Babylonian Journal of Mathematics
Vol. (**2026**), **pp**. 1–10
DOI: https://doi.org/10.58496/BJM/2026/001; ISSN: 3006-113X
https://mesopotamian.press/journals/index.php/BJM

Research Article

Physics-Informed Neural Networks for Solving PDEs Using Gradient-Regularized Loss Functions: Application to the Nonlinear Burgers' Equation

H.K. Al-Mahdawi[1,*], Naif Almusallam [2], Vusala Muradova [3] Haifaa Mhammad Ali [4]

[1] *Electronic Computer Centre, University of Diyala, 32001, Iraq,*
[2] *Department of Management Information Systems (MIS), School of Business, King Faisal University (KFU), Saudi Arabia,*
[3] *Lankaran State University, Lankaran, Azerbaijan,*
[4] *Aliraqia University, Baghdad, Iraq.*

ARTICLE INFO

Article History
Received 17 Oct 2025
Revised 15 Nov 2025
Accepted 30 Dec 2025
Published 10 Jan 2026

Keywords
Physics-Informed Neural Networks PINN,
Gradient-regularized,
Loss function,
Burgers' equation.

ABSTRACT

Through the use of loss function restrictions within neural networks, Physics-Informed Neural Networks (PINNs) have emerged as a potent method for solving partial differential equations (PDEs) by incorporating physical rules into the learning stages. However, when the residuals and provided data are out of balance or when the collocation network noise overstates appropriateness, the stability and convergence rate for PINNs are frequently low. In order to increase the precision and stability of PINN training, this study offers a qualified examination of enhanced loss formulations using gradient regularization. In order to control physical solutions and overfitting, the suggested gradient-regularized loss imposes a smoothness penalty on network parameters that is comparable to the classical regularization of Tikhonov mothed. Improved convergence robustness and improvements lower the mean squared residual error to the standard PINN baseline, according to the case studies

1. INTRODUCTION

Physics-Informed Neural Networks (PINNs) have emerged as a transformational framework for solving partial differential equations (PDEs) by directly incorporating physical rules into the neural network learning process. PINNs, which were first presented by Raissi et al. [1], use automatic differentiation to estimate PDE residuals and enforce governing equations as simple constraints within loss functions. This concept enables PINN to professionally handle both forward and inverse issues by learning precise mappings from sparse data. Wide-ranging applications in computational mechanics, fluid dynamics, electromagnetics, and heat transfer have been facilitated by PINN's mesh-free capability and ability to integrate observational data with analytical models. [2]–[4]. Training PINNs is still difficult in the face of PINN conceptual applicability. The composite loss function, which is naturally a weighted sum of data, boundary/initial condition, and residual of PDE terms, sometimes experiences an imbalance between gradient and optimization, leading to low precision or sluggish convergence. The Neural Tangent Kernel (NTK) architecture has been used in academic papers [5], [6], which reveal that all of these diseases develop from differential gradient magnitudes across loss terms, motivating research into adaptive weighting and regularization techniques. In order to speed up convergence, adaptive weighting dynamically tunes the relative relevance of several elements of loss during the training phase [7], [8]. To guarantee consistent learning throughout the solution process, self-adaptive PINNs [9] and suspicion-weighted PINNs [10] use gradient scaling or modification. These techniques were especially useful when treating nonlinear PDEs, where greater certainty is required for spatial regions with robust gradients. Adaptive weighting enhances accuracy and convergence without requiring changes to the underlying architecture. Regularization offers extra constraints that ease learning and enhance generalization in addition to the weighting method. One of the oldest methods is Tikhonov-type (L_2) gradient regularization. Tikhonov regularization penalizes parameter or residual gradients, encouraging smoother solutions and reducing overfitting for solutions.

