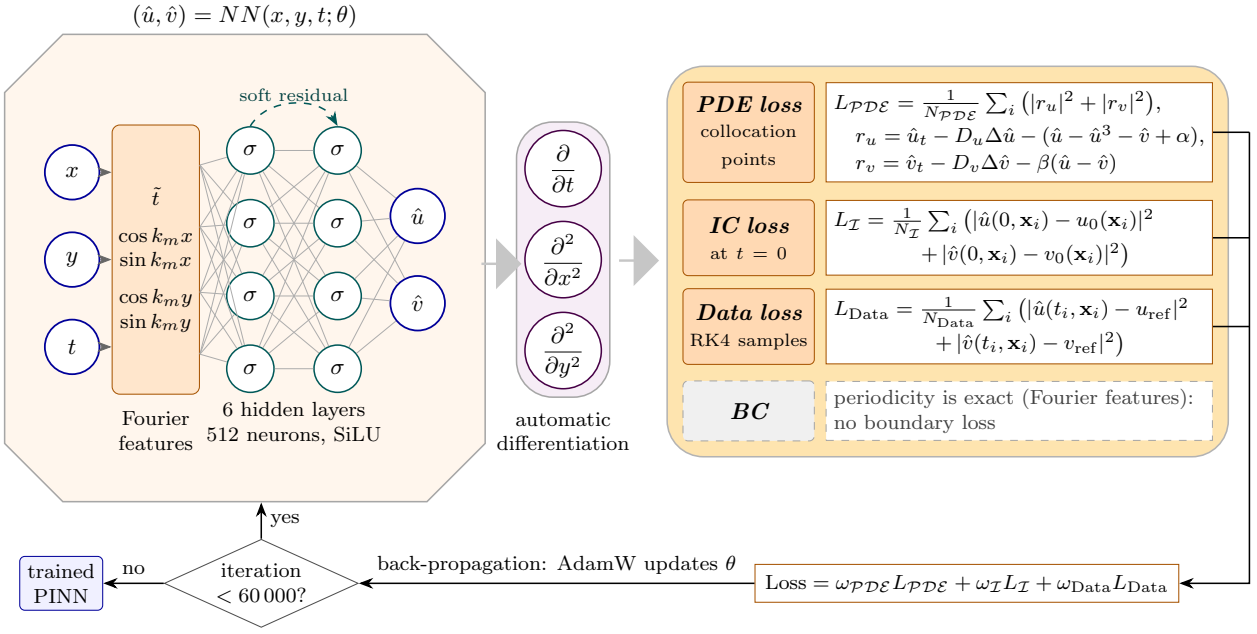


Figure 1. Architecture of the PINN with periodic feature embedding.

The total loss function is defined as

$$\text{Loss} = \omega_{\mathcal{PDE}} L_{\mathcal{PDE}} + \omega_{\mathcal{B}} L_{\mathcal{B}} + \omega_{\mathcal{I}} L_{\mathcal{I}} + \omega_{\text{Data}} L_{\text{Data}}, \tag{2.12}$$

$$\begin{aligned} L_{\mathcal{PDE}} = \frac{1}{N_{\mathcal{PDE}}} \sum_{i=1}^{N_{\mathcal{PDE}}} \Big[& \ell\big(\hat{u}_t(t_i, \mathbf{x}_i) - D_u \Delta \hat{u}(t_i, \mathbf{x}_i) - \mathcal{F}(\hat{u}, \hat{v})\big) \\ & + \ell\big(\hat{v}_t(t_i, \mathbf{x}_i) - D_v \Delta \hat{v}(t_i, \mathbf{x}_i) - \mathcal{G}(\hat{u}, \hat{v})\big) \Big], \end{aligned} \tag{2.13}$$

$$L_{\mathcal{B}} = \frac{1}{N_{\mathcal{B}}} \sum_{i=1}^{N_{\mathcal{B}}} \Big[\ell\big(\mathcal{B}(\hat{u}_i) - b_1(t_i, \mathbf{x}_i)\big) + \ell\big(\mathcal{B}(\hat{v}_i) - b_2(t_i, \mathbf{x}_i)\big) \Big], \tag{2.14}$$

$$L_{\mathcal{I}} = \frac{1}{N_{\mathcal{I}}} \sum_{i=1}^{N_{\mathcal{I}}} \Big[\ell\big(\mathcal{I}(\hat{u}_i) - i_1(\mathbf{x}_i)\big) + \ell\big(\mathcal{I}(\hat{v}_i) - i_2(\mathbf{x}_i)\big) \Big], \tag{2.15}$$

$$L_{\text{Data}} = \frac{1}{N_{\text{Data}}} \sum_{i=1}^{N_{\text{Data}}} \Big[\ell\big(\hat{u}(t_i, \mathbf{x}_i) - u_{\text{ref}}(t_i, \mathbf{x}_i)\big) + \ell\big(\hat{v}(t_i, \mathbf{x}_i) - v_{\text{ref}}(t_i, \mathbf{x}_i)\big) \Big], \tag{2.16}$$

where $\hat{u}, \hat{v}$ are the network predictions, $\omega_{\mathcal{PDE}}, \omega_{\mathcal{B}}, \omega_{\mathcal{I}}$, and ω_{Data} are the loss weights, $N_{\mathcal{PDE}}$, $N_{\mathcal{B}}$, $N_{\mathcal{I}}$, and N_{Data} are the numbers of collocation, boundary, initial, and data points used in each iteration, and $\ell(z) = z^2$, so that each term is a mean squared error. We refer to $L_{\mathcal{PDE}}$, $L_{\mathcal{B}}$, $L_{\mathcal{I}}$, and L_{Data} as the PDE loss (or physics loss), the boundary loss, the initial-condition (IC) loss, and the data loss, respectively. The derivatives in (2.13) are obtained by automatic differentiation of the network output with respect to t, x, and y. Because of the periodic embedding, the boundary loss vanishes identically and $\omega_{\mathcal{B}} = 0$; $L_{\mathcal{B}}$ is used only in the ablation variant without embedding. In (2.16), $u_{\text{ref}}, v_{\text{ref}}$ are the supervised samples of the RK4 solution (Section 2.6), and the loss weights are fixed during training. The network parameters are updated by back-propagation with the AdamW optimizer; the training settings are described in Section 3.3.

