Big Data and Information Analytics

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

BDIA, 9: 265–284.
DOI:10.3934/bdia.2025012
Received: 28 October 2025
Revised: 21 November 2025
Accepted: 24 November 2025
Published: 28 November 2025

Research article

Comparative study of PINNs and numerical methods in infectious disease reaction-diffusion models

Lingxi Xia and Bangsheng Han*

School of Mathematics, Southwest Jiaotong University, Chengdu 611756, China

* **Correspondence:** Email: hanbangsheng@swjtu.edu.cn.

Abstract: This study employed physics-informed neural networks (PINNs) and numerical methods to solve a two-dimensional spatial susceptible-infectious (SI) epidemic model incorporating nonlinear propagation terms and diffusion processes. By integrating governing equations, initial and boundary conditions, and sparse reference solution data, PINNs define a multi-objective loss function, forming a supervised learning framework grounded in physical mechanisms. The results demonstrated that PINNs can accurately capture the spatiotemporal evolution patterns and spatial distribution characteristics of the density field in the model. In terms of solution accuracy, the long time predictions of PINNs significantly outperformed those of the Euler method and the second-order Runge-Kutta method (RK2), showing strong agreement with the fourth-order Runge-Kutta method (RK4) benchmark solutions. This validates the effectiveness of PINNs in solving complex reaction-diffusion systems, offering an innovative approach for modeling the spatial dynamics of infectious diseases.

Keywords: physics-informed neural networks; reaction-diffusion equations; infectious disease models; Runge-Kutta methods; supervised learning

1. Introduction

In recent years, machine learning technology, with its powerful data-driven modeling and function approximation capabilities, has triggered a paradigm shift in the field of scientific computing [1]. It can establish effective surrogate models for physical systems with complex mechanisms or those that are challenging to fully describe by traditional mathematical models, offering new ideas to break through the inherent computational bottlenecks in high-dimensional problems [2].

Partial differential equations (PDEs), as core mathematical tools for describing natural phenomena, have long been the cornerstone of scientific computing. Traditional methods such as the finite element method, finite difference method, and finite volume method [3–5] have formed a mature system over

decades by discretizing the computational domain. However, these methods often involve cumbersome and time-consuming mesh generation when dealing with complex geometries or moving boundary problems [6]. More importantly, their computational costs grow exponentially with the increase in problem dimensions, leading to the so-called "curse of dimensionality" [7, 8].

To overcome these limitations, Raissi et al. [9] pioneered the proposal of PINNs in 2019 and applied them to solve both forward and inverse problems of PDEs. Subsequently, PINNs were rapidly extended and applied to a wide range of fields, including fluid mechanics [10, 11], solid mechanics [12, 13], materials science [14], and even medicine [15, 16], and gave rise to specialized software libraries such as DeepXDE [17]. These advances mark the emergence of the new paradigm of "physics-informed machine learning" [18].

Meanwhile, PINNs have demonstrated unique value in complex system modeling, particularly in the field of infectious disease dynamics. Researchers have extended them to various variant models, forming a unified framework such as disease-informed neural networks (DINNs). This framework not only enables forward simulation [19, 20] but also solves complex inverse problems to achieve optimal control of epidemic intervention strategies [21], constituting a complete methodological system from cognition to decision-making.

Classic SIR models and their variants can describe the temporal evolution of diseases in a well-mixed population but struggle to capture the profound impact of spatial heterogeneity on transmission paths and rates [22]. More importantly, during actual transmission, due to factors such as saturation effects and behavioral changes, infection terms often exhibit nonlinear characteristics. In this case, adopting a nonlinear transmission term of the form $\beta S^p I^q$ better reflects real transmission dynamics than the classic bilinear assumption [23,24]. Building on this, Sun [25] established an SI-type reaction-diffusion model to study the spatiotemporal dynamics and pattern formation of infectious diseases by simulating the spatial transmission of pathogens.

Despite the broad prospects of PINNs in scientific computing, no studies have yet applied them to solve the two-dimensional SI infectious disease reaction-diffusion model with nonlinear infection terms and spatial diffusion terms. Thus, there is no clear consensus on whether PINNs can effectively solve such complex equations. Additionally, comparative studies between PINNs and traditional numerical methods in solving this model remain lacking. Against this background, this study employs a supervised learning-based PINNs framework and three traditional numerical methods to solve the model, and conducts a comprehensive analysis and comparison from multiple perspectives, including temporal, spatial, and global errors.

The remainder of this paper is organized as follows: Section 2 presents the general form of the reaction-diffusion equation system. Section 3 details the finite difference method, Euler method, RK2 method, RK4 method, and PINNs. Section 4 provides numerical experiments on the SI model. Section 5 presents the main conclusions and future prospects.

2. Problem statement

In this section, we consider a dimensionless reaction-diffusion equation system:

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

where $u(t,\mathbf{x})$ and $v(t,\mathbf{x})$ are the unknown potential solutions, $u(t,\mathbf{x})_t$ and $v(t,\mathbf{x})_t$ denote the time derivatives, D_u and D_v are positive diffusion coefficients for u and v, Δ is the Laplacian operator, defined as $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2$, $D_u\Delta u(t,\mathbf{x})$ and $D_v\Delta v(t,\mathbf{x})$ are diffusion terms, $\mathcal{F}(u,v)$ and $\mathcal{G}(u,v)$ are reaction terms (also called source terms), $\mathcal{B}(\cdot)$ is the boundary operator for calculating boundary values, $b_1(t,\mathbf{x})$ and $b_2(t,\mathbf{x})$ are boundary conditions, $\mathcal{I}(\cdot)$ is the operator for calculating initial values, $i_1(t,\mathbf{x})$ and $i_2(t,\mathbf{x})$ are initial conditions, Ω is the computational domain, and $\partial\Omega$ is the boundary of the domain.

Subsequently, we will solve the reaction-diffusion equations of the above form, first introducing several numerical methods for solving such equations and discussing PINNs.

3. Methods

3.1. Finite difference method

The finite difference method is a core step for converting partial differential equation systems into discrete equation systems [26, 27].

Consider dividing the spatial domain into equidistant grid points with a step size h. Time is discretized into steps of Δt.

The spatial and temporal node equations are given by Eq (3.1):

$$
\begin{aligned}
&x_i = i\Delta x, \quad i = 0,1,2,\cdots,N_x,\\
&y_j = j\Delta y, \quad j = 0,1,2,\cdots,N_y,\\
&t^n = n\Delta t, \quad n = 0,1,2,\cdots,
\end{aligned}
\tag{3.1}
$$

where Δx=Δy =$h = \frac{L}{N_x} = \frac{L}{N_y}$.

Forward difference approximation is used for time derivatives:

$$
\frac{\partial u}{\partial t} \approx \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t}, \quad \frac{\partial v}{\partial t} \approx \frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta t},
\tag{3.2}
$$

Finite difference discretization is performed for the spatial Laplacian operator Δu at internal nodes ($1 \le i \le N_x - 1$, $1 \le j \le N_y - 1$), non-corner boundaries (e.g., $i = 0$, $1 \le j \le N_y - 1$), and corner points (e.g., $i = 0$, $j = 0$), as shown in Eqs (3.3)–(3.5):

$$
\Delta u_{i,j}^n = \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2},
\tag{3.3}
$$

$$
\Delta u_{0,j}^n = \frac{2u_{1,j}^n - 2u_{0,j}^n}{\Delta x^2} + \frac{u_{0,j+1}^n - 2u_{0,j}^n + u_{0,j-1}^n}{\Delta y^2},
\tag{3.4}
$$

$$
\Delta u_{0,0}^n = \frac{2u_{1,0}^n - 2u_{0,0}^n}{\Delta x^2} + \frac{2u_{0,1}^n - 2u_{0,0}^n}{\Delta y^2},
\tag{3.5}
$$

The discretization of Δv can be derived similarly. Here, i and j are spatial grid indices, and n is the time step index.

The initial conditions are discretized as:

$$\begin{aligned} u_{i,j}^0 &= i_1(0, x_i, y_j), \\ v_{i,j}^0 &= i_2(0, x_i, y_j), \end{aligned} \tag{3.6}$$

Taking Neumann boundary conditions as an example, the left boundary ($i = 0$) is discretized as:

$$u_{-1,j}^n = u_{1,j}^n. \tag{3.7}$$

The discretization for the right, bottom, and top boundaries can be derived similarly, and the boundary discretization for v is identical to that for u.

3.2. Semi-implicit Euler method

The semi-implicit Euler method employs a semi-implicit strategy of "explicit reaction terms + implicit diffusion terms" to solve Eq (2.1), converting the continuous solution of the partial differential equation in the time dimension into an iterative process of calculating future solutions based on known current solutions [28].