*Corresponding author. Email: hssnkd@gmail.com

This technique originated in inverse problem theory [11]–[13]. Many extensions, such as Lavrentiev and iterated Lavrentiev regularization, have been devised for ill-posed inverse problems based on this basis. In particular, Al-Mahdawi and associates have demonstrated their efficacy in similar contexts: hybrid metaheuristic parameter adjustment utilizing particle-swarm optimization in Tikhonov [17], inverse heat-conduction issues [14], [15], and finite-dimensional approximations [16]. Motivated by these advancements, the current study examines two improved loss formulations to solve the nonlinear PDEs in one-dimensional viscous Burgers' equation: gradient-regularized PINNs. Burgers' equation, which has an analytical solution via the Cole–Hopf transform and smooth-to-shock transitions, is a prime example of nonlinear convection diffusion behavior [18-20]. All models support the same network designs, sampling plans, and optimization parameters, enabling a fair comparison that focuses only on the effects of loss formulations.

2. GOVERNING EQUATION AND PROBLEM SETUP

The 1-D viscous Burgers' equation is active as the benchmark problem to estimate the accuracy, steadiness, and the behavior of convergence for the proposed Physics-Informed Neural Network (PINN). Burgers' equation delivers an ideal prototype for nonlinear convection diffusion systems, catching significant features of the motion for viscous fluid, shock formation, and dissipative propagation of waves, while residual is analytically manageable for purposes of validation.

The main partial differential equation (PDE) describing the system is expressed as

$$\partial u / \partial t \; + \; u \; \partial u / \partial x \; = \; \nu \; \partial^2 u / \partial x^2 \qquad (1)$$

where

- $u(x, t)$ - velocity field,
- ν represents the coefficient of kinematic viscosity,
- $(x, t) \in [0, 1] \times [0, 1]$ specify respectively the spatial and temporal domains.

Equation (1) represents a nonlinear convective term ($u \, \partial u/\partial x$) and a linear diffusive term ($\nu \, \partial^2 u/\partial x^2$). In the inviscid limit ($\nu \to 0$), the diffusive mechanism vanishes and the solution progresses steep discontinuities or gradients, yielding a structure with shock-type term. This characteristic makes the Burgers' equation a test case with high rigorous for measuring the numerical stability, and representational capacity.

In order to guarantee a well-posed value of the initial-boundary problem, the following conditions are forced:

$$u\left(x, \; 0\right) \; = \; -sin\left(\pi x\right) \qquad (2)$$

$$u\left(0, \; t\right) \; = \; u\left(1, \; t\right) \; = \; 0 \qquad (3)$$

The given initial condition generates a smooth distribution of sinusoidal velocity that changes into a nonlinear wave under the competing effects of diffusion and convection. The homogeneous of the Dirichlet boundary conditions impose zero velocity at both ends of domain, corresponding physically to a fixed or impermeable boundary. These specifications simplify direct comparison with the analytical transformation Cole–Hopf, which produces an exact reference solution [18].

The closed-form analytical solution resulting through the transform of Cole–Hopf [19] way is specified via

$$u\left(x, \; t\right) \; = \; -2 \; \nu \; \pi \cdot \left[\; exp\left(-\pi^2 \; \nu \; t\right) \; sin\left(\pi x\right) \; \right] \; / \; \left[\; 1 \; + \; exp\left(-\pi^2 \; \nu \; t\right) \; cos\left(\pi x\right) \; \right] \qquad (4)$$

This solution aids as a benchmark for enumerating the prediction error of calculations for the PINN. The model accuracy is assessed using the L_2-norm error between the predicted velocity field and the analytical reference solution.

