Journal of Network Computing and Applications (2025)
Clausius Scientific Press, Canada
DOI: 10.23977/jnca.2025.100111
ISSN 2371-9214 Vol. 10 Num. 1

Physics-Informed GNN Coupled with ESN for Solving Forward Problems of Spatiotemporal Partial Differential Equations

Yushen Tang[1], Jin Su[1,*]

[1]*School of Science, Xi'an Polytechnic University, Xi'an, Shaanxi, China*
**Corresponding author*

Keywords: Partial differential equations, Graph neural network, Echo state network, Physics-informed neural networks

Abstract: Partial Differential Equations (PDEs) are the foundation of modeling and simulation in numerous scientific and engineering fields. In recent years, breakthrough advancements in deep learning, particularly the rise of Physics-Informed Neural Networks (PINNs), have opened up a data-driven new paradigm for PDE solving and demonstrated enormous potential. However, PINN is essentially a global fitting method based on fully connected networks, and its core drawback is that the global fitting characteristics lead to a large amount of redundancy in high-order derivative calculations and insufficient modeling of spatiotemporal correlations. To address this, we propose the Physics-Informed E-GNN method, which modeling spatiotemporal features separately under a discrete learning framework to improve the accuracy of spatiotemporal prediction. Our method first discretizes the initial values of the PDE into a graph structure as input, feeds it into a Graph Neural Networks (GNN) to update the spatial features, and then inputs the updated feature vectors into an Echo State Networks **(**ESN) autoregressive module in the form of a time series to capture sequence correlations. We conducted comparative experiments on two classic partial differential equations (the 2D Burgers' equation and the 2D Convection-Diffusion equation) in irregular domains. The experimental results show that our proposed method achieves significant improvements in both solution accuracy and generality, and can effectively capture the complex patterns of changes in the PDE system.

1. Introduction

Many important engineering and physical problems are governed by PDEs. In recent years, PINN[1-3] shave attracted considerable attention as a representative deep learning technique for solving PDEs. Traditional PINN is a global fitting based on fully connected networks, with a loss function consisting of PDE residual loss, initial condition loss, and boundary condition loss. PINNs have been successfully applied in multiple fields due to their higher efficiency compared to traditional numerical methods[4-8].However, when facing complex problems, traditional PINNs struggle to simulate the changing trends of physical phenomena due to their inherent optimization difficulties [9]. Therefore, PINNs based on discrete learning and boundary hard coding have attracted widespread research attention[10-13].

GNNs[14][15]possess excellent modeling capabilities for graph structures. Through GNNs' node update and neighborhood propagation rules, they can efficiently fit the spatial interdependencies of PDEs. Additionally, GNNs can easily handle unstructured data in irregular domains, thereby enabling the solution of PDEs in such domains. Existing studies have solved PDEs using discrete learning with GNNs: for example, Zhang et al.[16] combined GNNs with the finite difference method, providing a new paradigm for discrete learning; Zeng et al.[17]integrated GNNs with the Runge-Kutta integration method, improving the accuracy of long-term predictions.

Current methods have addressed the challenge of capturing spatial features. However, for more complex spatiotemporal PDEs, there is a lack of targeted design for modules that capture local features or long-term dependencies, leaving significant room for improvement in PDE prediction models.

As a special type of Recurrent Neural Network (RNN), ESNs [18][19] simplify the training process by retaining an untrained recurrent layer (referred to as a "reservoir") and only training the output layer. ESNs are particularly adept at capturing complex spatiotemporal dynamic features: they project input data into a high-dimensional space via the reservoir, transforming the input data into spatiotemporal patterns with complex dynamics. ESNs have successfully solved various dynamic problems: Xu Peng et al. proposed a torsion balance dynamics prediction model with improved Reservoir Computing (RC), which uses an attention mechanism to enhance the long-term dependency features of time-series data; Ma et al.[20]proposed a deep projection-encoded echo state network, whose most notable feature is its ability to learn multi-scale dynamics through stacked ESN connected by subspace projections; Na et al.[21] developed a novel sparse learning hierarchical echo state network, which not only excavates and captures potential evolutionary patterns hidden beyond dynamic systems but also possesses feature selection capabilities.

Therefore, inspired by[22][23], we propose a novel physics-informed graph network recursive architecture, to sensitively capture the temporal evolution features of PDEs. Building on the traditional GNN model, this work combines the temporal modeling capability of ESNs with the local feature extraction advantage of GNNs to construct a G-ESN neural network. The E-GNN is used to fit the Right-Hand Side (RHS) operator, and the global solution is obtained through a time integration method (residual connections) under physics-informed constraints. We tested our model on two representative 2D PDE problems and compared it with PINNs in terms of generality and accuracy. Through solving two classic spatiotemporal PDEs, we verified the superiority of E-GNN over PINNs in terms of accuracy and generality.