Equation (3.8) represents the explicit update of reaction terms:

$$\begin{aligned} \hat{u}_{i,j}^n &= u_{i,j}^n + \Delta t \cdot \mathcal{F}(u_{i,j}^n, v_{i,j}^n), \\ \hat{v}_{i,j}^n &= v_{i,j}^n + \Delta t \cdot \mathcal{G}(u_{i,j}^n, v_{i,j}^n), \end{aligned} \tag{3.8}$$

Equation (3.9) represents the implicit solution of diffusion terms:

$$\begin{aligned} u_{i,j}^{n+1} &= (\mathbf{I} - D_u \Delta t \mathbf{L})^{-1} \hat{u}_{i,j}^n, \\ v_{i,j}^{n+1} &= (\mathbf{I} - D_v \Delta t \mathbf{L})^{-1} \hat{v}_{i,j}^n. \end{aligned} \tag{3.9}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{L}$ is the Laplacian matrix incorporating boundary conditions, and the operations correspond to the discrete value vector of grid nodes (i, j).

At each time step, reaction terms are first calculated explicitly based on the current $u_{i,j}^n$ and $v_{i,j}^n$ to obtain intermediate values $\hat{u}_{i,j}^n$ and $\hat{v}_{i,j}^n$. Then, the precomputed Laplacian matrix is used to solve the diffusion terms implicitly, yielding $u_{i,j}^{n+1}$ and $v_{i,j}^{n+1}$.

3.3. Semi-implicit second-order Runge-Kutta method

The RK2 method, also known as the improved Euler method, improves the accuracy of numerical solutions through a prediction-correction mechanism [29, 30].

Prediction step for System (2.1):

$$\begin{aligned} \tilde{u}_{i,j}^n &= \left(\mathbf{I} - \frac{D_u \Delta t}{2}\mathbf{L}\right)^{-1} \left(u_{i,j}^n + \frac{\Delta t}{2}\mathcal{F}(u_{i,j}^n, v_{i,j}^n)\right), \\ \tilde{v}_{i,j}^n &= \left(\mathbf{I} - \frac{D_v \Delta t}{2}\mathbf{L}\right)^{-1} \left(v_{i,j}^n + \frac{\Delta t}{2}\mathcal{G}(u_{i,j}^n, v_{i,j}^n)\right). \end{aligned} \tag{3.10}$$

These intermediate values $\tilde{u}_{i,j}^n$ and $\tilde{v}_{i,j}^n$, obtained from the prediction step, are then utilized in the correction step to compute the reaction terms more accurately. This two-step approach enhances the overall stability and precision.

Correction step:

$$\begin{aligned} u_{i,j}^{n+1} &= (\mathbf{I} - D_u \Delta t \mathbf{L})^{-1} \left(u_{i,j}^n + \Delta t \mathcal{F}(\tilde{u}_{i,j}^n, \tilde{v}_{i,j}^n) \right), \\ v_{i,j}^{n+1} &= (\mathbf{I} - D_v \Delta t \mathbf{L})^{-1} \left(v_{i,j}^n + \Delta t \mathcal{G}(\tilde{u}_{i,j}^n, \tilde{v}_{i,j}^n) \right), \end{aligned} \tag{3.11}$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{L}$ is the Laplacian matrix incorporating boundary conditions.

In the prediction step, a half time step $\Delta t/2$ is used to calculate reaction terms explicitly based on $u_{i,j}^n$ and $v_{i,j}^n$, and implicit diffusion is combined to obtain intermediate values $\tilde{u}_{i,j}^n$ and $\tilde{v}_{i,j}^n$. In the correction step, a full time step Δt is used to calculate corrected reaction terms based on the intermediate values, and implicit diffusion is again applied to obtain $u_{i,j}^{n+1}$ and $v_{i,j}^{n+1}$.

3.4. Semi-implicit fourth-order Runge-Kutta method

The RK4 method calculates temporal evolution through four progressive stages: it estimates the initial velocity based on the current solution, estimates intermediate points based on the current velocity, optimizes step-by-step, and finally weights the four velocities according to specific weights to obtain a more accurate total update [31]. The RK4 method effectively improves computational accuracy and stability through multi-stage iteration [32].

The iteration formulas for System (2.1) are:

$$\begin{aligned} u_{i,j}^{n+1} &= u_{i,j}^n + \frac{\Delta t}{6} \left(k_{1u,i,j} + 2k_{2u,i,j} + 2k_{3u,i,j} + k_{4u,i,j} \right), \\ v_{i,j}^{n+1} &= v_{i,j}^n + \frac{\Delta t}{6} \left(k_{1v,i,j} + 2k_{2v,i,j} + 2k_{3v,i,j} + k_{4v,i,j} \right). \end{aligned} \tag{3.12}$$

The four-stage coefficients for u are:

$$\begin{aligned} k_{1u,i,j} &= \mathcal{F}(u_{i,j}^n, v_{i,j}^n), \\ \tilde{u}_{1,i,j} &= \left(\mathbf{I} - \frac{D_u \Delta t}{2} \mathbf{L} \right)^{-1} \left(u_{i,j}^n + \frac{\Delta t}{2} k_{1u,i,j} \right), \\ k_{2u,i,j} &= \mathcal{F}(\tilde{u}_{1,i,j}, \tilde{v}_{1,i,j}), \\ \tilde{u}_{2,i,j} &= \left(\mathbf{I} - \frac{D_u \Delta t}{2} \mathbf{L} \right)^{-1} \left(u_{i,j}^n + \frac{\Delta t}{2} k_{2u,i,j} \right), \\ k_{3u,i,j} &= \mathcal{F}(\tilde{u}_{2,i,j}, \tilde{v}_{2,i,j}), \\ \tilde{u}_{3,i,j} &= (\mathbf{I} - D_u \Delta t \mathbf{L})^{-1} \left(u_{i,j}^n + \Delta t k_{3u,i,j} \right), \\ k_{4u,i,j} &= \mathcal{F}(\tilde{u}_{3,i,j}, \tilde{v}_{3,i,j}). \end{aligned} \tag{3.13}$$

The four-stage coefficients for v can be derived similarly.

At all four stages, reaction terms are calculated based on current values or intermediate values, and implicit diffusion is combined to obtain transition values. Finally, weighted summation is performed to obtain $u_{i,j}^{n+1}$ and $v_{i,j}^{n+1}$.

3.5. PINNs

The PINNs are an innovative framework that integrates data-driven approaches with physical modeling. In solving partial differential equations, this framework incorporates the governing equations describing the system's physical laws, as well as initial and boundary conditions, into the neural network's loss function, while fusing high-precision numerical solutions. By minimizing this loss function, the model can learn simultaneously from limited data and known physical laws, enabling modeling and prediction of complex systems [33, 34].

A neural network $NN(\mathbf{x}, t; \theta)$ is constructed, where θ represents the set of trainable weights W and biases b. By training the neural network NN, the optimal parameters θ^* are found.

The hidden output layers during the network's forward propagation are given by Eqs (3.14)–(3.16):

$$h_0 = \sigma_1(W_0\mathbf{x} + b_0), \tag{3.14}$$

$$h_l = \sigma_2(W_{l,2}\sigma_1(W_{l,1}LN(h_{l-1}) + b_{l,1}) + b_{l,2}) \odot h_{l-1}, \tag{3.15}$$

$$\begin{aligned} u &= \alpha \cdot \psi(W_{l,1}LN(h_l) + b_{l,1}), \\ v &= \alpha \cdot \psi(W_{l,2}LN(h_l) + b_{l,2}), \end{aligned} \tag{3.16}$$

where $\mathbf{x}$ is the input vector, W and b are weights and biases, $\odot$ denotes element-wise multiplication, $l = 1, \cdots, n-1$ is the hidden layer index, σ_1, σ_2, and ψ are activation functions, LN represents layer normalization, α is a scaling coefficient, and u, v are the outputs.

To train this physics-informed neural network, the following loss function is defined:

$$Loss = \omega_{\mathcal{PDE}}L_{\mathcal{PDE}} + \omega_{\mathcal{B}}L_{\mathcal{B}} + \omega_{\mathcal{I}}L_{\mathcal{I}} + \omega_{\mathcal{D}ata}L_{\mathcal{D}ata}, \tag{3.17}$$

$$\begin{aligned} L_{\mathcal{PDE}} = \frac{1}{N_{\mathcal{PDE}}} \sum_{i=1}^{N_{\mathcal{PDE}}} &(l(\hat{u}(t_i, \mathbf{x}_i)_t - D_u\Delta\hat{u}(t_i, \mathbf{x}_i) - \mathcal{F}(\hat{u}, \hat{v})) \\ &+ l(\hat{v}(t_i, \mathbf{x}_i)_t - D_v\Delta\hat{v}(t_i, \mathbf{x}_i) - \mathcal{G}(\hat{u}, \hat{v}))) \end{aligned}, \tag{3.18}$$

$$L_{\mathcal{B}} = \frac{1}{N_{\mathcal{B}}} \sum_{i=1}^{N_{\mathcal{B}}} (l(\mathcal{B}(\hat{u}_i) - b_1(t_i, \mathbf{x}_i)) + l(\mathcal{B}(\hat{v}_i) - b_2(t_i, \mathbf{x}_i))), \tag{3.19}$$

$$L_{\mathcal{I}} = \frac{1}{N_{\mathcal{I}}} \sum_{i=1}^{N_{\mathcal{I}}} (l(\mathcal{I}(\hat{u}_i) - i_1(t_i, \mathbf{x}_i)) + l(\mathcal{I}(\hat{v}_i) - i_2(t_i, \mathbf{x}_i))), \tag{3.20}$$

$$L_{\mathcal{D}ata} = \frac{1}{N_{\mathcal{D}ata}} \sum_{i=1}^{N_{\mathcal{D}ata}} (l(\hat{u}(t_i, \mathbf{x}_i) - u_{ref}(t_i, \mathbf{x}_i)) + l(\hat{v}(t_i, \mathbf{x}_i) - v_{ref}(t_i, \mathbf{x}_i))), \tag{3.21}$$

where $\mathbf{x}$ is the input vector, $\hat{u}$ and $\hat{v}$ are the predicted solutions, and $\omega_{\mathcal{PDE}}$, $\omega_{\mathcal{B}}$, $\omega_{\mathcal{I}}$, and $\omega_{\mathcal{D}ata}$ are weight factors for different components of the loss function. $N_{\mathcal{PDE}}$, $N_{\mathcal{B}}$, $N_{\mathcal{I}}$, and $N_{\mathcal{D}ata}$ are the numbers of collocation points in the computational domain, boundary, initial domain, and reference solution, respectively. $l(\cdot)$ is a metric function, typically the L^2-norm or its variants.