At every iteration, $N_{\mathcal{PDE}}$ collocation points are sampled in $[0, T_c] \times \Omega$. Following a temporal curriculum strategy, the time window T_c is increased from $0.5T$ to $0.8T$ after one quarter of the iterations and to T after half of them. After the first third of the training, half of the collocation points are sampled using the residual-based adaptive distribution (RAD) method proposed by Wu et al. [20] from $N_c = 10^5$ uniformly distributed candidate points, which are refreshed every 200 iterations and selected with probability

$$p_i \propto \frac{|r_i|}{\frac{1}{N_c}\sum_{j=1}^{N_c} |r_j|} + 1, \qquad |r_i| = \big(r_u(t_i, \mathbf{x}_i)^2 + r_v(t_i, \mathbf{x}_i)^2\big)^{1/2}, \tag{2.17}$$

where r_u, r_v are the residuals in (2.13); the other half is sampled uniformly.

2.6. Reference solution, data sets, and error metrics

The errors of all methods are measured against a reference solution computed by a Fourier pseudo-spectral method on an 80×80 grid, combined with the exponential time-differencing fourth-order Runge–Kutta (ETDRK4) integrator with $\Delta t = 2 \times 10^{-3}$. Cox and Matthews [30] developed exponential

time-differencing schemes, including ETDRK4, for systems with stiff linear terms. Kassam and Trefethen [31] subsequently improved the numerical evaluation of the ETDRK4 coefficients using contour integrals to avoid cancellation errors. Since the initial data are given only at the 20×20 nodes, the continuous initial field is defined as their trigonometric interpolant. Doubling the resolution or halving the time step changes the reference solution by less than 5.6×10^{-12}, so its error is negligible. The reference solution is used only for evaluation.

The supervised data are taken from the RK4 solution at all grid nodes and time levels, excluding $t = 0$ (used only in $L_{\mathcal{I}}$) and the output times $t = 0.2, 0.4, \ldots, 12$, which are reserved for testing. From this data pool, 2000 points are held out as a validation set that is used only to monitor training, and a fixed random fraction ρ of the remaining points forms the training set. Two test sets are used: $\mathcal{T}_1$ consists of the 20×20 nodes at the 60 output times, and $\mathcal{T}_2$ of the nodes of the 80×80 reference grid that are not grid nodes, at the same times; $\mathcal{T}_2$ tests the prediction between the grid nodes.

For $w \in \{u, v\}$ and a test set $\mathcal{T}$, we report the relative L^2 error and the maximum absolute error of the approximation $\hat{w}$ (PINN or finite-difference solution) with respect to the reference solution w_{ex},

$$\varepsilon_w = \frac{\left(\sum_{(t,\mathbf{x})\in\mathcal{T}} |\hat{w}(t,\mathbf{x}) - w_{\mathrm{ex}}(t,\mathbf{x})|^2 \right)^{1/2}}{\left(\sum_{(t,\mathbf{x})\in\mathcal{T}} |w_{\mathrm{ex}}(t,\mathbf{x})|^2 \right)^{1/2}}, \qquad e_{\max,w} = \max_{(t,\mathbf{x})\in\mathcal{T}} |\hat{w}(t,\mathbf{x}) - w_{\mathrm{ex}}(t,\mathbf{x})|. \tag{2.18}$$

Errors on $\mathcal{T}_2$ are marked by the superscript "off" (e.g., $\varepsilon_u^{\mathrm{off}}$), and $\varepsilon_w(t)$ denotes the error at a single output time t. For the finite-difference schemes, values at off-grid points are obtained by trigonometric interpolation.

3. Numerical results

3.1. FitzHugh–Nagumo equation

The FHN system studied in this paper is given by

$$\begin{cases} \frac{\partial u}{\partial t} = D_u \Delta u + u - u^3 - v + \alpha, \quad x, y \in \Omega,\ t \in [0, T], \\ \frac{\partial v}{\partial t} = D_v \Delta v + \beta(u - v), \quad x, y \in \Omega,\ t \in [0, T], \\ u(x, y, 0) = u_0(x, y), \quad v(x, y, 0) = v_0(x, y), \quad x, y \in \bar{\Omega}, \\ u(t, -L, y) = u(t, L, y),\ u(t, x, -L) = u(t, x, L), \quad x, y \in \partial\Omega, \\ v(t, -L, y) = v(t, L, y),\ v(t, x, -L) = v(t, x, L), \quad x, y \in \partial\Omega, \end{cases} \tag{3.1}$$

where u is the fast excitation variable (e.g., membrane potential), v is the slow recovery variable, D_u, D_v are diffusion coefficients, $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2$, α is an external stimulus, and β controls the recovery rate. Periodic boundary conditions are used to simulate wave propagation in an extended domain.

The parameters are $D_u = 1$, $D_v = 10$, $\alpha = 0.01$, $\beta = 0.25$, $L = 10$, and $T = 12$. The initial values at the 20×20 grid nodes are independent samples of $\mathcal{N}(0, 0.05^2)$. The Euler, RK2, and RK4 methods use $N = 20$ nodes per direction, i.e., $\Delta x = \Delta y = 1$ and $\Delta t = 0.01$. Using the absolute-stability framework described by [32], the time-step bounds for the discretized diffusion operator on the present grid are $\Delta t \le 0.025$ for Euler and RK2 and approximately $\Delta t \le 0.0348$ for RK4. The chosen time step $\Delta t = 0.01$ satisfies these diffusion-only stability bounds.

The spatial mean of u increases from about 0 to 0.72 at $t \approx 6$ and then decays. Figure 3 shows that the random initial perturbations are smoothed by diffusion and organize into large-scale patterns.

3.2. *Convergence of the finite-difference schemes*

Tables 1 and 2 and Figure 2 show the temporal and spatial convergence of the finite-difference schemes, measured by the relative L^2 error of u and v combined. In the temporal study, $\Delta x = 1$ is fixed, and the errors are computed with respect to a solution with $\Delta t = 1.25 \times 10^{-4}$. In the spatial study, RK4 is used with $\Delta t = 0.002$ for $\Delta x \geq 0.25$ and $\Delta t = 5 \times 10^{-4}$ for $\Delta x = 0.125$, which satisfies the stability condition on every grid, and the errors are computed with respect to the pseudo-spectral reference at the 20×20 nodes. The observed orders are 1.00, 2.01, and 4.15 in time and 2.02 in space, in agreement with theory.

Table 1. Temporal convergence of the finite-difference schemes.