2. Theoretical Background

Consider the following spatiotemporal dynamic system described by a nonlinear PDE:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathcal{F}[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_x \mathbf{u}, \nabla_x^2 \mathbf{u}, \nabla_x \mathbf{u} \cdot \mathbf{u}, \ldots, \lambda] = 0. \tag{1}$$

Where $\mathbf{u}(x,t) \in R^n$ denotes the solution variable defined over the time interval $t \in [0,T]$ and the spatial domain Ω; $\frac{\partial \mathbf{u}}{\partial t}$ represents the first-order time derivative; ∇ stands for the gradient operator in the space x; and $\mathcal{F}[\cdot]$ is a nonlinear function controlled by the parameter λ. The I/BCs are formulated in the form of

$$I[u,u_t;t=0,x\in\Omega]=0,$$
$$B[u,\nabla_x u,\mathrm{L}\ ;x\in\partial\Omega]=0. \tag{2}$$

Based on the initial conditions and boundary conditions of the aforementioned partial differential equation, we focus on sparse observation nodes $\{x_1,\ x_2,\ldots,\ x_M\}$ and uniformly discretized time intervals $t_k=t_0+k\delta t$, $k=0,1,2,\mathrm{L}\ ,N$ within spatiotemporal domain $\Omega\times[0,T]$.The global solution of the PDE within the spatiotemporal domain is obtained through numerical methods. In this paper, our goal is to find the solution of the partial differential equation using physical laws as constraints without relying on data.

3. Method

3.1 Residual Connection Model Based on Physics-Informed

3.1.1 Residual Connection

In our implementation, when initial conditions are given, the state of the system at each discrete time step is determined through Euler time integration, a global residual connection was constructed to advance the time evolution of the PDE solution. The model framework is illustrated in Figure 1.

Specifically, we use the forward Euler method to uniformly discretize the time derivative over the continuous time interval, thereby obtaining a discrete dynamic system, as shown in Eq. 3

$$\frac{\partial \mathbf{u}}{\partial t}\approx\frac{\mathbf{u}(\mathbf{x},t+\delta t)-\mathbf{u}(\mathbf{x},t)}{\delta t}, \tag{3}$$

According to Eq.3, we designed a coupled neural network N^{θ} to approximate the RHS, then, we constructed a global residual connection to drive the temporal evolution of the PDE solution. The time integration process is illustrated in Figure 1.

$$\mathbf{u}_{k+1}=\mathbf{u}_k+\delta t(N^{\theta}(t_k,\mathbf{u}_k)), \tag{4}$$

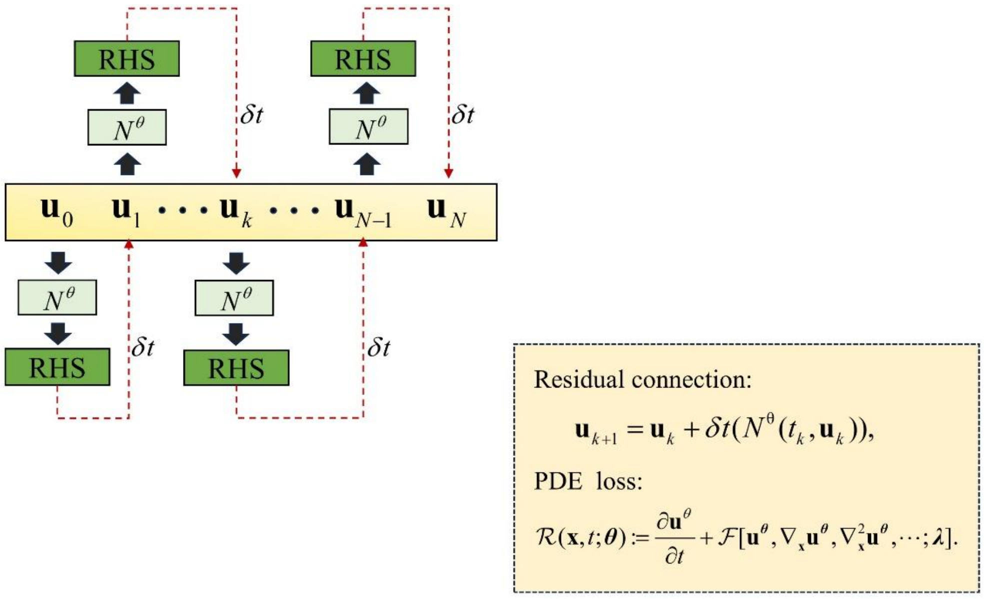
RHS
N^{θ}
δt
RHS
N^{θ}
δt
$\mathbf{u}_0$ $\mathbf{u}_1$ $\cdots$ $\mathbf{u}_k$ $\cdots$ $\mathbf{u}_{N-1}$ $\mathbf{u}_N$
N^{θ}
RHS
δt
N^{θ}
RHS
δt
Residual connection:
$\mathbf{u}_{k+1}=\mathbf{u}_k+\delta t(N^{\theta}(t_k,\mathbf{u}_k)),$
PDE loss:
$\mathcal{R}(\mathbf{x},t;\theta):=\frac{\partial\mathbf{u}^{\theta}}{\partial t}+\mathcal{F}[\mathbf{u}^{\theta},\nabla_{\mathbf{x}}\mathbf{u}^{\theta},\nabla_{\mathbf{x}}^{2}\mathbf{u}^{\theta},\cdots;\lambda].$