4. Problem solution

4.1. SI equation

The SI model studied in this paper [25] can be expressed as the following system of equations:

$$\begin{aligned}
&\frac{\partial S}{\partial t} = A - dS - \beta S I^2 + D_1 \Delta S, \quad x, y \in \Omega, \quad t \in [0, T], \\
&\frac{\partial I}{\partial t} = \beta S I^2 - (d + \mu) I + D_2 \Delta I, \quad x, y \in \Omega, \quad t \in [0, T], \\
&S(x, y, 0) > 0, \quad I(x, y, 0) > 0, \quad x, y \in \bar{\Omega}, \quad t = 0, \\
&\left.\frac{\partial S}{\partial n}\right|_{(x,y)\in\partial\Omega} = 0, \quad \left.\frac{\partial I}{\partial n}\right|_{(x,y)\in\partial\Omega} = 0, \quad x, y \in \partial\Omega, \quad t \in [0, T].
\end{aligned} \tag{4.1}$$

where the susceptible density S and the infectious density I vary with time t in the two-dimensional space (x, y), A is the birth rate of susceptible individuals, d is the natural death rate of both susceptible and infectious individuals, β is the infection rate coefficient, μ is the disease-related death rate of infectious individuals, and D_1 and D_2 are the diffusion coefficients of susceptible and infectious individuals, respectively. $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2$ is the Laplacian operator, representing the effect of spatial diffusion. The initial conditions are $S(x, y, 0) > 0$ and $I(x, y, 0) > 0$, and the boundary conditions are Neumann boundary conditions. Ω is the spatial domain, $\partial\Omega$ is the domain boundary, and n is the unit normal vector to the boundary.

Based on the above theoretical framework, authors in [25] numerically simulated the SI model using the Euler method. The spatial domain was $[0, 100] \times [0, 100]$, with the following coefficients: $A = 1.0$, $d = 1.0$, $\beta = 42.0$, $\mu = 1.8$, $D_1 = 6.0$, $D_2 = 1.0$; the simulation duration was $T = 100$ s; in the numerical solution, the spatial step size was $h = 1.0$, and the time step size was $\Delta t = 0.01$. The simulation results are shown in Figure 1.

Figure 1 shows how different infection rates shape the final density of infectious individuals I. When $\beta = 32$, initial random perturbations gradually evolve into irregular "red-eye" [35] dot-like patterns. As $\beta = 35$, the system spontaneously forms clear, parallel stripe patterns, with alternating infected and healthy regions. With a further increase to $\beta = 40$, clear, parallel stripe patterns still form. When $\beta = 42$, the infection eventually stabilizes into uniformly scattered "blue-eye" dot-like patterns, similar to scattered epidemic outbreak points on a map. These distinct patterns are all stable states reached after long-term iteration.

From an epidemiological perspective, these morphological changes have practical epidemiological significance: stripe patterns may simulate diseases transmitted along rivers or transportation lines, such as cholera, while scattered spots resemble host aggregation areas, as seen in focal epidemics caused by rodent communities [36]. By adjusting key parameters, the model successfully reproduces the diverse spatial distributions that diseases may exhibit, providing an intuitive perspective for understanding epidemic formation [37–39].

In the remaining computational part of this paper, the spatial domain is modified to $\Omega = [0, 20] \times [0, 20]$. This approach not only enhances computational efficiency but also enables the amplified visualization of pattern structures in subsequent figures, thereby more clearly revealing their morphological details, while other parameters remain consistent with those in [25].

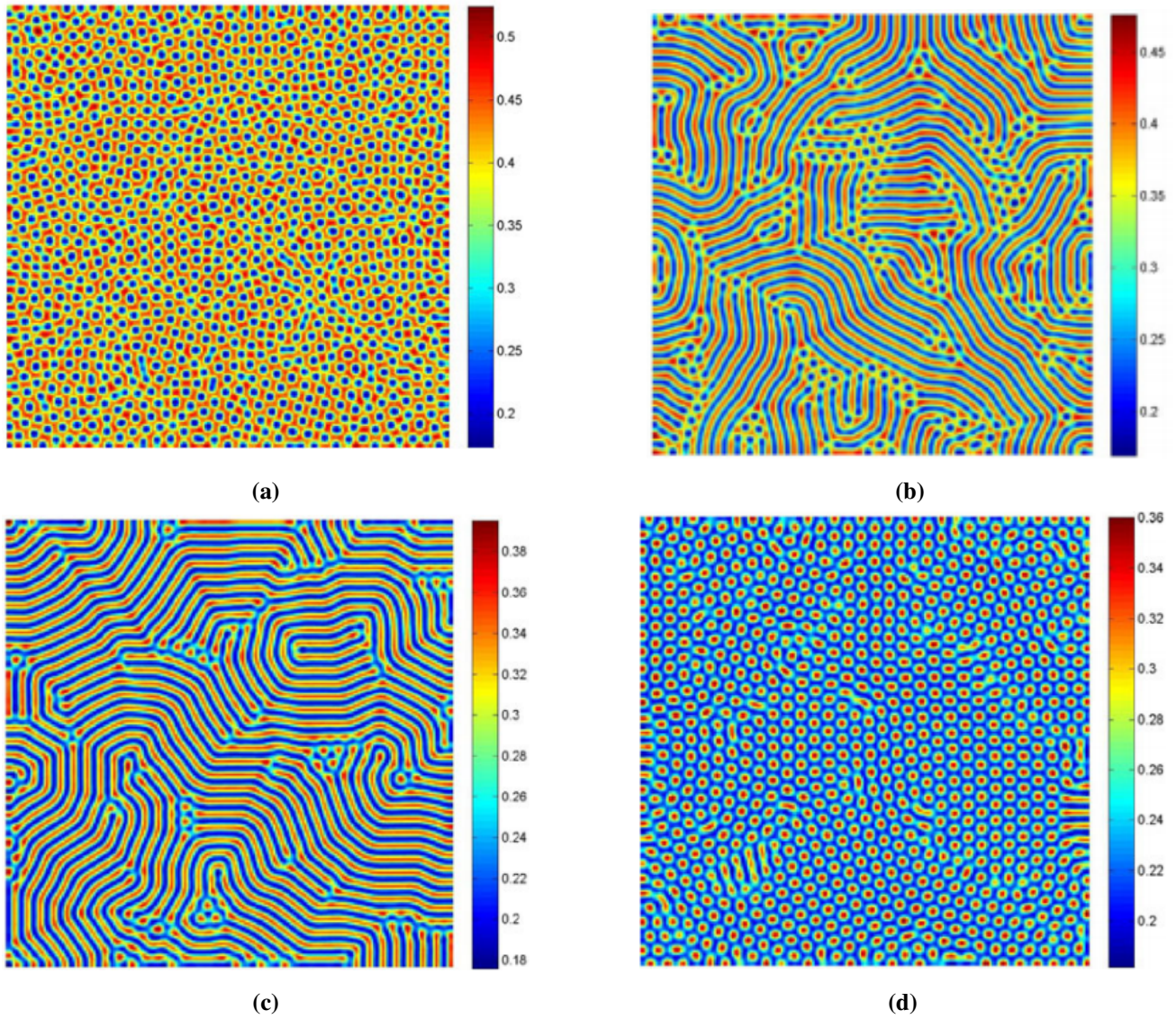


Figure 1. Final numerical solutions of the infectious density I for the SI model under $\beta = 42$, 35, 40, and 32 using the Euler numerical method.

The architecture of the physics-informed neural networks implemented in this paper is shown in Figure 2. The selection of the input layer, hidden layers, output layer, and activation function for the PINNs in the infectious disease model is shown in Figure 3.

The neural network takes coordinates $\mathbf{x} = [t, x, y]^T$ as input and outputs the densities of susceptible individuals S and infectious individuals I through hidden layers. The loss function consists of PDE loss, initial condition loss, boundary condition loss, and a data fitting term.