Δt	Euler error	RK2 error	RK4 error	Euler order	RK2 order	RK4 order
0.02	6.31×10^{-3}	5.48×10^{-5}	1.71×10^{-7}	–	–	–
0.01	3.16×10^{-3}	8.80×10^{-6}	7.95×10^{-9}	1.00	2.64	4.43
0.005	1.58×10^{-3}	2.19×10^{-6}	4.34×10^{-10}	1.00	2.01	4.20
0.0025	7.90×10^{-4}	5.46×10^{-7}	2.54×10^{-11}	1.00	2.00	4.10

Table 2. Spatial convergence of the five-point finite-difference discretization.

Δx	RK4 error	RK4 order
1	4.33×10^{-3}	–
0.5	9.63×10^{-4}	2.17
0.25	2.35×10^{-4}	2.04
0.125	5.83×10^{-5}	2.01

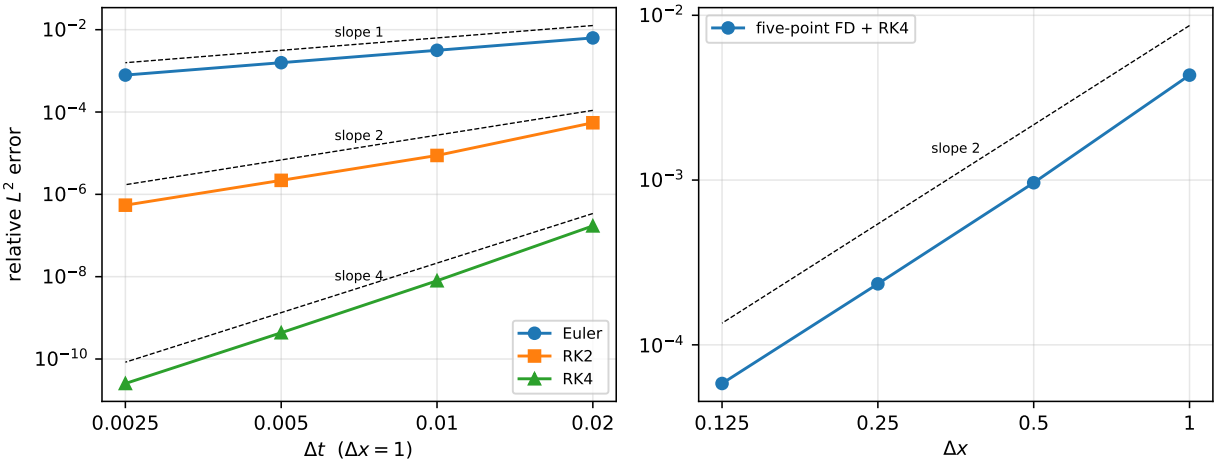


Figure 2. Temporal (left) and spatial (right) convergence of the finite-difference schemes; dashed lines indicate the theoretical slopes.

With $\Delta t = 0.01$, the temporal errors of RK2 and RK4 are negligible compared with the spatial discretization error, so that both methods have practically the same total error in Table 4. For the Euler method, the first-order temporal error is of the same order as the spatial error, so that its error is

about 30% larger than that of RK4 for u and about four times larger for v. As shown in Figure 5, the relative errors of all three schemes are largest at early times, when the solution consists of grid-scale random perturbations, which the five-point stencil damps too slowly; they decrease as these modes are damped and grow again toward $t = 12$, when the amplitude of u decreases.

3.3. PINN configuration

Table 3 summarizes the PINN configuration. The network consists of an input layer that maps the 41 features to 512 neurons, $n - 2 = 6$ hidden layers with 512 neurons each and a linear output layer; the input and hidden layers use the sigmoid linear unit (SiLU) activation function, and the soft residual weights are $\alpha_1 = 0.8$ and $\alpha_2 = 0.2$. The network weights are initialized by the Kaiming normal method, and the biases are set to zero. Miyato et al. [33] introduced spectral normalization as a weight-normalization technique to stabilize discriminator training in generative adversarial networks. Here, we apply spectral normalization to the first, third, and fifth hidden layers of the PINN to promote stable training.

All PINN variants are trained for 60,000 iterations. As shown in Figure 7, the test error decreases until about 50,000 iterations, after which its relative standard deviation is at most 1.9%, so a larger budget would not change the results noticeably. Training the PINN takes 163 min on a GPU, whereas one RK4 simulation takes 0.34 s on a CPU; once trained, the PINN evaluates the solution at arbitrary points within a fraction of a second.

Table 3. Configuration of the PINN.

Item	Setting
Network	periodic embedding (2.9) with $M_f = 10$ (41 inputs) $\to$ 512 $\to$ 6 soft-residual layers (512, SiLU) $\to$ 2 outputs; 1,598,466 parameters
Loss weights	$\omega_{\mathcal{PDE}} = 1$, $\omega_{\mathcal{I}} = 10$, $\omega_{\text{Data}} = 3$, $\omega_{\mathcal{B}} = 0$ (fixed)
Optimizer	AdamW (weight decay 10^{-6}), learning rate $10^{-3} \to 10^{-6}$ (cosine annealing), gradient clipping (norm 1), 60,000 iterations, single precision (FP32)
Collocation points	$N_{\mathcal{PDE}} = 20,000$ per iteration, resampled at every iteration; temporal curriculum; RAD for 50% of the points after 20,000 iterations (candidate set 10^5, refreshed every 200 iterations)
Initial-condition points	all $20 \times 20 = 400$ nodes at $t = 0$
Boundary points	none (exact periodicity); 1000 per iteration only in the ablation without embedding
Supervised data	data pool of 456,000 RK4 values; $\rho = 10\%$ of the 454,000 values left after removing the validation set $\Rightarrow$ 45,400 fixed training samples; mini-batch of 20,000 samples per iteration
Validation set	2000 held-out RK4 values (monitoring only)
Test set $\mathcal{T}_1$	24,000 points (20×20 nodes $\times$ 60 output times), compared with the reference and with RK4
Test set $\mathcal{T}_2$	360,000 off-grid points ($80 \times 80 - 20 \times 20$ points $\times$ 60 output times)
Repetitions	3 random seeds for the main configuration, one seed for each ablation variant
Hardware	one NVIDIA GeForce RTX 5090 GPU, PyTorch 2.8 (CUDA 12.8)

3.4. *Comparison with the finite-difference schemes*

Table 4 lists the errors of all methods on the test sets. The PINN errors at grid and off-grid points are almost identical, and their relative standard deviation over the three random seeds is about 1% for u and 12% for v. For u, the PINN error is about 23% lower than that of RK4, and its maximum error is less than half that of RK4, although the PINN is trained on samples of the RK4 solution. For v, in contrast, the PINN is less accurate than RK2 and RK4, although more accurate than the Euler method.