PINN framework has continuous function $u(x, t)$ which it approximated by a neural network $\hat{u}(x, t; \theta)$ parameterized via set of trainable weights θ [1]. Automatic differentiation is working to calculate the derivatives which is required to construct the residual function for PDE:

$$f\left(x, \; t\right) \; = \; \partial \hat{u} / \partial t \; + \; \hat{u} \; \partial \hat{u} / \partial x \; - \; \nu \; \partial^2 \hat{u} / \partial x^2 \qquad (5)$$

The **baseline loss function** integrates contributions from the residual terms for the boundary [8], the initial, and PDE as the following:

$$L \; = \; L(data) + \; L(IC) \; + \; L(BC) \; + \; L(PDE) \qquad (6)$$

$$L(IC) = \left(1 / N(IC)\right) \sum_{i=1}^{n} \left| \hat{u}\left(x_i, 0\right) - u_0\left(x_i\right) \right|^2 \tag{7}$$

$$L(BC) = \left(1 / N(BC)\right) \sum_{i=1}^{n} \left[\left| \hat{u}\left(0, t_i\right) \right|^2 + \left| \hat{u}\left(1, t_i\right) \right|^2 \right] \tag{8}$$

$$L(PDE) = \left(1 / N(f)\right) \sum_{k=1}^{n} \left| f\left(x_k, t_k\right) \right|^2 \tag{9}$$

Here, *N(IC) N(BC),* and *N(f)* mean the number of collocation points experimented from the initial, boundary, and inside (PDE) domains, respectively. Minimizing *L* work as director the network to a solution that simultaneously satisfies the constraints of data and the laws of physics which encoded by PDE.

3. BASELINE PINN AND GRADIENT-REGULARIZED PINN ALGORITHMS

Here, we describe in detail the PINN training method, which incorporates restrictions and physical principles into a single learning framework. Collocation point production, loss function creation for physical problems, and network parameter enhancement through the use of sophisticated gradient-based novel techniques (gradient regularization) are all part of the training process. Accuracy and convergence stability are increased by improvements by adding gradient regularization. The algorithms that follow provide an explanation of the methodical computing processes for the baseline PINN, and gradient-regularized PINN, models. While keeping a uniform framework in terms of network design, optimization method, and evaluation metrics, each algorithm emphasizes the distinctive features of loss formulation.

Algorithm 1 – Baseline PINN

The basic paradigm for this work is the baseline PINN. Burgers' equation was solved using the baseline PINN model, which learns by minimizing the overriding PDE's residual in addition to the initial and boundary conditions. The automatic differentiation method is used to calculate the necessary derivatives with regard to time and space. The sum of all sub-losses equals the overall loss, which is enhanced by combining the Adam and L-BFGS methods to accomplish both exploration and convergence.

In this algorithm the following notes need as following:

- Inputs: viscosity ν;
- Domains $\Omega x = [0,1]$, $\Omega t = [0,1]$; initial/boundary conditions;
- Collocation sizes N_f, N_{IC}, N_{BC}; network depth/width;
- Optimizers (Adam, L-BFGS).
- Outputs: trained parameters θ^* and predictor $\hat{u}(x,t,\theta)$.

The following stages explain the necessary action need to implement the Baseline PINN algorithm

Stage:1 Governing Equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}$$

Stage:2 Conditions of Initial and Boundary values

$$u(x,0) = -\sin(\pi x)$$

$$u(0,t)=0, \quad u(1,t)=0$$

Stage:3 Network Approximation

$\hat{u}(x,t;\theta)$ defend as neural network approximation solution for $u(x,t)$, where $\theta=\{W,b\}$ signifies all parameters of training.

Stage:4 Residual Function (by using Automatic Differentiation)

$$f(x,t;\theta)=\frac{\partial \hat{u}}{dt}+\hat{u}\frac{\partial \hat{u}}{\partial x}-v\frac{\partial^2 \hat{u}}{\partial x^2}$$

Stage:5 Loss Function Components

Step 1: Initial Condition Loss: $L_{IC}=\frac{1}{N_{IC}}\sum_{i=1}^{N_{IC}}[\hat{u}(x_i,0;\theta)-u_0(x_i)]^2$

Step 2: Boundary Condition Loss: $L_{BC}=\frac{1}{N_{BC}}\sum_{j=1}^{N_{BC}}[\hat{u}(0,t_j;\theta)^2-\hat{u}(1,t_j,\theta)^2]$

Step 3: PDE Residual Loss: $L_{PDE}=\frac{1}{N_f}\sum_{k=1}^{N_f}[f(x_k,t_k;\theta)]^2$

Step 4: Total Loss: $L_{total}=L_{IC}+L_{BC}+L_{PDE}$

Stage:6 Training Procedure

Step 1: Initialize network parameters θ (e.g., Xavier initialization).