Figure 1. E-GNN Framework Based on Residual Connections

3.1.2 Physics-Informed Loss Function

During model training, for the Eq.1, the loss function $\mathcal{R}(\mathbf{x},t;\boldsymbol{\theta})$ is constructed from the equation information.

$$\mathcal{R}(\mathbf{x},t;\boldsymbol{\theta}) := \frac{\partial \mathbf{u}^{\theta}}{\partial t} + \mathcal{F}[\mathbf{u}^{\theta}, \nabla_{\mathbf{x}}\mathbf{u}^{\theta}, \nabla_{\mathbf{x}}^{2}\mathbf{u}^{\theta}, \ldots; \boldsymbol{\lambda}]. \tag{5}$$

To achieve approximate calculation of differential operators on unstructured grids, the finite difference method is suitable for computing Laplacian operators on regular grids but performs poorly on unstructured grids. For gradient and Laplacian values, a discrete difference method based on the least squares method [16](Zhang et al., 2024) is adopted. In gradient calculation, a linear relationship is constructed through the coordinate difference matrix and value difference matrix between the current node and its nearest neighbors, and then the gradient is solved via least squares optimization. In Laplacian calculation, a test function containing coordinates and quadratic terms is selected to capture local curvature information. After constructing the test function difference matrix, the least squares method is used to solve the weight coefficients, and then the Laplacian value is obtained, thereby addressing the insufficient applicability of the finite difference method on unstructured grids.

3.2 GNN

In terms of spatial processing, we first discretize the physical domain at time $k \in [0,T]$ into graph data $G^{k}(V,E)$, Among them, $V=\{v_1,v_2,\ldots,v_M\}$ denotes the node set, and $E=\{e_1,e_2,\ldots,e_L\}$ denotes the edge set. Where each node represents a spatial position, and treat the edges of the mesh as the edges of the graph. Each node typically carries feature information describing its own attributes, referred to as node features, which is a vector composed of the solution $\mathbf{u}_k(i)$ at the node and the node type. Similarly, each edge usually carries information describing the relationship between two connected nodes, which is determined by the distance between nodes i and j.

Based on the graph structure constructed above, we adopt an edge-aggregation-based GNN to capture the spatial interactions between nodes in the physical system and complete feature updates. Its core lies in realizing explicit modeling of spatial topological relationships and physical interactions through a message-passing mechanism between nodes and their neighborhoods. The update process is illustrated in Figure 2.

It consists of two steps: edge update and node update.

Edge update integrates the features of the sending node, receiving node, and the current edge feature. Three independent nonlinear transformation networks Φ_s, Φ_r, Φ_e are used to map the features $\mathbf{v}_k^f(i)$ and $\mathbf{v}_k^f(j)$ of the receiving node i and the sending node j respectively

$$\mathbf{V}_k^f(i) = \Phi_s(\mathbf{v}_k^f(i)) \tag{6}$$

$$\mathbf{V}_k^f(j) = \Phi_r(\mathbf{v}_k^f(j)) \tag{7}$$

Then, combined with the original edge feature $\mathbf{e}_k^f(ij)$, the updated edge feature $\mathbf{e}_k^{\prime f}(ij)$ is obtained through the edge feature transformation function Φ_e, that is

$$\mathbf{e}_k^{\prime f}(ij) = \Phi_{\mathrm{e}}\left(\mathbf{V}_k^f(i) + \mathbf{V}_k^f(j) + \mathbf{e}_k^f(ij)\right), \tag{8}$$

The update of node features is based on the aggregated result of neighboring edge features. First, the features of all neighboring edges of the receiving node *j* are summed and aggregated; then, the message is concatenated with the node's own features, and the node features are updated through the nonlinear transformation network Φ_{n}

$$\mathbf{v}_k'^f(i) = \Phi_{\mathrm{n}}([\mathbf{v}_k^f(i); \sum_{j\in \mathrm{N}_i} \mathbf{e}_k'^f(ij)]). \tag{9}$$