Table 4. Errors of the different methods with respect to the reference solution (notation as in Section 2.6); PINN values are mean ± standard deviation over three random seeds.

Method	ε_u	ε_v	$e_{\max,u}$	$\varepsilon_u^{\text{off}}$	$\varepsilon_v^{\text{off}}$	Time(s)
Euler	6.72×10^{-3}	3.09×10^{-3}	3.15×10^{-2}	6.60×10^{-3}	3.09×10^{-3}	0.08 (CPU)
RK2	5.24×10^{-3}	7.56×10^{-4}	3.25×10^{-2}	5.08×10^{-3}	7.56×10^{-4}	0.16 (CPU)
RK4	5.24×10^{-3}	7.51×10^{-4}	3.25×10^{-2}	5.08×10^{-3}	7.51×10^{-4}	0.34 (CPU)
PINN (3 seeds)	$(4.01 \pm 0.05) \times 10^{-3}$	$(2.01 \pm 0.24) \times 10^{-3}$	$(1.51 \pm 0.04) \times 10^{-2}$	$(4.01 \pm 0.05) \times 10^{-3}$	$(2.01 \pm 0.24) \times 10^{-3}$	9783 (GPU)

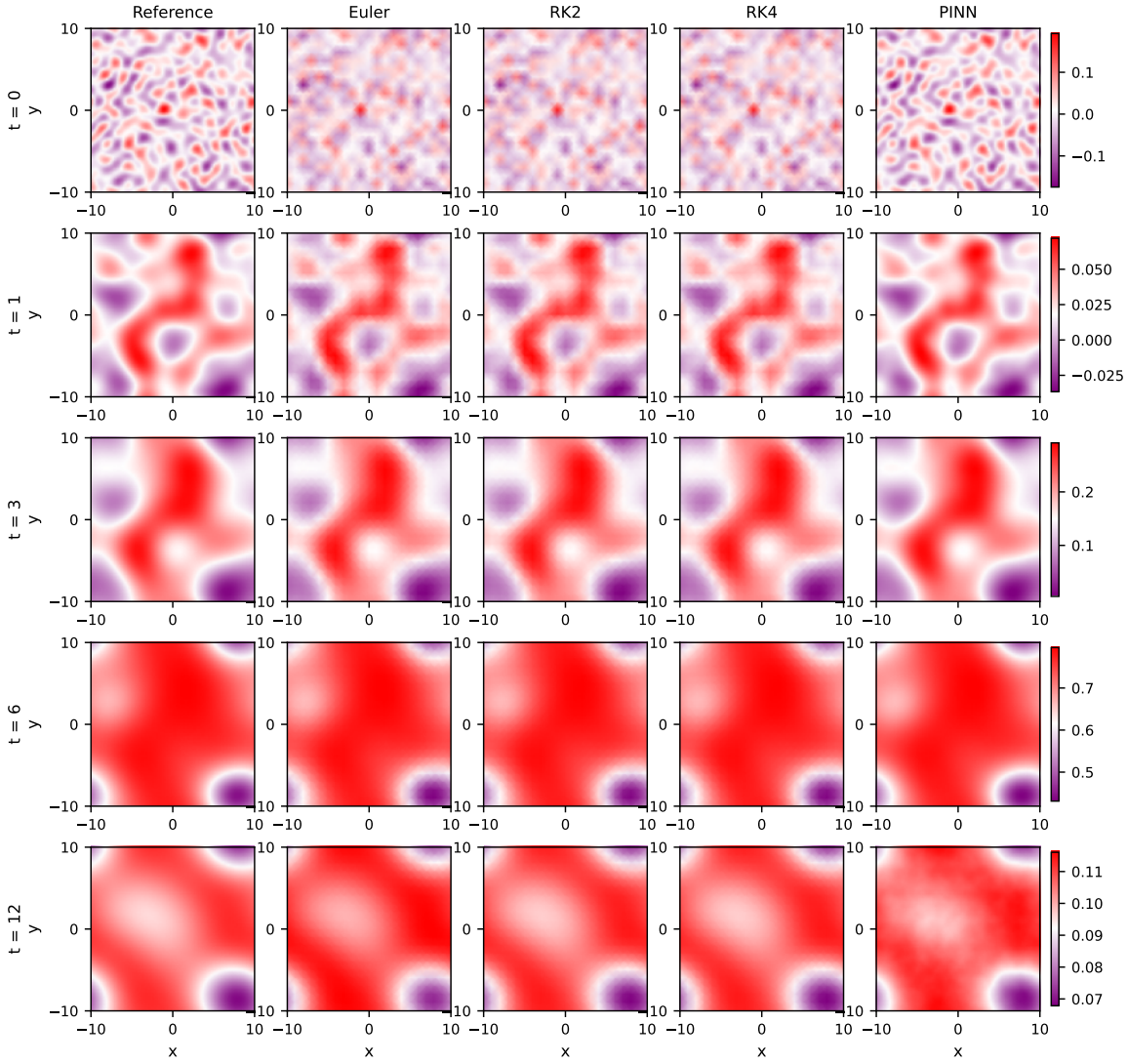


Figure 3. Spatiotemporal evolution of the excitation variable u obtained by different methods.

As shown in Figure 3, all methods reproduce the evolution of the pattern, and at early times the PINN also reproduces the smooth reference field between the grid nodes. Figure 4 shows that at $t = 1$, the RK4 error consists of grid-scale spots, whereas the PINN error is smaller and smoother; at $t = 6$, the two error maps are very similar; and at $t = 12$, when the amplitude of the pattern is small, the PINN error contains small-scale fluctuations. The large errors near the lower edge at $t = 6$ also occur for RK4 and are therefore caused by the local structure of the solution rather than by the boundary treatment of the PINN.