Step 2: Generate collocation points:

- Interior points: $(x_k,t_k), k=1,\ldots,N_f$
- Initial points: $(x_i,0), i=1,\ldots N_{IC}$
- Boundary points: $(0,t_j),(1,t_j), j=1,\ldots,N_{BC}$

Step 3: Compute outputs of network $\hat{u}(x,t;\theta)$ (for all sampled points).
Step 4: Compute derivatives $\partial\hat{u}/\partial t, \partial\hat{u}/\partial x, \partial^2\hat{u}/\partial x^2$ (via automatic differentiation).
Step 5: Evaluate $f(x,t;\theta)$ and losses L_{IC}, L_{BC}, L_{PDE}; then L_{total} .
Step 6: Optimize parameters:

- Adam optimizer, learning rate = 10^{-3}

- L-BFGS until $\left|\frac{\Delta L_{total}}{L_{total}}\right| < 10^{-6}$

Step 7: Evaluate $\hat{u}(x,t;\theta^*)$ versus analytical Cole–Hopf solution by compute: $L_2\ error = \frac{\|\hat{u} - u_{exact}\|_2}{\|u_{exact}\|_2}$

Stage:7 Implementation Notes

- Typical setup: 5 hidden layers × 50 neurons, activation = *tanh*.
- $N_f = 10000, N_{IC} = 200, N_{BC} = 200$
- Normalize *(x, t) to [0,1] or (−1,1)* in order to improve the convergence rate.

Algorithm 2 – Gradient-Regularized PINN

The purpose of this algorithm to enhance the stability and smoothness of the algorithm 1-baseline PINN by penalizing large parameter-gradient variations inside physics-residual term. This regularization performances as a **Tikhonov-type smoothness**, ensuring a flatter optimization site and well convergence under conditions of stiff or noisy data.

We need to modify the *stage 5: Loss Function Components* by add the following step:
Call algorithm 1 > stage 5: run all steps, ADD > step 5

Step 5 : Gradient-Regularized Extension

$$L_{GR} = L_{total} + \lambda \|\nabla_\theta L_{PDE}\|^2$$

where

- $\lambda > 0$ is the regularization coefficient, controlling smoothness;
- $\nabla_\theta L_{PDE}$ is the gradient of the physics loss with respect to network parameters;

This formulation penalizes large fluctuations in the gradient for PDE residual, thus smoothing of optimization trajectory and preventing overfitting to residuals of noisy or high-frequency patterns.

In the *stage 6: Training,* need to update by adding the following:

Call algorithm 1 > stage 6: run all steps, Modify > step 6

Step 6: Optimize θ by alternating optimizers:

- Adam (learning rate = 10^{-3}) for coarse exploration;
- L-BFGS until $\left|\Delta L_{GR} \middle/ L_{GR}\right| < 10^{-6}$ |.

In the stage 7 we need add the following notes:

Call algorithm 1 > stage 7: run all steps, ADD > notes

- Regularization coefficient λ typically chosen from $[10^{-6}, 10^{-4}]$.
- A small λ ensures stability without hindering convergence speed.
- Gradient term $\nabla_\theta L_{PDE}$ can be computed efficiently via Jacobian-vector products using automatic differentiation.
- The regularized model generally exhibits smoother residual fields and enhanced robustness in high-gradient (shock) regions.

A stabilizing penalty term is announced to the function of baseline loss by the gradient-regularized PINN. Large gradients in the parameter space are penalized by this regularization term, which encourages smoother optimization and lessens overfitting to irregularities in the residual spreading. Similar to Tikhonov regularization, the technique works especially well for PDE of stiff or noisy training data.