The neural network takes time and spatial coordinates as three-dimensional input. Its basic structure is trained with 200,000 iterations, including 9 hidden layers with 320 neurons each. On this basis, the network is further optimized into an improved version; Figure 3 adopts a network structure with 400,000 iterations, 10 hidden layers, and 350 neurons per layer. All hidden layers of the network use the tanh activation function, and the final output is two-dimensional, representing the susceptible

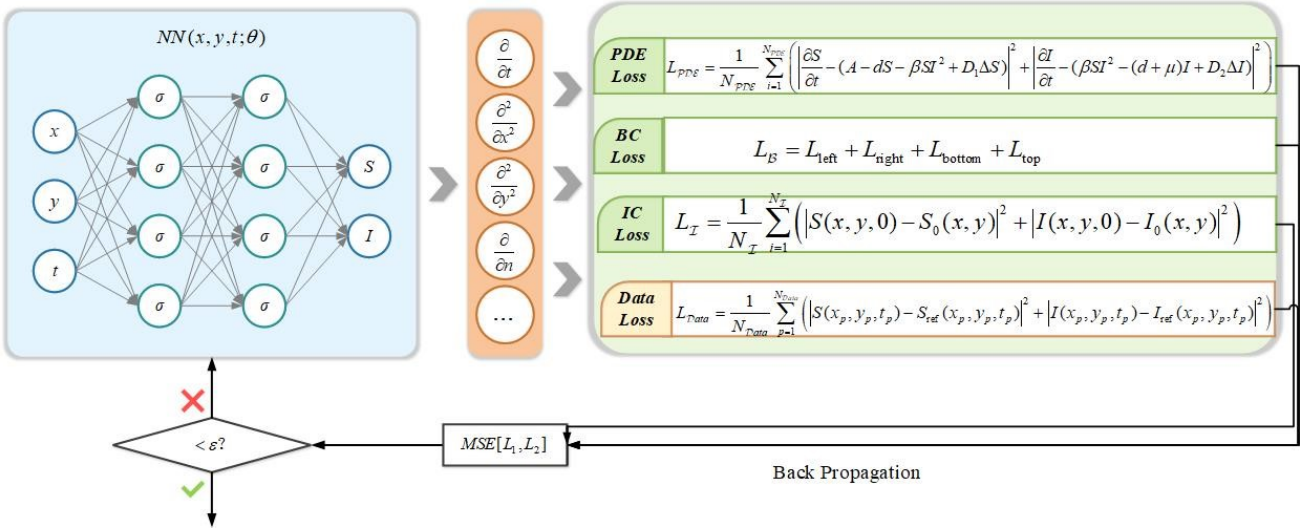


Figure 2. Architecture of the physics-informed neural network.

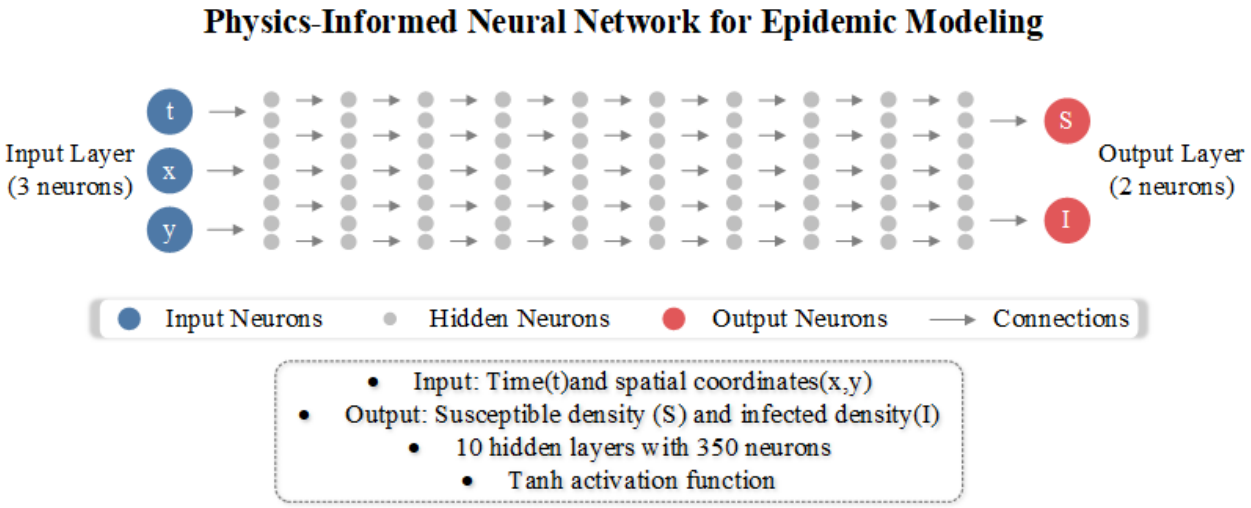


Figure 3. Design of the advanced physics-informed neural networks architecture: selection of input layer, hidden layers, output layer, and activation functions.

density S and infectious density I, respectively.

The forward propagation process of this network is as follows:

$$\begin{aligned} h_0 &= \tanh(W_0\mathbf{x} + b_0), \\ h_l &= \text{sigmoid}(W_{l,2}\tanh(W_{l,1}\text{LayerNorm}(h_{l-1}) + b_{l,1}) + b_{l,2}) \odot h_{l-1}, \end{aligned} \tag{4.2}$$

$$\begin{aligned} S &= 0.1 \times \text{softplus}(W_{l,1}\text{LayerNorm}(h_9) + b_{l,1}), \\ I &= 0.1 \times \text{softplus}(W_{l,2}\text{LayerNorm}(h_9) + b_{l,2}). \end{aligned} \tag{4.3}$$

During forward propagation, the neural network takes spatiotemporal coordinates $\mathbf{x}$ as input and first obtains initial features h_0 through a fully connected hidden layer activated by tanh. Subsequently,

the data flows through 9 identical hidden layers: for each layer, based on the output h_{l-1} of the previous layer, LayerNorm normalization is first performed, followed by feature transformation consisting of two linear transformations and tanh activation. Then, a sigmoid gate is used to perform element-wise multiplication with h_{l-1} to achieve weighted residual fusion. Finally, LayerNorm normalizes the 11th layer, which is processed by two independent linear layers separately, and the non-negative susceptible density S and infectious density I are output through a 0.1-scaled softplus activation function.

This paper incorporates 100 fourth-order Runge-Kutta numerical solutions at 1s intervals from 1 to 100 s as sparse data points into the loss function.

After multiple experimental adjustments, the weight coefficients for each constraint in the loss function were finally determined as $\omega_{\mathcal{PDE}} = 10$, $\omega_{\mathcal{B}} = 2$, $\omega_{\mathcal{I}} = 100$, $\omega_{\mathcal{D}ata} = 10$.

For optimization, the Adam optimizer was used, with an initial learning rate of 0.0005. Meanwhile, a cosine annealing scheduling strategy with warm-up was employed: linear learning rate warm-up was performed for the first 10,000 training steps, followed by decay to 10^{-7} according to a cosine function. This strategy effectively improved the stability and convergence efficiency of training.

This study systematically evaluated the performance of the Euler method, RK2 method, RK4 method, and PINNs in solving the SI infectious disease model, focusing on four scenarios with $\beta = 42$, 35, 40, and 32. The PINNs were divided into the basic PINN structure and an improved PINN, which incorporated increased network depth, a greater number of neurons per layer, and an extended number of iterations. Figure 4 shows the results of solving Eq (4.1) using the Euler method, RK2 method, RK4 method, PINNs, and Improved PINNs when the infection rate is $\beta = 42$.

As shown in Figure 4, in the early stage of simulation, $t < 20$ s, the solutions of all methods show little difference. However, after long-term simulation, the Euler method and RK2 method exhibit significant numerical diffusion and distortion in the "blue-eye" regions due to error accumulation. In contrast, the solutions of the RK4 method, PINNs, and Improved PINNs remain stable at all times and are highly consistent. This indicates that PINNs can effectively suppress error accumulation in long-term predictions, with accuracy comparable to that of traditional high-order numerical methods.

In addition to investigating the solution of Eq (4.1) by numerical methods and PINNs from a temporal perspective, the solution can also be explored from a spatial perspective. Figure 5 shows the variation of the density of infectious individuals I with time t at spatial points $(0, 0)$, $(5, 5)$, $(10, 10)$, $(15, 15)$, and $(20, 20)$ using the five methods.

Figure 5 clearly shows that the solutions of the RK4 method, PINNs, and Improved PINNs are more accurate than those of the Euler method and RK2 method. For the central point $(5, 5)$, the Improved PINNs, which employs deeper network layers, more neurons, and more iterations, achieves higher accuracy than the basic PINNs. Other spatial points show varying degrees of improvement in different time domains.

To understand the errors between PINNs and the RK4 method, we constructed error maps of the solutions from PINNs and the RK4 method at $t = 50$ s, as shown in Figure 6.

As shown in Figure 6, the solution of the Improved PINNs method locally approximates the solution of the RK4 method.

From a global perspective, we characterized the global relative errors of the PINNs and Improved PINNs methods relative to the RK4, RK2, and Euler methods, facilitating a global understanding of the solution performance.