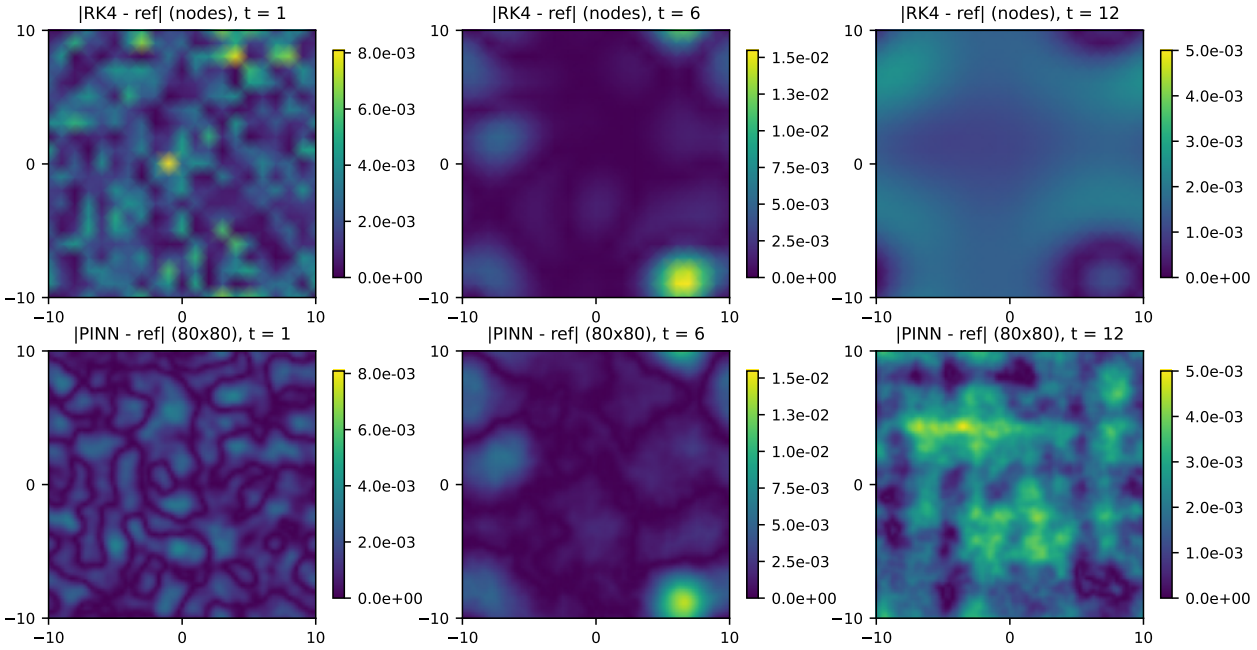


Figure 4. Absolute errors of u for RK4 (top) and the PINN (bottom).

Figure 5 shows where the PINN gains its accuracy. In the initial transient ($0 < t \leq 2$), the error of the PINN is about one third of that of RK4. For $t > 2$, the two errors are almost equal, and for $t > 8$ the PINN error is slightly larger. The PINN therefore does not show a superior long-time behavior. The time series in Figure 6 cannot be distinguished at the scale of the plot.

The reason is the consistency of the training data with the governing equations. Since the RK4 data satisfy the semi-discrete system (2.4), their residual in the exact PDE (3.1) is $D_u(\mathbf{L}_h - \Delta)u$ for u and $D_v(\mathbf{L}_h - \Delta)v$ for v. This residual is large in the transient, when the solution contains grid-scale components (root mean square (RMS) value 1.93×10^{-1} for $t \leq 0.5$), but below 10^{-3} for $t > 2$, far below the residual level of the trained PINN (1.48×10^{-2}). The physics loss can therefore correct the data only in the transient; afterward the PINN follows the data and inherits their error, which explains the nearly identical error maps at $t = 6$. Accordingly, a network trained on the same data without the physics loss (dotted line in Figure 5) has the same error as RK4 in the transient.

For v, the PINN is less accurate than RK2 and RK4 at all times. Its relative error is largest in the transient, when v is very small, and afterward remains about twice that of RK4.

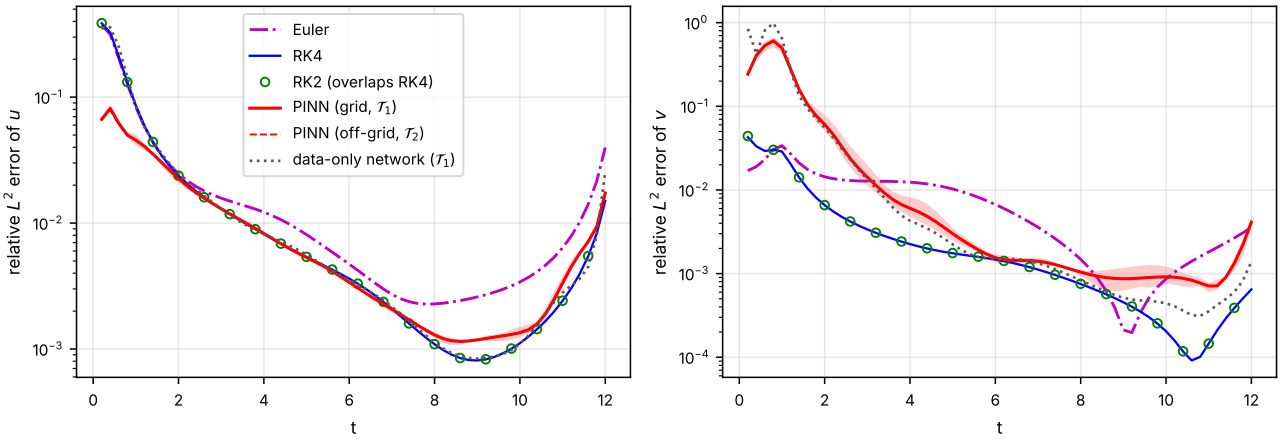


Figure 5. Relative L^2 errors of u (left) and v (right) versus time; the shaded band shows the range over three random seeds.