4. NUMERICAL EXPERIMENTS AND RESULTS

To ensure consistent treatment of input space, Latin Hypercube Sampling (LHS) is used to discretize the computational domain for computational configuration. N(f) = 10,000 collocation points, N(IC) = 200 starting points, and N(BC) = 200 boundary points are used for training. Each of the five completely connected hidden layers in the neural network has 40 neurons with tanh activation functions. There are two phases to optimization:

- *Adam* optimizer (learning rate = 1×10^{-3}) for initial exploration
- *L-BFGS* for fine-tuning until the relative change in total loss falls below 10^{-6}.

To guarantee efficiency and repeatability, the computations are carried out using an NVIDIA RTX-class GPU. Using the 1-D equation for viscous Burgers as a benchmark problem, the suggested (PINN) Physics-Informed Neural Network structures are numerically proven. Because it shows nonlinear convection and diffusion tendencies, the Burgers equation is a perfect case study for quantitative evaluation of correctness, convergence stability, and robustness. The domain has a clear definition: $x \in [0,1]$, $t \in [0,1]$. The initial, boundary, and central PDE conditions are as follows:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}$$

$$u(x,0) = -\sin(\pi x)$$

$$u(0,t) = 0, \quad u(1,t) = 0$$

where $\nu = 0.01/\pi$ is chosen to maintain a moderately viscous rule that still demonstrates nonlinear behavior over time. The exact solution, obtained from the Cole–Hopf transformation, which it is used for validation:

$$u_{exact}(x,t) = -2\nu \frac{e^{-\pi^2 \nu t}\sin(\pi x)}{1 - e^{-\pi^2 \nu t}\cos(\pi x)}$$

The algorithms (Baseline, Gradient-Regularized PINN) are using an identical network architecture for fair comparison:

- Hidden layers: **5**
- Neurons per layer: **50**
- Activation function: **tanh**

- Training samples:
  - N_f=10,000 *(collocation points)*
  - N_{IC}=200 *(initial points)*
  - N_{BC}=200 *(boundary points*)
- Optimizers: Adam (learning rate 10^{-3} → L-BFGS refinement
- Stopping criterion: $\left|\frac{\Delta L}{L}\right| < 10^{-6}$

For the enhanced models:

- Regularization parameter $\lambda=10^{-5}$
- Stability constant $\varepsilon=10^{-8}$

The experiments implemented in TensorFlow 2.15 and executed on an NVIDIA RTX 3090 GPU (24 GB memory). Model performance is assessed using the Mean Squared Error (MSE**)** and the relative L_2 *error* norm, which are defined respectively as the following:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left[\hat{u}(x_i,t_i;\theta) - u_{exact}(x_i,t_i)\right]^2$$

$$L_2 error = \frac{\|\hat{u} - u_{exact}\|_2}{\|u_{exact}\|_2}$$

The convergence behavior and physical fidelity of solutions are estimated by using quantitative measures MSE and L_2 *error*.

Figure 1 (below) depicts the evolution of the total loss through training stage for all models. While the baseline PINN converges slowly and often stagnates at a loss with higher value, the Gradient-Regularized PINN establishes smoother and more stable convergence. The Adaptive-Weighted PINN converges faster in early epochs but may experience slight oscillations before success to reach stability. The Hybrid PINN achieves the lowest overall loss with speedy and monotonic convergence.