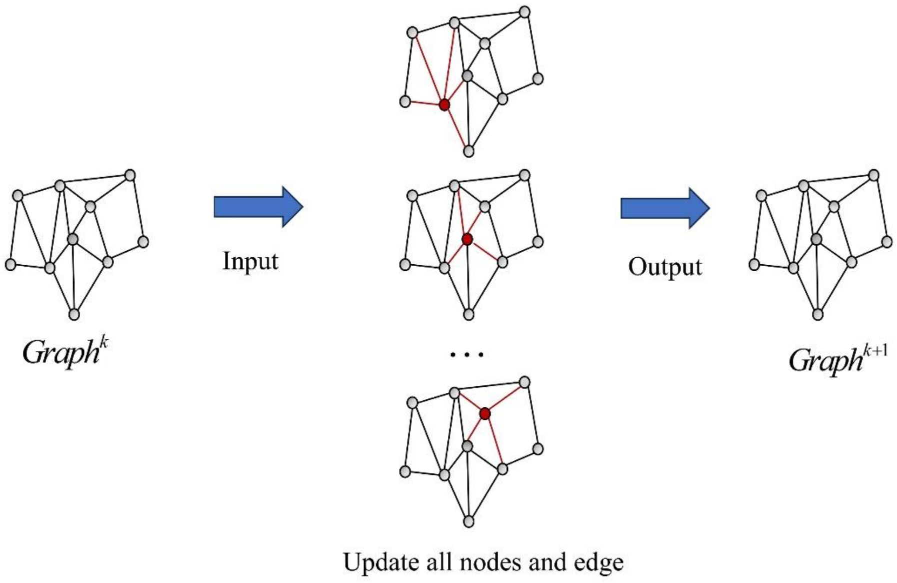

Figure 2. GNN Block

To adapt to the input requirements of the subsequent temporal processor, we convert the node features output by the GNN from a graph structure to a time-series structure. We then expand the dimension through an unsqueeze operation to obtain the time series

$$\mathbf{s}_k = unsqueeze(\mathbf{v}_k'^f(i), 1), \tag{10}$$

3.3 ESN

ESNs as a special type of RNNs, feature a self-feedback mechanism and a sparsely connected reservoir structure. They can efficiently capture dynamic evolution information in time-series signals, and their network structure is illustrated in Figure 3(b).

The core advantage of ESNs lies in efficient training: the reservoir is designed with fixed sparse connections, and only the output layer weights need to be learned. This significantly reduces the scale of trainable parameters, leading to a marked decrease in the time complexity of the training process. Its state update process can be expressed as:

$$\mathbf{h}_{k+1} = (1-\alpha)\mathbf{h}_k + \alpha \tanh\left(\mathbf{W}_{in}\mathbf{s}_k + \mathbf{W}\mathbf{h}_k\right), \tag{11}$$

Among them, $\mathbf{h}_k$ denotes the hidden state of the reservoir, α represents the leakage rate (which controls the attenuation degree of state update), $\mathbf{W}_{in}$ is the input weight matrix, and $\mathbf{W}$ is the fixed sparse connection matrix inside the reservoir. The activation function $\tanh$ introduces nonlinear characteristics into state evolution, enabling the ESN to characterize complex temporal dependencies.

After obtaining the updated state information $\mathbf{h}_{k+1}$, the output $\mathbf{y}_{k+1}$ at the next time step is obtained through the learnable weight $\mathbf{W}_{out}$.

$$\mathbf{y}_{k+1} = \mathbf{W}_{out}\mathbf{h}_{k+1}. \tag{12}$$

3.4 Establishment of the G-ESN Network Structure

Although GNNs perform remarkably in spatial correlation modeling—being able to accurately capture local and global spatial interaction features under complex topological structures via the message-passing mechanism between nodes, and enhance the representation capability for non-Euclidean spaces—they still have limitations in solving PDEs. Relying solely on spatial dimension information cannot fully characterize the dynamic evolution process of the system. The spatiotemporal evolution of PDE solution fields is often the result of the synergistic effect of spatial correlations and temporal dynamics. These solution fields exhibit long-short-term dependencies and dynamic change laws over time, which are difficult for GNN architectures alone to effectively capture. In contrasts, by virtue of their reservoir memory mechanism, can efficiently learn the evolutionary laws of the temporal dimension from time-series data through dynamic state propagation and autoregressive properties. This endows spatial features with dynamic evolution constraints, making ESNs more compatible with the spatiotemporal coupling mechanism of PDE systems. This paper proposes an E-GNN coupled network model. By integrating ESNs into GNNs, the model fully leverages the advantages of GNNs in spatial correlation modeling and the strong temporal dynamic capture capability of ESNs. The structure of the coupled network is illustrated in Figure 3.