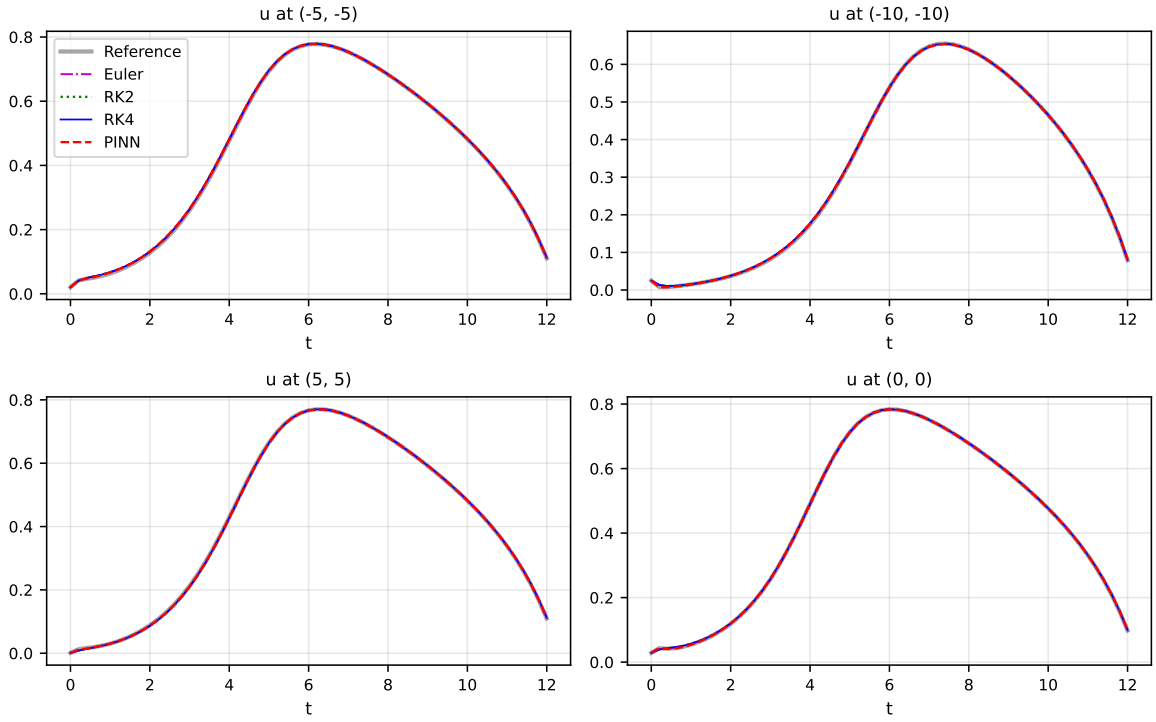


Figure 6. Temporal evolution of u at four spatial points.

3.5. Training loss and actual error

Since the total loss (2.12) is a weighted sum of terms with different magnitudes, its value is not a measure of accuracy. Figure 7 therefore shows each unweighted loss component together with the test and validation errors. The loss values and the error are not in a fixed relation: the first two curriculum stages, the losses are already small while the error over the whole interval is still large, because the later times have not yet been trained; toward the end of the training, the PDE loss fluctuates strongly, because the collocation points are resampled at every iteration, while the error remains constant. Moreover,

the physics-only variant in Section 3.6 reaches a residual as small as that of the full method but fails completely. All accuracy statements in this paper are therefore based on the error metrics (2.18).

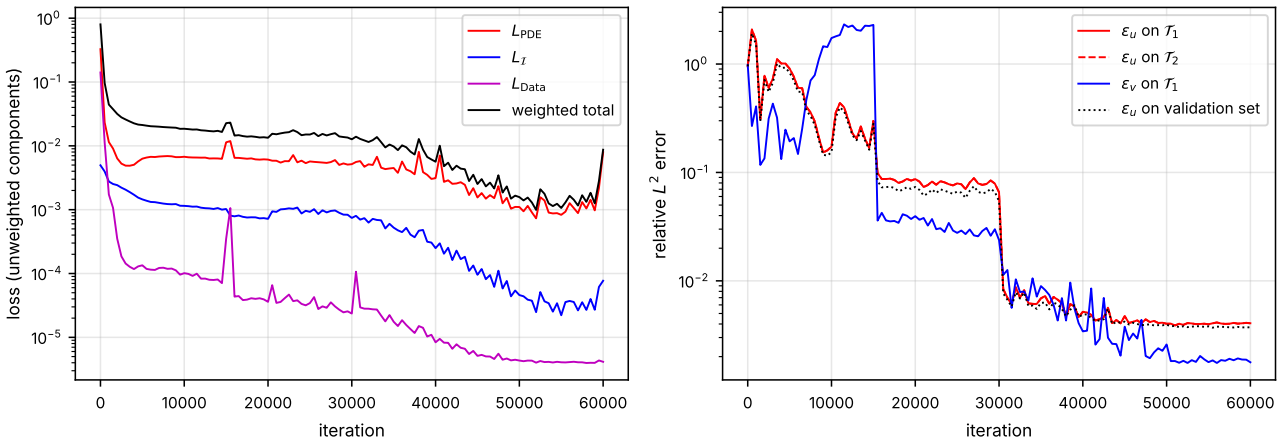


Figure 7. Training history of the PINN: loss components (left) and errors (right).

3.6. Ablation study

Table 5 lists variants in which exactly one component is removed while all other settings are unchanged; without the periodic embedding, periodicity is imposed by a soft boundary loss instead. Each variant was trained with the same single seed as the full method, so differences smaller than the variability between seeds in Table 4, about 1% for u and 12% for v, are not significant.

Table 5. Results of the ablation study.

Variant	ε_u	ε_v	$\varepsilon_u^{\mathrm{off}}$	PDE residual (RMS)	Train (min)
Full method	4.06×10^{-3}	1.77×10^{-3}	4.05×10^{-3}	1.55×10^{-2}	163
w/o periodic embedding (soft BC loss)	4.02×10^{-3}	9.91×10^{-4}	4.05×10^{-3}	6.58×10^{-3}	195
w/o temporal curriculum	4.00×10^{-3}	1.95×10^{-3}	4.00×10^{-3}	1.51×10^{-2}	163
w/o residual-based adaptive sampling	3.97×10^{-3}	1.58×10^{-3}	3.96×10^{-3}	1.40×10^{-2}	162
w/o soft residual connections	4.48×10^{-3}	2.93×10^{-3}	4.49×10^{-3}	1.75×10^{-2}	148
w/o PDE loss (data-only network)	5.34×10^{-3}	2.77×10^{-3}	1.10×10^{-1}	1.19×10^{1}	17
w/o data loss (physics-only PINN)	1.00	9.83×10^{-1}	1.00	1.40×10^{-2}	157

Removing the periodic embedding, the temporal curriculum or the adaptive sampling does not reduce the accuracy. The variant with the soft boundary loss is even more accurate for v, but it requires longer training and satisfies periodicity only approximately; the benefits of the embedding in this problem are thus exactness by construction and efficiency rather than accuracy. The curriculum is not needed because the supervised data cover the whole time interval, and the adaptive sampling concentrates points in regions of large residual without improving the overall accuracy. In contrast, removing the soft residual connections increases the errors of u and v by about 10% and 65%, respectively.