To evaluate accuracy from a global perspective, Figure 7 shows the global relative errors of PINNs

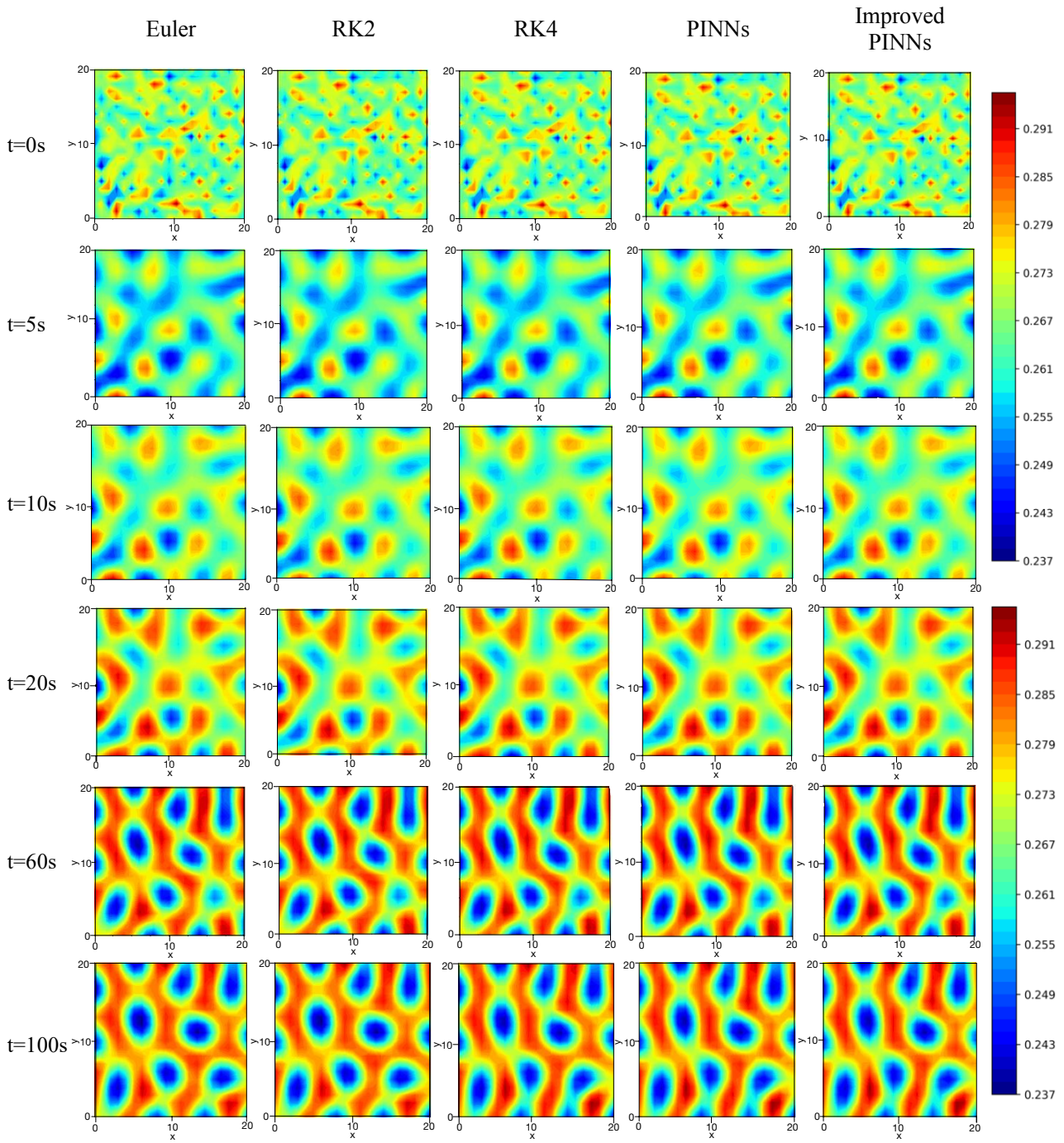


Figure 4. Comparison of spatiotemporal evolution of the infectious density I solved by different methods when $\beta = 42$.

and Improved PINNs relative to each numerical method. The results indicate that the minimum error between the Improved PINNs and the benchmark solution of the RK4 method is 5.79×10^{-3}, which is lower than the 6.25×10^{-3} of the basic PINNs. This quantitatively confirms that increasing network capacity and training costs helps further improve the approximation accuracy of PINNs.

The log details the dynamic evolution of five types of loss functions during training. The total loss

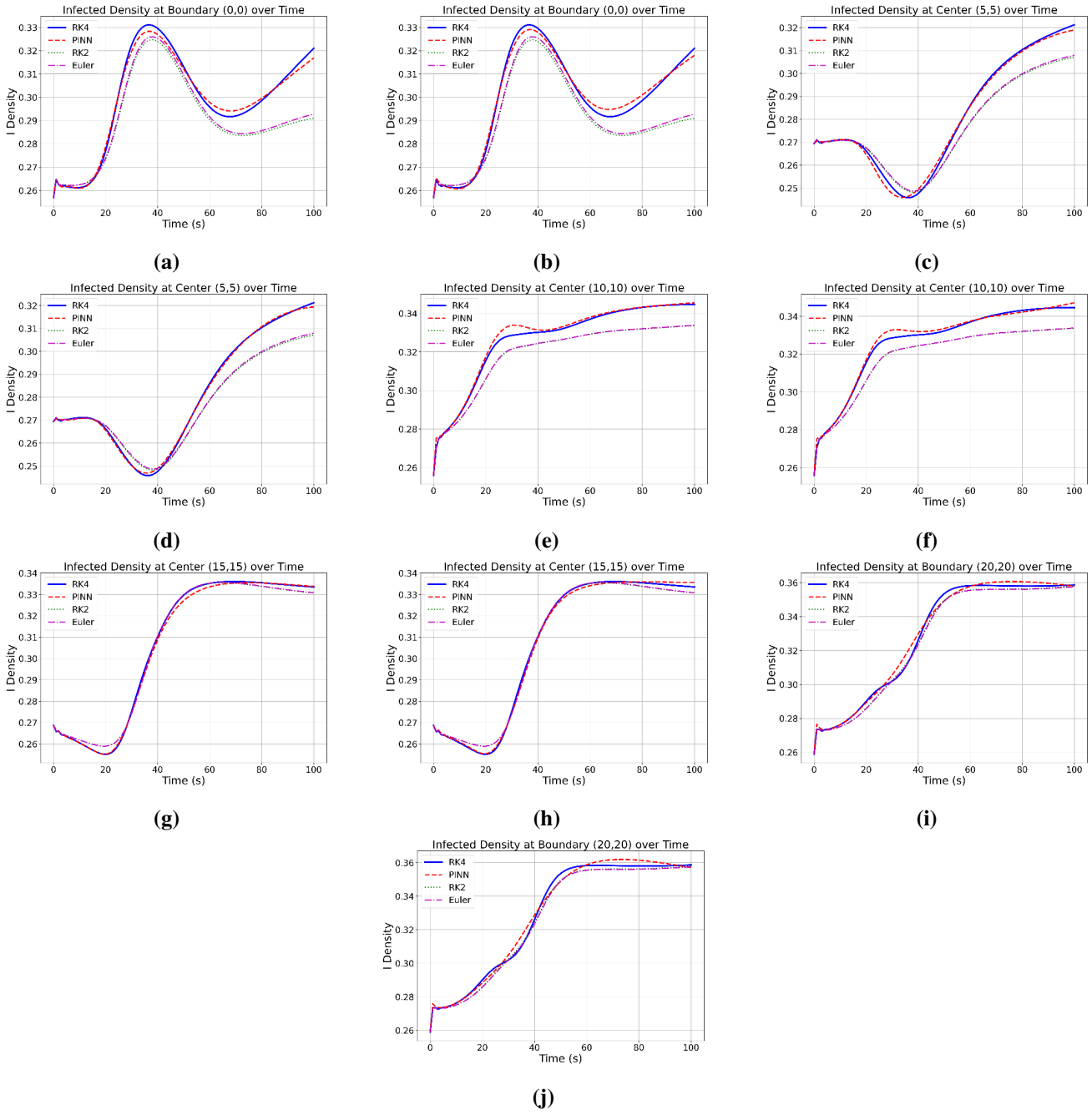
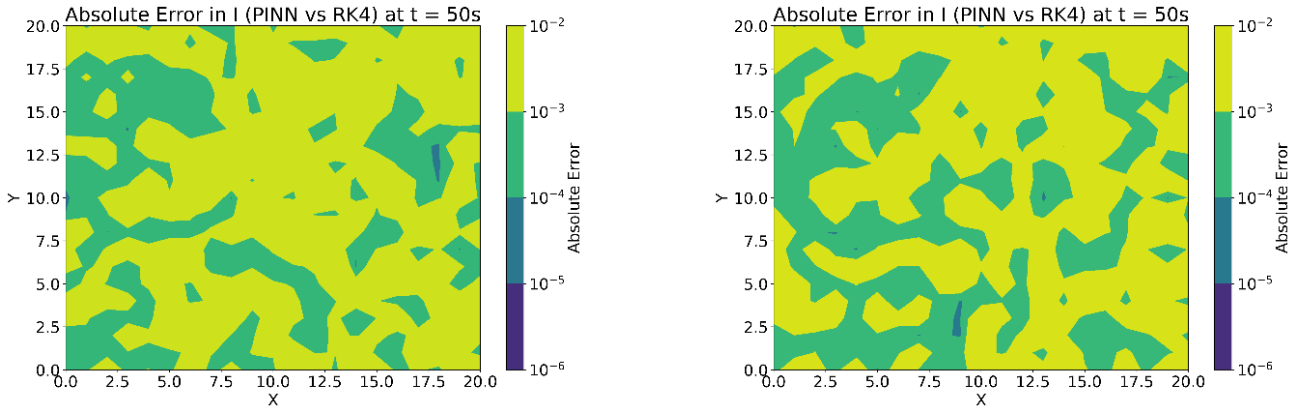