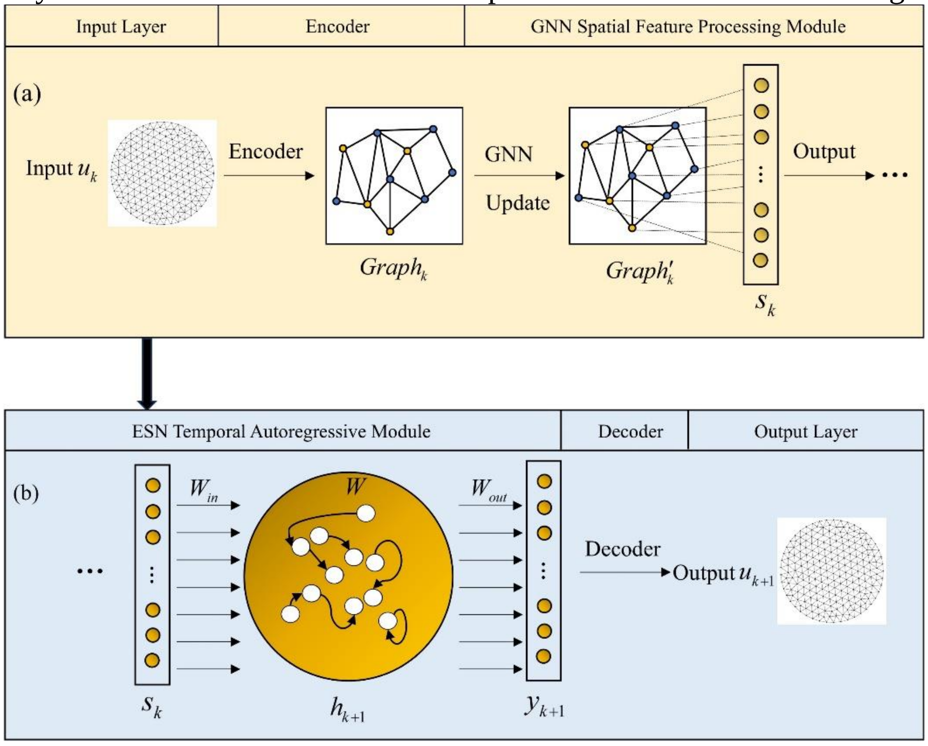

Figure 3. Structure of the E-GNN Network

The network consists of four components: an encoder, a GNN block, an ESN block, and a decoder. The functions of these four components are as follows: The encoder and GNN work together to convert the PDE solution at the previous time step into structured high-dimensional graph features; the GNN explores the local correlations and global spatial interactions of the solution field through the node message-passing mechanism. After the graph features are converted into a time series, they are input into the ESN autoregressive module. Using the dynamic memory and state propagation mechanism of the ESN's reservoir, the module establishes temporal dependencies between node states, enabling efficient capture of the temporal evolution laws of the PDE. Finally, the decoder maps the learned spatiotemporal fusion features back to the original solution space, yielding the numerical solution of the PDE at the prediction time step.

The algorithm framework of E-GNN is as Algorithm 1.

Algorithm 1 E-GNN

```
Input: Initial value: $\mathbf{u}_0$
Hyperparameters:
$\psi_{GNN} = \{\Phi_s, \Phi_r, \Phi_e, \Phi_n\}$        //Various parameters of GNN
$\tau_{ESN} = \{\alpha, \mathbf{W}_{in}, \mathbf{W}, \mathbf{W}_{out}\}$        // Various parameters of ESN
Output: $\mathbf{U}$        //A list of approximate states of the PDE solution
1:Function $NN(\mathbf{u}_0, \psi_{GNN}, \tau_{ESN})$
2: $\mathbf{U} = List()$
3: $Append(\mathbf{u}_0, \mathbf{U})$
4: $V = GNN(\mathbf{u}_0, \psi_{GNN})$        //Use GNN to extract spatial features from the initial state $\mathbf{u}_0$
5:     for $k$ from $0, 1, \ldots, N$
6:         $T = ESN(V, \tau_{ESN})$        //Use ESN to extract temporal features
7:         $\mathbf{u}_{k+1} = Integrator(\mathbf{u}_k, T)$
8:         $Append(\mathbf{u}_{k+1}, \mathbf{U})$        //Update the list with the new state
9:     End for
10:Return $\mathbf{U}$
```

Algorithm 1. E-GNN

4. Numerical Experiments

4.1 Network Settings

The specific experiments consist of two groups: (1) comparing the forward problem-solving accuracy between E-GNN and PINN, and (2) investigating the parameter inversion performance of the E-GNN for inverse problems. All numerical implementations in this paper are coded using PyTorch. We use the same network structure settings for all PDE instances. The spatial processor consists of one layer of a nonlinear graph convolutional network. The structure of the MLP includes three layers, where the dimension of the hidden layer is 128.The number of nodes in the graph is 7301. For the ESN used as the temporal processor, the reservoir radius is set to 0.95 and the reservoir size is configured to 100. The physics-informed weight is 0.5.