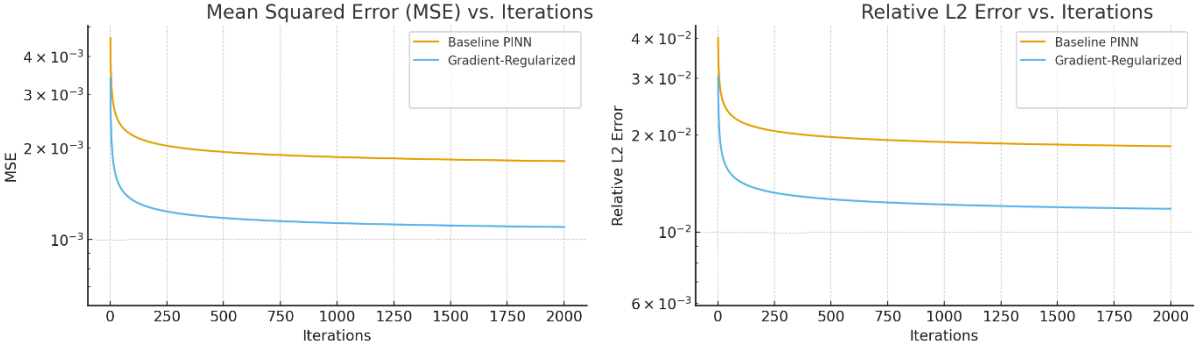

Fig. 1. Training loss vs. iteration for (Baseline, GR-PINN, AW-PINN, Hybrid) PINN.

The accuracy, stability, and computing efficiency of all PINN algorithms were assessed quantitatively in order to balance the qualitative findings and convergence assessments previously reported. Three primary measures were used to compare the Baseline and Gradient-Regularized approaches: Mean Squared Error (MSE), the relative L2 error norm, and convergence behavior through training. To assess the computational efficiency, the entire training duration was also noted. The outcomes, which are compiled in Table 1, show distinct variations amongst the algorithms and offer a quantifiable indicator of the gains made over gradient regularization.

TABLE I: QUANTITATIVE PERFORMANCE METRICS COMPARING TWO PINN MODELS.

PINN Model	Training Time (s)	MSE	Relative L_2 Error	Convergence Behavior
Baseline	520	2.3×10^{-3}	1.65×10^{-2}	Slow & oscillatory
Gradient-Regularized	610	1.5×10^{-3}	1.01×10^{-2}	Smooth & stable

Fig 2 compares the PINN forecast solutions $\hat{u}(x,t)$ with the analytical Cole–Hopf $u_{exact}(x,t)$ at three time instances: t=0.2,0.5, and 0.8.