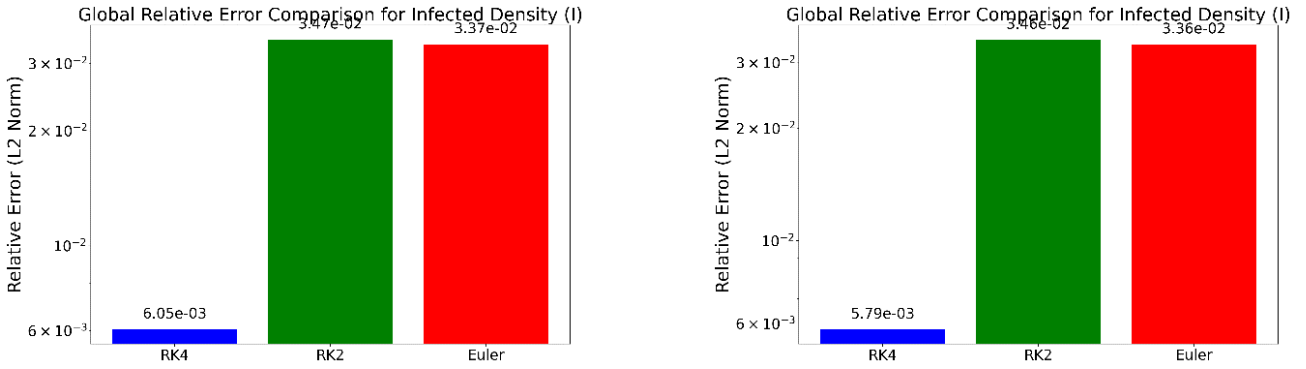
Figure 5. Comparison of the density of infectious individuals with time t at spatial points $(0, 0)$, $(5, 5)$, $(10, 10)$, $(15, 15)$, and $(20, 20)$ when $\beta = 42$. (a),(c),(e),(g),(i) Prediction results based on the basic PINNs. (b),(d),(f),(h),(j) Results the Improved PINNs method.

continuously decreases, and the PDE loss referring to the residual of the governing equation, the IC loss denoting the initial condition loss, the BC loss for the boundary constraint loss, and the data loss representing the observation point fitting loss all show a nonlinear convergence trend. By analyzing the loss curves, the dominant optimization objectives in different training stages can be clearly identified, providing key insights for diagnosing model convergence, adjusting weight strategies, and optimizing



(a) **(b)**

Figure 6. When $\beta = 42$, (a) shows the error between the solutions of PINNs and the RK4 method for the infectious density I at $t = 50$ s; (b) shows the error between the solutions of the Improved PINNs and the RK4 method.



(a) **(b)**

Figure 7. (a,b) Global relative errors of the PINNs and Improved PINNs methods, respectively relative to the RK4, RK2, and Euler methods when $\beta = 42$.

the training process [40].

Subsequently, Figures 9–11 characterize the solution of Eq (4.1) using the Euler method, RK2 method, RK4 method, and Improved PINNs under $\beta = 40$, 35, and 32, respectively.

As shown in Figure 9, the four methods show only minor differences at each time step.

As shown in Figure 10, the four methods exhibit slight differences.

As shown in Figure 11, the solutions of the four methods are approximately the same at $t = 5$ s and $t = 20$ s. However, at $t = 100$ s, there are significant differences between the solutions of the Euler method, RK2 method, RK4 method, and Improved PINNs, with obvious variations in the size range of the "red-eye" regions. The solutions of the Improved PINNs and RK4 method remain highly consistent at all time instants.

The results indicate that under large step size conditions, the solution accuracy of traditional numerical methods is significantly limited, with the error accumulation effect of the Euler method

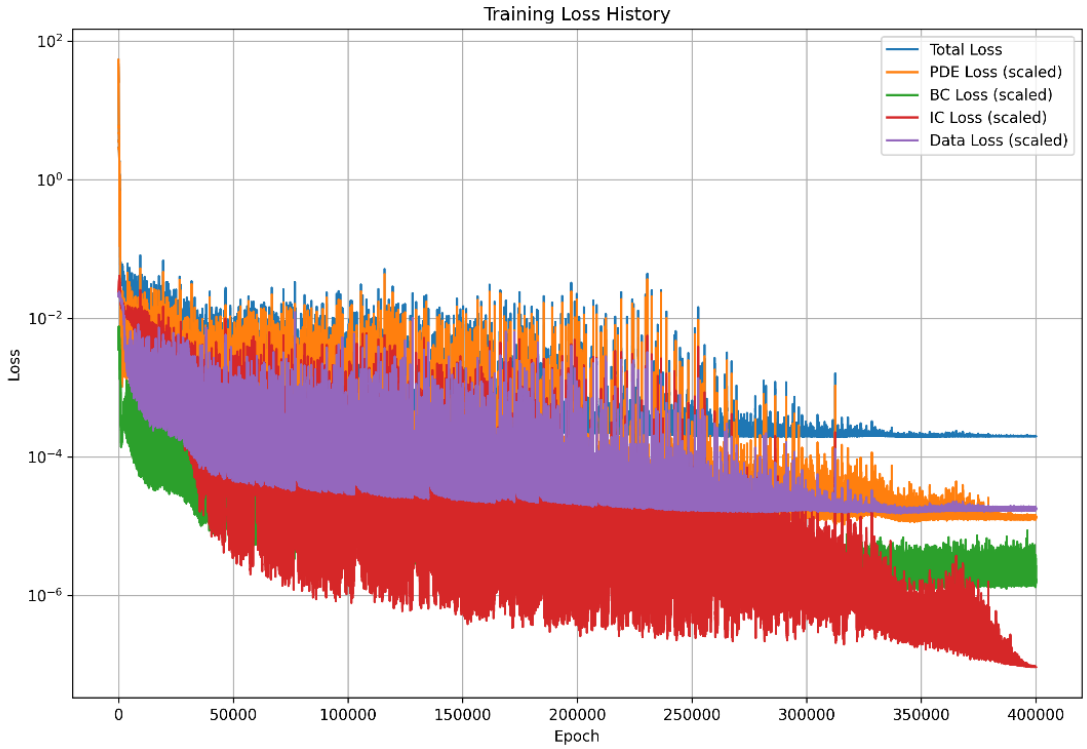


Figure 8. Training log of the physics-informed neural networks.

being particularly prominent. In contrast, the RK4 method can provide more accurate numerical solutions and thus serves as a reliable benchmark. The core conclusion of this study is that while traditional numerical methods achieve accuracy comparable to PINNs in short-term simulations, the Improved PINNs demonstrate significant advantages in long-term scenarios: they substantially outperform the Euler and RK2 methods and exhibit performance on par with the RK4 method in capturing complex spatial dynamics.

The reason why the Improved PINNs achieves such accuracy improvement lies in the inherent characteristics of the method itself. By taking both time and space variables as network inputs, the method achieves a globally continuous representation of the solution across the entire time domain from 0 to 100 seconds. This feature effectively overcomes the limitations of traditional numerical methods, which only provide solutions at discrete time points and suffer from global error accumulation with increasing steps.