For the stochastic gradient descent of the network, the Adam optimizer is adopted. A learning rate scheduler is used to dynamically adjust the learning rate during the training process: the initial learning rate is 0.01, and the learning rate is reduced by a factor of 0.98 every 25 epochs.

4.2 PINN Model Settings

To ensure the effectiveness of the baseline model, we set the number of hidden layers to 4, with 64 neurons in each layer, and select the hyperbolic tangent function (Tanh) as the activation function. In addition, 7301 residual sampling points are chosen, which can effectively calculate the residual within the domain, and the Adam optimizer is adopted, with the learning rate set to $1e-3$.

4.3 Evaluation Metrics

In all numerical experiments in this article, we set the root mean square error (RMSE) at the collocation point as the evaluation criterion

$$L(\theta)=\sqrt{\frac{1}{N_\tau}\sum_{N_\tau}^{k=1}\frac{\left\|\mathbf{u}^*(\mathbf{x},t_k)-\mathbf{u}^\theta(\mathbf{x},t_k)\right\|_2^2}{mn}}, \tag{13}$$

Among them, N_τ denotes the total number of time steps, and $U^*(x,t_k)$ denotes the reference solution.

4.4 Two-Dimensional Burgers Equation

The 2D Burgers equation is mainly used to describe the incompressible flow process of viscous fluids, and is particularly capable of capturing the coupling effect between the nonlinear convection effect and the viscous diffusion effect in the flow. The equations are as follows:

$$\begin{aligned} u_t+u\nabla u_x+v\nabla u_y-\frac{1}{R}\Delta u=0,\\ v_t+u\nabla v_x+v\nabla v_y-\frac{1}{R}\Delta v=0, \end{aligned} \tag{14}$$

The analytical solution is as follows:

$$\begin{aligned} u(x,y,t)=\frac{3}{4}-\frac{1}{4\left(1+\mathrm{e}^{(R(-t-4x+4y))/32}\right)},\\ v(x,y,t)=\frac{3}{4}+\frac{1}{4\left(1+\mathrm{e}^{(R(-t-4x+4y))/32}\right)}, \end{aligned} \tag{15}$$

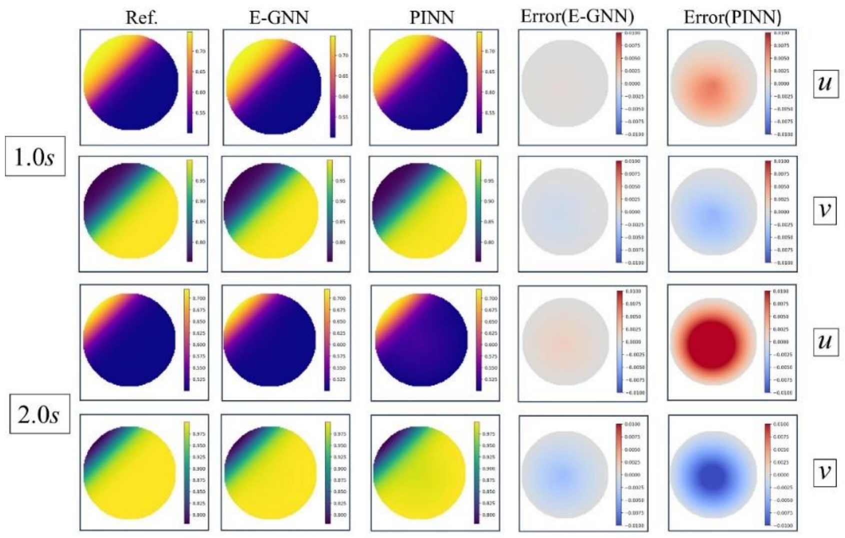

Figure 4. Prediction Solutions and Error Plots of the 2D Burgers' Equation on E-GNN and PINN

u_t and v_t are the instantaneous velocity components of the fluid in the x -direction and y -direction, $\frac{1}{R}$ is the kinematic viscosity coefficient of the fluid, which reflects the intensity of the diffusion effect of fluid viscosity on the flow.