The last two variants each remove one source of information. The data-only network fits the training data at the grid nodes but develops grid-aligned oscillations between them, so that its off-grid error is about 0.1. The physics-only PINN fails completely: it decays to a spurious state of small

amplitude and misses the excitation of the medium, although its residual is as small as that of the full method (because for $u \approx v \approx 0$ the residual of the u-equation is only $-\alpha = -0.01$). This failure to propagate the information of a small initial perturbation is a known failure mode of PINNs [13, 19, 34]. Specifically, Krishnapriyan et al. [13] demonstrated that standard PINNs can fail to propagate initial-condition dynamics forward in time, Wang et al. [19] showed that violating temporal causality causes optimization to converge to spurious low-residual states, and Daw et al. [34] analyzed such propagation failures and proposed evolutionary sampling strategies to mitigate them.

3.7. Effect of data sparsity

Table 6 and Figure 8 compare the PINN with the data-only network for fractions ρ of supervised data between 100% and 0.1%. The PINN error is practically independent of ρ, even with only 454 samples, whereas the error of the data-only network grows sharply below $\rho = 1\%$ and is about 45 times larger than that of the PINN at $\rho = 0.1\%$; between the grid nodes, its error is 0.1 or larger for all ρ. A few hundred samples thus suffice to select the correct solution, which the physics loss extends to the whole space-time domain; the failure of the physics-only variant shows, however, that some data are indispensable.

Table 6. Errors of the PINN and the data-only network for different fractions of supervised data.

ρ	Training samples	PINN ε_u	Data-only ε_u	PINN $\varepsilon_u^{\text{off}}$	Data-only $\varepsilon_u^{\text{off}}$
1	454,000	4.00×10^{-3}	5.29×10^{-3}	3.99×10^{-3}	1.13×10^{-1}
0.1	45,400	4.06×10^{-3}	5.34×10^{-3}	4.05×10^{-3}	1.10×10^{-1}
0.01	4540	4.02×10^{-3}	8.06×10^{-3}	4.02×10^{-3}	1.05×10^{-1}
0.001	454	4.25×10^{-3}	1.90×10^{-1}	4.25×10^{-3}	2.05×10^{-1}

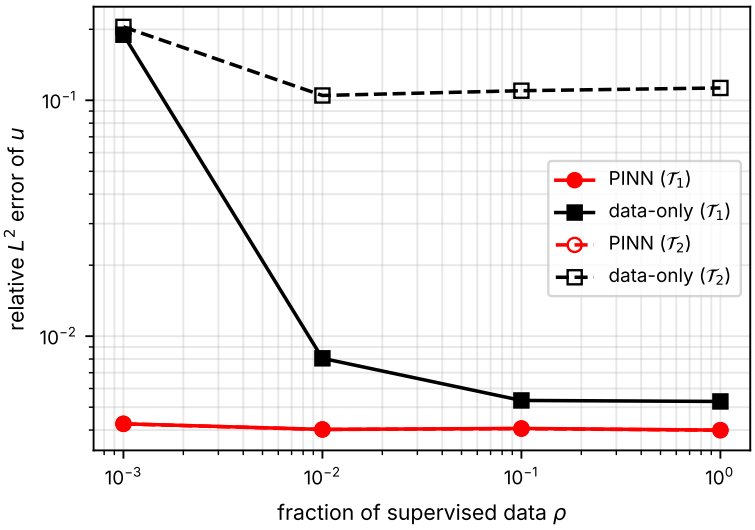


Figure 8. Relative L^2 error of u versus the fraction of supervised data.

4. Conclusions

This study compared a PINN with the Euler, RK2, and RK4 methods for the two-dimensional FHN system, measuring accuracy against a high-accuracy reference solution on independent test points rather than by the training loss. The finite-difference schemes attain their theoretical convergence orders, and the error of RK2 and RK4 is dominated by the spatial discretization.

The results highlight three advantages of the PINN. First, the physics loss corrects the discretization error of the training data, so that the PINN is more accurate than RK4 for the excitation variable, although it is trained on RK4 samples. Second, the PINN represents the solution as a continuous function that is as accurate between the grid nodes as at the nodes and, once trained, can be evaluated at arbitrary points within a fraction of a second, without any interpolation. Third, the PINN remains accurate with very sparse data, down to 0.1% of the samples, whereas a purely data-driven network fails between the grid nodes.

These advantages come with limitations: The gain is confined to the initial transient, the recovery variable is less accurate than with RK2 and RK4, some supervised data are required, and the training is far more expensive. The PINN is therefore best used as a complement to classical solvers, for the continuous reconstruction of solutions from sparse data and for inverse problems.

The study is limited to one parameter set and one IC. Although PINNs show competitive accuracy and potential in this study, challenges remain for real-world applications. Future work may proceed in several directions: Methodologically, training efficiency and stability can be improved through advanced network architectures, dynamic loss balancing [14], and accelerated optimization algorithms. In terms of applications, PINNs can be extended to high-dimensional, complex-geometry, and multiscale coupled systems, and to inverse problems such as the identification of the FHN parameters from sparse observations. Theoretically, convergence analysis, error estimation, and uncertainty quantification can be strengthened to provide more rigorous support. Semi-supervised or unsupervised learning paradigms can also be explored for low-data scenarios. Hybrid frameworks combining PINNs with traditional numerical methods can be developed to integrate global continuity with computational efficiency.

Use of AI tools declaration

The authors declare that Artificial Intelligence (AI) tools were used in the preparation of this article. Claude was used to assist with language editing, formatting and reference organization. All experiments were run by the authors, who checked all content and take full responsibility for this article.

Acknowledgments

The authors received no financial support for this research.

Conflict of interest

The authors declare that there are no conflicts of interest.