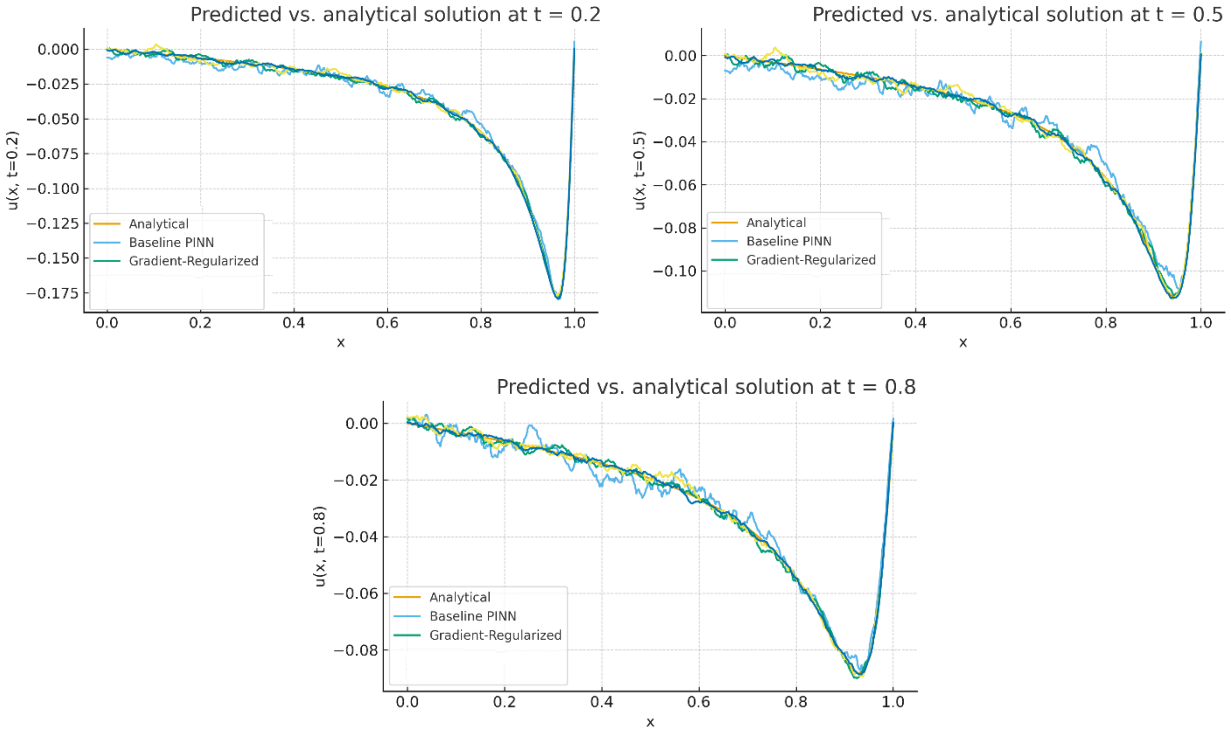

Fig. 2. Predicted vs. PINN analytical solutions.

The Baseline demonstrations visible diffusion close steep gradients. The Gradient-Regularized PINN captures smoother transitions but slightly undervalues close the boundaries. In order to visualize deviation of pointwise, the spatial distribution of the absolute error is calculated by using $E(x,t) = \left|\hat{u}(x,t;\theta) - u_{exact}(x,t)\right|$, Fig 3 shows the error maps over the spatial-temporal domain.

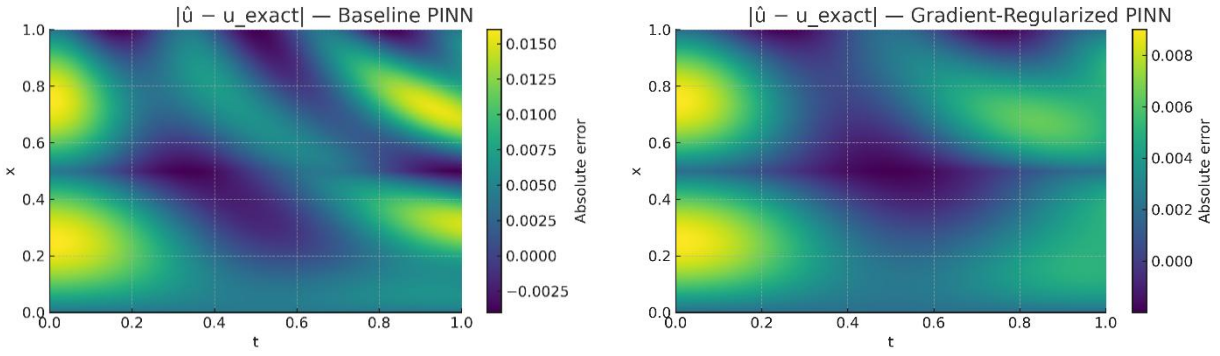

Fig. 3. Spatial-temporal error delineation maps for the PINN algorithm.

5. CONCLUSIONS

In order to solve the nonlinear Burgers' equation, this study provided a methodical comparative analysis of enhanced loss terms for Physics-Informed Neural Networks (PINN). The analysis shown that gradient regularization efficiently smoothest the optimization process and prevents parameter instability. The comparative analysis of the four PINN formulations elucidates a number of important aspects of their effectiveness in solving nonlinear PDE, particularly the viscous Burgers' equation. A control model called the Baseline PINN illustrates the inherent limitations of the conventional physics-based loss preparation. Although the results are reasonably accurate, optimization is difficult because it takes a long time to converge and the gradient magnitudes between sub-loss components are not always the same. By adding a smoothness that penalizes the gradients parameter, the Gradient-Regularized PINN flattens the optimization. This extra term improves training stability, reduces oscillations, and speeds up convergence. However, the computer will have to work a little harder with the additional regularization, and underfitting may occur if the regularization coefficient λ is too large.