5. Conclusions

5.1. Main conclusions

From the perspectives of spatiotemporal continuity and global solution, this study systematically compared the performance of the Euler method, RK2 method, RK4 method, and PINNs in solving

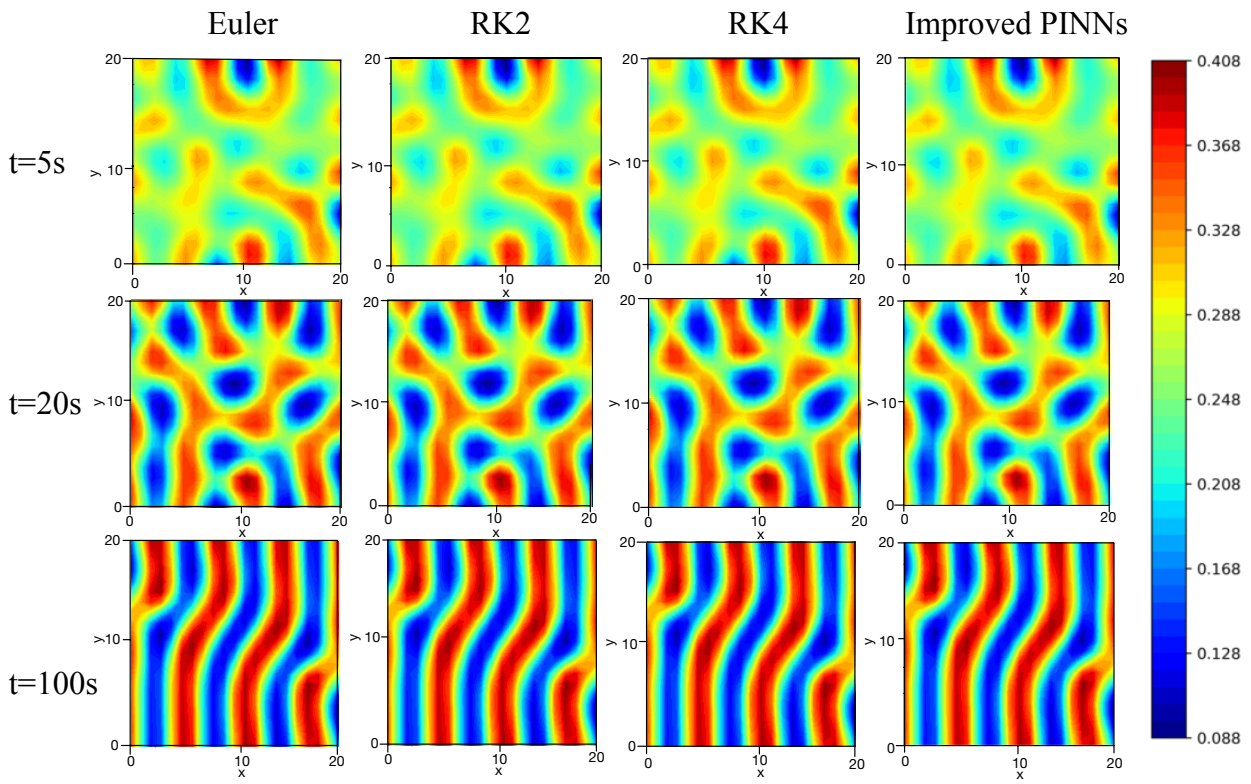


Figure 9. Comparison of spatiotemporal evolution of the infectious density I solved by different methods when the infection rate $\beta = 40$.

the spatially diffusive SI infectious disease model. The results show that under the supervision of supervised learning, PINNs, by integrating the governing equation, initial-boundary conditions, and high-precision numerical solutions into training constraints, significantly suppress error accumulation in long-term simulations and obtain more accurate solutions than the Euler method and RK2 method.

In addition, PINNs exhibit significant application potential in scenarios involving complex geometric domains, high-dimensional problems, scarce data, or real-time inference, thanks to their mesh-free solution characteristics, ability to integrate physical laws with observational data, and efficiency of "one-time training, global inference".

5.2. Future prospects

While PINNs offer a new paradigm for infectious disease dynamics modeling, they still face common challenges when dealing with such reaction-diffusion systems, including training instability, slow convergence rates, and sensitivity to hyperparameters. Future research can focus on in-depth exploration of the following three key directions: first, architectural innovation: introducing new network types such as Transformers and adaptive loss weight mechanisms to improve the model's prediction performance and training efficiency under extreme conditions; second, multi-scale modeling: developing cross-scale integration methods that can uniformly characterize individual behavior and population transmission dynamics, thereby enabling refined simulation and evaluation of the effectiveness of intervention measures; third, theoretical improvement: establishing a quantitative relationship between network structure and model error, and developing unsupervised

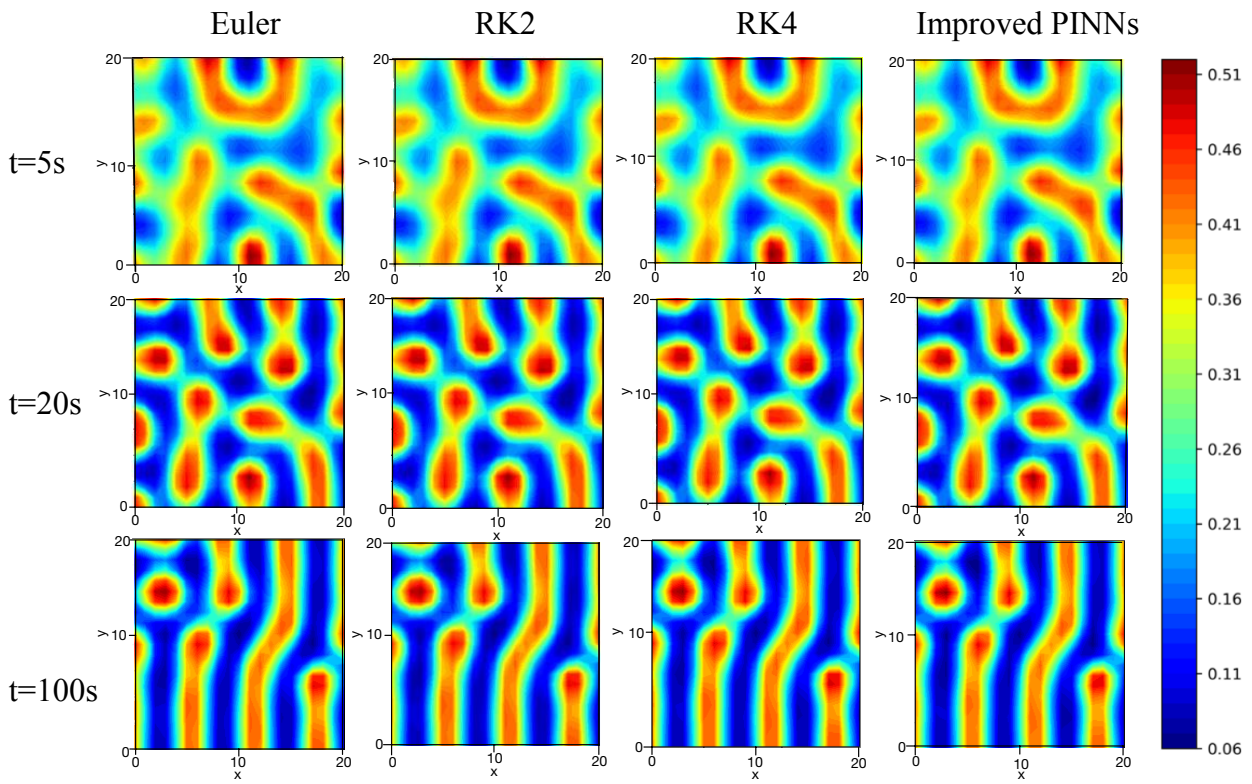


Figure 10. The infectious density I solved by different methods when the $\beta = 35$.

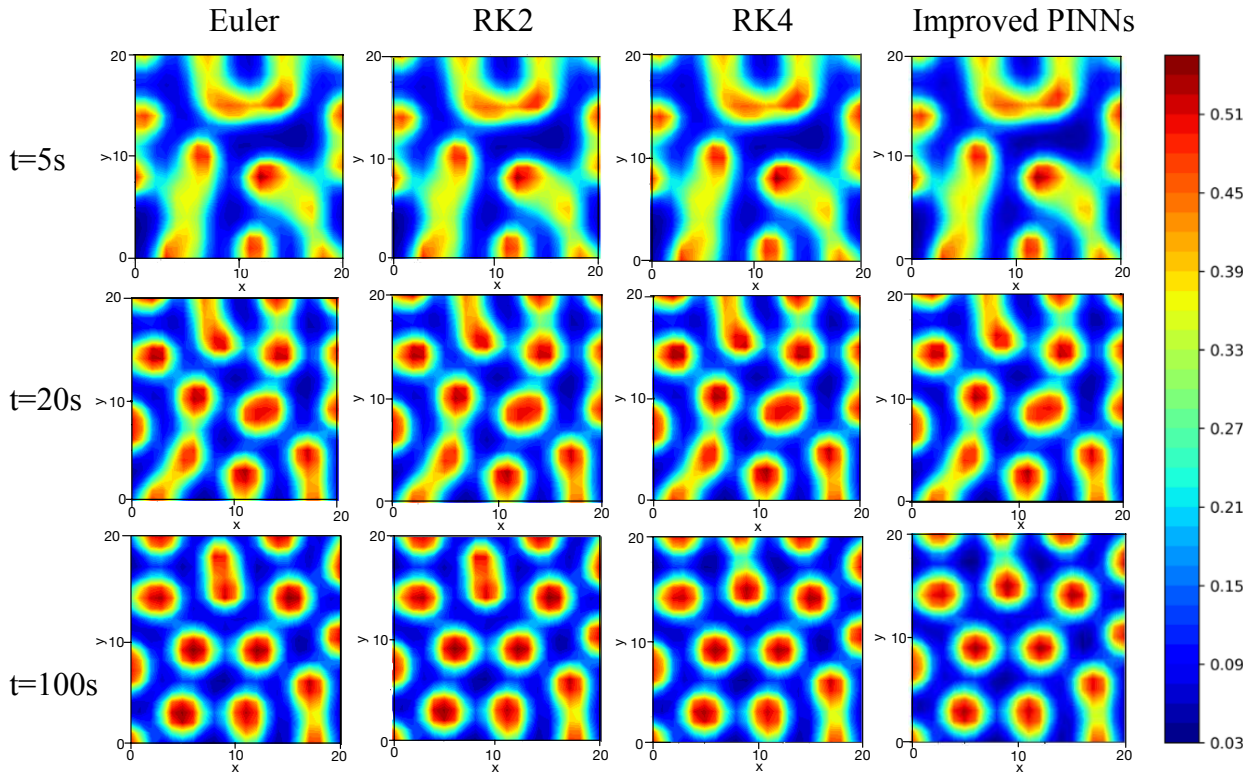


Figure 11. The infectious density I solved by different methods when the $\beta = 32$.

learning paradigms to enhance the model's robustness and generalization ability.