The solution domain is $(0.5, 0.5)$ circular region with the center coordinates of a and a radius of 0.5. The time step is set to 0.05, and $R = 80$.

Table 1. Comparison of RMSE between the Predicted Solutions of E-GNN and PINN for the Burgers Equation at Various Times

Time / RMSE	$T = 1$	$T = 1.5$	$T = 2$	$T = 2.5$
E-GNN	$5.37e-04$	$1.12e-03$	$1.78e-03$	$2.44e-03$
PINN	$3.36e-03$	$6.81e-03$	$1.03e-02$	$1.33e-02$

As can be seen from Table 1 and Figure 4 with the elapse of time, E-GNN maintains a stable solution accuracy on the order of 10^-3 when solving the 2D Burgers equation, while PINN exhibits a significant accuracy degradation at $T = 2.5$ seconds during the solution process.

4.5 Two-Dimensional Convection-Diffusion Equation

The two-dimensional convection-diffusion equation can describe the process in which a physical quantity (such as concentration and temperature) in a fluid changes with time due to both fluid flow and its own diffusion. Specifically, the two-dimensional convection-diffusion equation is expressed as:

$$\frac{\partial u}{\partial t} + \mathbf{c} \cdot \nabla u = \nu \Delta u \tag{16}$$

where, u denotes the field variable, $\mathbf{c}$ represents the convective velocity vector, and ν stands for the diffusion coefficient.

The analytical solution is as follows

$$u(x, y, t) = \exp\left(-2\pi^2 \nu t\right) \cdot \sin\left(\pi(x - c_x t)\right) \cdot \sin\left(\pi(y - c_y t)\right) \tag{17}$$

The solution domain is $(0.5, 0.5)$ circular region with the center coordinates of a and a radius of 0.5. The time step is set to 0.05, and the convective velocity is $\mathbf{c}(c_x, c_y) = (0.4, 0)$, $\nu = 0.0125$.

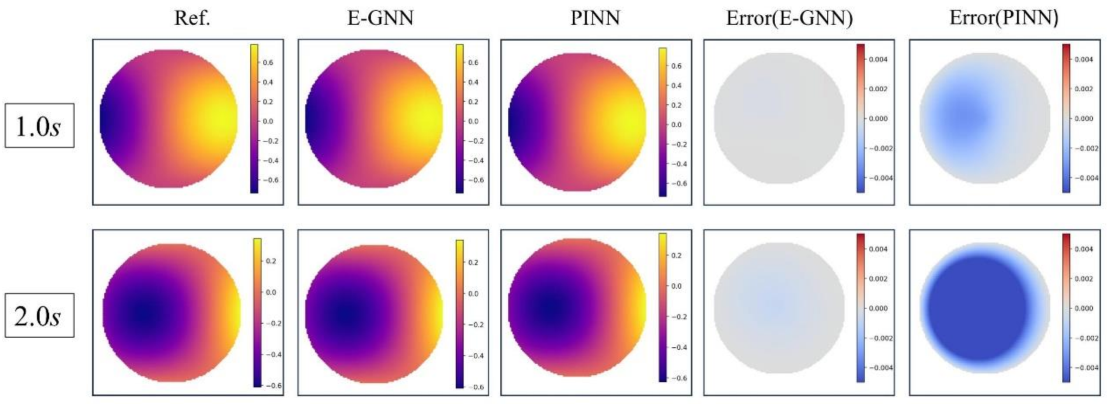

Figure 5. Prediction Solutions and Error Plots of the 2D Convection-Diffusion Equation on E-GNN and PINN

Table 2. Comparison of RMSE between the Predicted Solutions of E-GNN and PINN for the Convection-Diffusion Equation at Various Times

Time / RMSE	$T=1$	$T=1.5$	$T=2$	$T=2.5$
E-GNN	$8.18e-05$	$1.69e-04$	$3.35e-03$	$6.18e-03$
PINN	$1.51e-03$	$4.37e-03$	$8.27e-02$	$1.33e-02$

As can be seen from Table 2 and Figure 5, with the passage of time, E-GNN exhibits a significantly lower RMSE value than the PINN method when solving the 2D convection-diffusion equation. This advantage is attributed to our ESN (Echo State Network) autoregressive module. Owing to its inclusion of memory information, the ESN can effectively capture the field transport characteristics dominated by the convection term and the smooth evolution law of the diffusion term, thereby accurately describing the dynamic dependencies in the time dimension. Meanwhile, its echo state property can reduce numerical dispersion and dissipation effects, avoiding the accuracy drift of PINN caused by gradient vanishing or weakened physical constraints during long-term evolution. Consequently, E-GNN maintains stable and high-precision solution performance across the entire time domain.