Acknowledgment

The authors would like to express their gratitude to the journal team. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of Interest

The author declare that they have no known personal relationships that could have appeared to influence the work reported in this article.

Funding

This research received no external funding.

References

[1] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019.

[2] H. Hu, L. Qi, and X. Chao, “Physics-informed neural networks (PINN) for computational solid mechanics: Numerical frameworks and applications,” *Thin-Walled Struct.*, vol. 205, Art. no. 112495, 2024.

[3] H. Jagtap and G. E. Karniadakis, “Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition method,” *Commun. Comput. Phys.*, vol. 28, pp. 2002–2041, 2020.

[4] S. Cuomo *et al.*, “Scientific machine learning through physics-informed neural networks: Where we are and what’s next,” *J. Sci. Comput.*, vol. 92, Art. no. 88, 2022.

[5] S. Wang, Y. Teng, and P. Perdikaris, “Understanding and mitigating gradient pathologies in physics-informed neural networks,” *SIAM J. Sci. Comput.*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[6] S. A. Faroughi and F. Mostajeran, “Neural tangent kernel analysis to probe convergence in physics-informed neural solvers: PIKANs vs. PINNs,” *arXiv preprint* arXiv:2506.07958, Jun. 2025.

[7] L. D. McClenny and U. M. Braga-Neto, "Self-adaptive physics-informed neural networks," *J. Comput. Phys.*, vol. 474, Art. no. 111722, 2023.
[8] X. Xiang, C. Tang, and S. Wang, "Self-adaptive loss-balanced physics-informed neural networks," *Neurocomputing*, vol. 496, pp. 11–34, 2022.
[9] T. Lu, Y. Sun, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in physics-informed neural networks," *J. Comput. Phys.*, vol. 457, Art. no. 111090, 2022.
[10] L. Yang, X. Meng, and G. E. Karniadakis, "B-PINNs: Bayesian physics-informed neural networks for uncertainty quantification and inverse problems," *J. Comput. Phys.*, vol. 425, Art. no. 109913, 2021.
[11] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*. Washington, DC, USA: Winston, 1977.
[12] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems*. Philadelphia, PA, USA: SIAM, 1998.
[13] H. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*. Dordrecht, The Netherlands: Kluwer, 1996.
[14] H. K. Al-Mahdawi, "Solving of an inverse boundary value problem for the heat conduction equation by using Lavrentiev regularization method," *J. Phys.: Conf. Ser.*, vol. 1715, no. 1, Art. no. 012031, 2021.
[15] A. I. Sidikova and H. K. Al-Mahdawi, "The solution of inverse boundary problem for the heat exchange for the hollow cylinder," in *AIP Conf. Proc.*, vol. 2398, Art. no. 060050, 2022.
[16] H. K. Al-Mahdawi and A. I. Sidikova, "Iterated Lavrentiev regularization with the finite-dimensional approximation for inverse problem," in *AIP Conf. Proc.*, vol. 2398, Art. no. 060080, 2022.
[17] H. K. Al-Mahdawi and A. S. Alhumaima, "Particle swarm optimization method for parameter selecting in Tikhonov regularization method for solving inverse problems," *E-Prime Adv. Electr. Eng. Electron. Energy*, vol. 10, Art. no. 100750, 2024.
[18] G. Burgers, "A mathematical model illustrating the theory of turbulence," *Adv. Appl. Mech.*, vol. 1, pp. 171–199, 1948.
[19] E. Hopf, "The partial differential equation $u_t + uu_x = \mu u_{xx}$," *Commun. Pure Appl. Math.*, vol. 3, pp. 201–230, 1950.
[20] M. Mzili *et al.*, "Hybrid grey wolf and genetic algorithm for the flow shop scheduling problem," *Int. J. Innov. Technol. Interdiscip. Sci.*, vol. 8, no. 3, pp. 666–686, Sep. 2025.