References

1. Polyanin AD, Sorokin VG, Vyazmin AV, (2018) Reaction-diffusion models with delay: Some properties, equations, problems, and solutions. *Theor Found Chem Eng* 52: 334–348. https://doi.org/10.1134/S0040579518030132
2. FitzHugh R, (1961) Impulses and physiological states in theoretical models of nerve membrane. *Biophys J* 1: 445–466. https://doi.org/10.1016/S0006-3495(61)86902-6
3. Nagumo J, Arimoto S, Yoshizawa S, (1962) An active pulse transmission line simulating nerve axon. *Proc IRE* 50: 2061–2070. https://doi.org/10.1109/JRPROC.1962.288235
4. Ilati M, Dehghan M, (2017) Application of direct meshless local Petrov-Galerkin (DMLPG) method for some Turing-type models. *Eng Comput* 33: 107–124. https://doi.org/10.1007/s00366-016-0458-x
5. Nishiyama H, Noi T, (2016) Conservative difference schemes for the numerical solution of the Gardner equation. *Comput Appl Math* 35: 75–95. https://doi.org/10.1007/s40314-014-0183-2
6. Kaiser MW, Fries TP, (2025) Simultaneous solution of incompressible Navier-Stokes flows on multiple surfaces. *Arch Appl Mech* 95: 233. https://doi.org/10.1007/s00419-025-02935-z
7. DiBenedetto E, Gianazza U, (2023) *Partial Differential Equations*, 3rd Ed., Cham: Birkhäuser. https://doi.org/10.1007/978-3-031-46618-2
8. Raissi M, Perdikaris P, Karniadakis GE, (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys* 378: 686–707. https://doi.org/10.1016/j.jcp.2018.10.045
9. Hu B, McDaniel D, (2023) Applying physics-informed neural networks to solve Navier-Stokes equations for laminar flow around a particle. *Math Comput Appl* 28: 102. https://doi.org/10.3390/mca28050102
10. Bai J, Rabczuk T, Gupta A, Alzubaidi L, Gu Y, (2023) A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics. *Comput Mech* 71: 543–562. https://doi.org/10.1007/s00466-022-02252-0
11. Sarabian M, Babaee H, Laksari K, (2022) Physics-informed neural networks for brain hemodynamic predictions using medical imaging. *IEEE Trans Med Imaging* 41: 2285–2303. https://doi.org/10.1109/TMI.2022.3161653
12. Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L, (2021) Physics-informed machine learning. *Nat Rev Phys* 3: 422–440. https://doi.org/10.1038/s42254-021-00314-5
13. Krishnapriyan AS, Gholami A, Zhe S, Kirby RM, Mahoney MW, (2021) Characterizing possible failure modes in physics-informed neural networks. *Adv Neural Inf Process Syst* 34: 26548–26560..
14. Wang S, Teng Y, Perdikaris P, (2021) Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM J Sci Comput* 43: A3055–A3081. https://doi.org/10.1137/20M1318043
15. Ren P, Rao C, Liu Y, Wang JX, Sun H, (2022) PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. *Comput Methods Appl Mech Eng* 389: 114399. https://doi.org/10.1016/j.cma.2021.114399

16. Rao C, Ren P, Wang Q, Buyukozturk O, Sun H, Liu Y, (2023) Encoding physics to learn reaction-diffusion processes. *Nat Mach Intell* 5: 765–779. https://doi.org/10.1038/s42256-023-00685-7

17. Xia L, Han B, (2025) Comparative study of PINNs and numerical methods in infectious disease reaction-diffusion models. *Big Data Inf Anal* 9: 265–284. https://doi.org/10.3934/bdia.2025012

18. Dong S, Ni N, (2021) A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *J Comput Phys* 435: 110242. https://doi.org/10.1016/j.jcp.2021.110242

19. Wang S, Sankaran S, Perdikaris P, (2024) Respecting causality for training physics-informed neural networks. *Comput Methods Appl Mech Eng* 421: 116813. https://doi.org/10.1016/j.cma.2024.116813

20. Wu C, Zhu M, Tan Q, Kartha Y, Lu L, (2023) A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Comput Methods Appl Mech Eng* 403: 115671. https://doi.org/10.1016/j.cma.2022.115671

21. Garvie MR, (2007) Finite-difference schemes for reaction-diffusion equations modeling predator-prey interactions in MATLAB. *Bull Math Biol* 69: 931–956. https://doi.org/10.1007/s11538-006-9062-3

22. Arif MS, Abodayeh K, Nawaz Y, (2025) A hybrid finite difference approach for solving fuzzy stochastic SIR-β model with diffusion and incidence rate. *Eur J Pure Appl Math* 18: 6292. https://doi.org/10.29020/nybg.ejpam.v18i3.6292

23. Madzvamuse A, Chung AHW, (2014) Fully implicit time-stepping schemes and non-linear solvers for systems of reaction-diffusion equations. *Appl Math Comput* 244: 361–374. https://doi.org/10.1016/j.amc.2014.07.004

24. Boscarino S, Filbet F, Russo G, (2016) High order semi-implicit schemes for time dependent partial differential equations. *J Sci Comput* 68: 975–1001. https://doi.org/10.1007/s10915-016-0168-y

25. Nie Q, Zhang YT, Zhao R, (2006) Efficient semi-implicit schemes for stiff systems. *J Comput Phys* 214: 521–537. https://doi.org/10.1016/j.jcp.2005.09.030

26. Salah M, Matbuly MS, Civalek O, Ragb O, (2024) Calculation of four-dimensional unsteady gas flow via different quadrature schemes and Runge–Kutta 4th order method. *Adv Appl Math Mech* 16: 437–458. https://doi.org/10.4208/aamm.OA-2021-0373

27. Kelleci A, Yıldırım A, (2011) Numerical solution of the system of nonlinear ordinary differential equations arising in kinetic modeling of lactic acid fermentation and epidemic model. *Int J Numer Methods Biomed Eng* 27: 585–594. https://doi.org/10.1002/cnm.1321

28. Zhang Q, Wu C, Kahana A, Karniadakis GE, Kim Y, Li Y, et al. (2025) Artificial to spiking neural networks conversion with calibration in scientific machine learning. *SIAM J Sci Comput* 47: C559–C577. https://doi.org/10.1137/24M1643232

29. Jiang Q, Gou Z, (2025) Solutions to two- and three-dimensional incompressible flow fields leveraging a physics-informed deep learning framework and Kolmogorov-Arnold networks. *Int J Numer Methods Fluids* 97: 665–673. https://doi.org/10.1002/fld.5374

30. Cox SM, Matthews PC, (2002) Exponential time differencing for stiff systems. *J Comput Phys* 176: 430–455. https://doi.org/10.1006/jcph.2002.6995

31. Kassam AK, Trefethen LN, (2005) Fourth-order time-stepping for stiff PDEs. *SIAM J Sci Comput* 26: 1214–1233. https://doi.org/10.1137/S1064827502410633

32. LeVeque RJ, (2007) *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, Philadelphia: SIAM. https://doi.org/10.1137/1.9780898717839

33. Miyato T, Kataoka T, Koyama M, Yoshida Y, (2018) Spectral normalization for generative adversarial networks. In: *International Conference on Learning Representations (ICLR)*.

34. Daw A, Bu J, Wang S, Perdikaris P, Karpatne A, (2023) Mitigating propagation failures in physics-informed neural networks using retain-resample-release (R3) sampling. In: *Proceedings of the 40th International Conference on Machine Learning*, 202: 7264–7302.



AIMS Press

© 2026 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0)