5. Conclusion

These results fully verify that the E-GNN coupled network, under the constraint of physics-informed loss, can not only leverage the GNN’s capability to accurately model the geometry of irregular domains and PDE residuals but also capture the dynamic dependencies of parameter evolution through the memory characteristics of ESN. Eventually, it achieves stable and high-precision inversion of multiple types of physical parameters, providing an efficient and feasible new paradigm for solving the forward problems of PDEs.

References

[1] Raissi M, Karniadakis G E. Hidden physics models: Machine learning of nonlinear partial differential equations[J]. Journal of Computational Physics, 2018, 357: 125-141.

[2] Raissi M, Perdikaris P, Karniadakis G E. Machine learning of linear differential equations using Gaussian processes[J]. Journal of Computational Physics, 2017, 348: 683-693.

[3] Raissi M, Perdikaris P, Karniadakis G E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations[J]. Journal of Computational physics, 2019, 378: 686-707.

[4] Xu C, Cao B T, Yuan Y, et al. Transfer learning based physics-informed neural networks for solving inverse problems in engineering structures under different loading scenarios[J]. Computer Methods in Applied Mechanics and Engineering, 2023, 405: 115852.

[5] Khadijeh M, Cerqueglini V, Kasbergen C, et al. Multistage physics informed neural network for solving coupled multiphysics problems in material degradation and fluid dynamics[J]. Engineering with Computers, 2025: 1-31.

[6] Goswami S, Anitescu C, Chakraborty S, et al. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture[J]. Theoretical and Applied Fracture Mechanics, 2020, 106: 102447.

[7] Zhang R, Su J, Feng J. Solution of the Hirota equation using a physics-informed neural network method with embedded conservation laws[J]. Nonlinear Dynamics, 2023, 111(14): 13399-13414.

[8] Bai J, Rabczuk T, Gupta A, et al. A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics[J]. Computational Mechanics, 2023, 71(3): 543-562.

[9] Krishnapriyan A, Gholami A, Zhe S, et al. Characterizing possible failure modes in physics-informed neural networks[J]. Advances in neural information processing systems, 2021, 34: 26548-26560.

[10] Ren P, Rao C, Liu Y, et al. PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs[J]. Computer Methods in Applied Mechanics and Engineering, 2022, 389: 114399.

[11] Shi X, Chen Z, Wang H, et al. Convolutional LSTM network: A machine learning approach for precipitation nowcasting[J]. Advances in neural information processing systems, 2015, 28.

[12] Cao R, Su J, Feng J, et al. PhyICNet: Physics-informed interactive learning convolutional recurrent network for spatiotemporal dynamics[J]. Electronic Research Archive, 2024, 32(12).
[13] Gao H, Sun L, Wang J X. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain[J]. Journal of Computational Physics, 2021, 428: 110079.
[14] Gori M, Monfardini G, Scarselli F. A new model for learning in graph domains[C]//Proceedings. 2005 IEEE international joint conference on neural networks, 2005. IEEE, 2005, 2: 729-734.
[15] Scarselli F, Gori M, Tsoi A C, et al. The graph neural network model[J]. IEEE transactions on neural networks, 2008, 20(1): 61-80.
[16] Zhang H, Jiang L, Chu X, et al. Combining physics-informed graph neural network and finite difference for solving forward and inverse spatiotemporal PDEs[J]. Computer Physics Communications, 2025, 308: 109462.
[17] Zeng B, Wang Q, Yan M, et al. PhyMPGN: Physics-encoded Message Passing Graph Network for spatiotemporal PDE systems[J]. CoRR, 2024.
[18] Jaeger H. Echo state network[J]. scholarpedia, 2007, 2(9): 2330.
[19] Lukoševičius M, Jaeger H. Reservoir computing approaches to recurrent neural network training[J]. Computer science review, 2009, 3(3): 127-149.
[20] Ma Q, Shen L, Cottrell G W. DeePr-ESN: A deep projection-encoding echo-state network[J]. Information Sciences, 2020, 511: 152-171.
[21] Na X, Ren W, Liu M, et al. Hierarchical echo state network with sparse learning: A method for multidimensional chaotic time series prediction[J]. IEEE transactions on neural networks and learning systems, 2022, 34(11): 9302-9313.
[22] Li Y, Li K, Wang S, et al. A spatiotemporal separable graph convolutional network for oddball paradigm classification under different cognitive-load scenarios[J]. Expert Systems with Applications, 2025, 262: 125303.
[23] Zhu X, Xu J, Fu Z, et al. Novel dynamic data-driven modeling based on feature enhancement with derivative memory LSTM for complex industrial process[J]. Neurocomputing, 2025, 626: 129619.