The advancement of the above technical paths aims to address the bottlenecks that limit the practical application of PINNs. By deeply integrating with traditional numerical methods, efforts will be made to improve the modeling accuracy and stability of PINNs in complex real-world scenarios, such as those with scarce data and unknown parameters, ultimately constructing a more reliable and practical infectious disease prediction and early warning system. This development path not only aims to provide strong technical support for infectious disease prevention and control decisions but also, more importantly, opens up a new research paradigm that integrates physical mechanisms with data-driven intelligence for the numerical solution of complex dynamic systems.

Use of AI tools declaration

The authors acknowledge the use of Artificial Intelligence (AI) tools in this research. Specifically, all computations, including the implementation of numerical methods and the training of Physics-Informed Neural Networks (PINNs), were completed using the PyTorch 2.1.1 framework on a computational platform equipped with an NVIDIA GeForce RTX 4090 GPU.

Acknowledgment

We are very grateful to anonymous recommenders for their careful reading and valuable comments. This work was supported by Natural Science Foundation of China (11801470, 12161052, U22A20231), Natural Science Foundation of Sichuan Province(2022NSFSC1819, 2023NSFSC1345), Central Government Funds for Guiding Local Scientific and Technological Development (2023ZYD0002), Humanities and Social Science Fund of Ministry of Education(23YJA890038) and the Fundamental Research Funds for the Central Universities (2682024ZTPY051).

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. 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
2. Bengio Y, Lecun Y, Hinton G, (2021) Deep learning for AI. *Commun ACM* 64: 58–65. https://doi.org/10.1145/3448250
3. Wang X, Yin ZY, Wu W, Zhu HH, (2025) Differentiable finite element method with Galerkin discretization for fast and accurate inverse analysis of multidimensional heterogeneous engineering structures. *Comput Methods Appl Mech Eng* 437: 117755. https://doi.org/10.1016/j.cma.2025.117755
4. Li Y, Wang F, (2025) Local randomized neural networks with finite difference methods for interface problems. *J Comput Phys* 529: 113847. https://doi.org/10.1016/j.jcp.2025.113847

5. Cen J, Zou Q, (2024) Deep finite volume method for partial differential equations. *J Comput Phys* 517: 113307. https://doi.org/10.1016/j.jcp.2024.113307
6. DiBenedetto E, Gianazza U, (2023) *Partial Differential Equations*, Springer Nature. https://doi.org/10.1007/978-3-031-46618-2
7. Bellman R, (1966) Dynamic programming. *Science* 153: 34–37. https://doi.org/10.1126/science.153.3731.34
8. Benner P, Goyal P, Kramer B, Peherstorfer B, Willcox K, (2020) Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms. *Comput Methods Appl Mech Eng* 372: 113433. https://doi.org/10.1016/j.cma.2020.113433
9. Raissi M, Perdikaris P, Karniadakis GE, (2019) PINN: 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
10. Cai S, Mao Z, Wang Z, Yin M, Karniadakis GE, (2021) PINN (PINNs) for fluid mechanics: A review. *Acta Mech Sin* 37: 1727–1738. https://doi.org/10.1007/s10409-021-01148-1
11. Hu B, McDaniel D, (2023) Applying PINN to solve Navier-Stokes equations for laminar flow around a particle. *Math Comput Appl* 28: 102.https://doi.org/10.3390/mca28050102
12. 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
13. Haghighat E, Raissi M, Moure A, Gomez H, Juanes R, (2021) A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Comput Methods Appl Mech Eng* 379: 113741.https://doi.org/10.1016/j.cma.2021.113741
14. Lee S, Popovics J, (2022) Applications of PINN for property characterization of complex materials. *RILEM Tech Lett* 7: 178–188.https://doi.org/10.21809/rilemtechlett.2022.174
15. Sarabian M, Babaee H, Laksari K, (2022) PINN for brain hemodynamic predictions using medical imaging. *IEEE Trans Med Imaging* 41: 2285–2303. https://doi.org/10.1109/TMI.2022.3161653
16. Kissas G, Yang Y, Hwuang E, Witschey WR, Detre JA, Perdikaris P, (2020) Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using PINN. *Comput Methods Appl Mech Eng* 358: 112623. https://doi.org/10.1016/j.cma.2019.112623
17. Lu L, Meng X, Mao Z, Karniadakis GE, (2020) A deep learning library for solving differential equations. *SIAM Rev* 63: 208–228. https://doi.org/10.1137/19M1274067
18. Cuomo S, Schiano Di Cola V, Giampaolo F, Rozza G, Raissi M, Piccialli F, (2022) Scientific machine learning through physics-informed neural networks: Where we are and what's next. *J Sci Comput* 92: 88. https://doi.org/10.1007/s10915-022-01939-z
19. Shaier S, Raissi M, Seshaiyer P, (2021) Data-driven approaches for predicting spread of infectious diseases through DINNs: Disease informed neural networks, preprint, arXiv:2110.05445. https://doi.org/10.48550/arXiv.2110.05445
20. Chen X, Li F, Lian H, Wang P, (2025) A deep learning framework for predicting the spread of diffusion diseases. *Electron Res Arch* 33(4): 2475–2502.

21. Yin S, Wu J, Song P, (2023) Optimal control by deep learning techniques and its applications on epidemic models. *J Math Biol* 86: 36. https://doi.org/10.1007/s00285-023-01873-0
22. Pastor-Satorras R, Castellano C, Van Mieghem P, Vespignani A, (2015) Epidemic processes in complex networks. *Rev Mod Phys* 87: 925–979. https://doi.org/10.1103/RevModPhys.87.925
23. Liu W, Hethcote HW, Levin SA, (1987) Dynamical behavior of epidemiological models with nonlinear incidence rates. *J Math Biol* 25: 359–380. https://doi.org/10.1007/BF00277162
24. Liu W, Levin SA, Iwasa Y, (1986) Influence of nonlinear incidence rates upon the behavior of SIRS epidemiological models. *J Math Biol* 23: 187–204. https://doi.org/10.1007/BF00276956
25. Sun GQ, (2012) Pattern formation of an epidemic model with diffusion. *Nonlinear Dyn* 69: 1097–1104. https://doi.org/10.1007/s11071-012-0330-5
26. 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
27. 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
28. Madzvamuse A, Chung AHW, (2014) Fully implicit time-step 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
29. 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
30. Bochacik T, Przybyłowicz P, Stępień Ł, (2025) Convergence and stability of randomized implicit two-stage Runge-Kutta schemes. *BIT Numer Math* 65: 7. https://doi.org/10.1007/s10543-024-01050-9
31. Salah M, Matbuly MS, Civalek O, Ragb O, (2023) Calculation of four-dimensional unsteady gas flow via different quadrature schemes and Runge-Kutta 4th order method. *Adv Appl Math Mech* 15: 1–22. https://doi.org/10.4208/aamm.OA-2021-0373
32. 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
33. Zhang Q, Wu C, Kahana A, Karniadakis GE, Kim Y, Li Y, Panda P, (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
34. 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
35. Chang L, Wang X, Sun G, Wang Z, Jin Z, (2024) A time independent least squares algorithm for parameter identification of Turing patterns in reaction-diffusion systems. *J Math Biol* 88: 5. https://doi.org/10.1007/s00285-023-02026-z

36. Chang LL, Gao S, Wang Z, (2022) Optimal control of pattern formations for an SIR reaction–diffusion epidemic model. *J Theor Biol* 111003. https://doi.org/10.1016/j.jtbi.2022.111003

37. Chang LL, Gong W, Jin Z, Sun GQ, (2022) Sparse optimal control of pattern formations for an SIR reaction-diffusion epidemic model. *SIAM J Appl Math* 82: 1764–1790. https://doi.org/10.1137/22M1472127

38. Zhou R, Wu S, Bao X, (2024) Propagation dynamics for space-time periodic and degenerate systems with nonlocal dispersal and delay. *J Nonlinear Sci* 34: 69. https://doi.org/10.1007/s00332-024-10048-0

39. Zhang X, Wu S, Zou L, Hsu C, (2025) Spreading dynamics for a time-periodic nonlocal dispersal epidemic model with delay and vaccination. *J Math Biol* 90: 54. https://doi.org/10.1007/s00285-025-02214-z

40. Cao F, Fan K, Zhang H, Yuan D, Liu J, (2025) Adaptive residual splitting in PINNs for solving complex PDEs. *J Comput Phys* 114297. https://doi.org/10.1016/j.jcp.2025.114297

AIMS Press