Black Sea Journal of Engineering and Science
doi: 10.34248/bsengineering.1645207

Open Access Journal
e-ISSN: 2619 – 8991

Research Article
Volume 8 - Issue 4: 970-981 / July 2025

# THE REFINED PHYSICS-INFORMED NEURAL NETWORKS FOR NONLINEAR CONVECTION-REACTION-DIFFUSION EQUATIONS USING EXPONENTIAL SCHEMES

**Burhan BEZEKCI**[1]

[1]*Kilis 7 Aralik University, Faculty of Engineering and Architecture, Department of Electrical - Electronics Engineering, 79000, Kilis, Türkiye*

**Abstract:** The nonlinear convection-reaction-diffusion equations model complex real-world phenomena across scientific and engineering disciplines. However, solving these equations analytically is often impossible due to their nonlinear nature. As a result, researchers have turned to numerical and computational methods to find approximate solutions. These methods, while effective, can struggle with issues such as stability, accuracy, and the ability to handle sharp gradients or complex interactions between convection, diffusion, and reaction terms. To address these challenges, this work introduces an enhanced Physics-Informed Neural Network (PINN) framework for convection-reaction-diffusion equations that incorporates exponential finite difference scheme residuals with the aim of enhancing solution accuracy and stability. To validate its performance, the framework has been tested on four well-known nonlinear partial differential equations: Burgers' Equation, Fisher's Equation, the Burgers-Huxley Equation, and the Newell-Whitehead-Segel Equation. The results obtained using the modified PINN framework are systematically compared with those obtained using traditional Physics-Informed Neural Networks and the Galerkin Finite Element Method. The comparisons reveal that the proposed framework consistently outperforms both approaches in terms of accuracy. These improvements highlight the effectiveness of integrating exponential finite difference scheme residuals into the PINN framework, making it a powerful and reliable tool for solving nonlinear convection-reaction-diffusion equations.

**Keywords:** Deep learning, Exponential scheme, Convection-reaction-diffusion, Burgers-Huxley, Newell-Whitehead-Segel

***Corresponding author**: Kilis 7 Aralik University, Faculty of Engineering and Architecture, Department of Electrical - Electronics Engineering, 79000, Kilis, Türkiye
**E mail:** burhanbezekci@kilis.edu.tr (B. BEZEKCI)
Burhan BEZEKCİ https://orcid.org/0000-0001-7460-4091

**Received:** February 26, 2025
**Accepted:** April 4, 2025
**Published:** July 15, 2025

**Cite as:** Bezekci B. 2025. The refined physics-informed neural networks for nonlinear convection-reaction-diffusion equations using exponential schemes. BSJ Eng Sci, 8(4): 970-981.

## 1. Introduction

Nonlinear partial differential equations (PDEs) are essential mathematical tools for describing dynamic processes in a wide range of scientific and engineering fields. They are used extensively in areas such as fluid dynamics, plasma physics, environmental modeling, and mathematical biology, where they capture complex interactions and evolving spatiotemporal patterns. Among these, nonlinear convection-reaction-diffusion (CRD) equations are especially important because they describe systems influenced by the combined effects of diffusion, convection, and reaction mechanisms. These equations arise in many real-world applications, including chemical reaction modeling, environmental pollution transport, heat transfer in engineering systems, and the dynamics of biological populations (Nekhamkina and Freed 2000; Morton 1996; Murray 2007; Bezekci 2025; Bezekci 2025; Seinfeld and Pandis 1998; Bejan 2013; Dong et al. 2022). A general mathematical form of these equation 1 can be expressed as;

$$\frac{\partial u}{\partial t} = \nabla \cdot [D\nabla u] - \nabla \cdot (vu) + S \qquad (1)$$

where $u$ represents the variable of interest, such as species concentration in mass transfer or the temperature in heat transfer. The term $v$denotes the velocity field, while $D$ is the diffusivity coefficient. The symbol $\nabla$ represents the gradient operator, and $S$ accounts for any sources or sinks, describing the generation or depletion of $u$. To ensure a well-posed problem, appropriate initial conditions and boundary conditions must be defined, as they govern the behavior of $u$ within the domain.

Analytical solutions to CRD equations are rarely possible due to their nonlinear nature and the complexity of their dynamics, except in a few simplified cases. Consequently, numerical methods have become the primary focus of research, with many significant contributions made to the field. Among these, total variation diminishing (TVD) schemes are well-known for balancing stability and higher-order accuracy. Methods using limiters, such as TVD-Superbee, TVD-Minmod, TVD-MUSCL, and TVD-

Koren, have been applied successfully to a variety of challenging problems (Barth and Jespersen 1989; Harten 1997; van Leer 1979; Koren 1993). Other notable approaches include hybrid nodal-integral/finite element methods (John and Schmeyer 2008), high-order finite volume methods (Costa et al. 2018), and characteristic-based finite volume element methods (Phongthanapanich and Dechaumphai 2008). Many of these methods incorporate advanced features such as positivity-preserving schemes, higher-order reconstructions, and meshfree frameworks (Peddavarapu and Srinivasan 2021), making them versatile tools for addressing nonlinear CRD equations in complex applications.

Although these numerical methods have significantly broadened the use of CRD models, they come with certain limitations. For instance, TVD schemes can produce numerical dissipation near sharp discontinuities, and many approaches are computationally demanding or difficult to implement, especially when working with non-uniform meshes or strongly nonlinear systems. Even with these challenges, CRD models remain essential for studying complex systems, driving ongoing improvements in numerical techniques to meet the growing demands of advanced applications.

Recent developments in scientific computing have introduced Physics-Informed Neural Networks (PINNs) that offer a powerful framework for solving PDEs by embedding governing physical laws directly into the neural network's loss function (Raissi et al. 2019; Baydin et al. 2018). By using automatic differentiation to enforce constraints such as initial and boundary conditions, PINNs avoid the need for meshing or large labeled datasets, enabling accurate solutions even with sparse data (Baydin et al. 2018; Raissi et al. 2019). They are effective for both forward problems, predicting system evolution, and inverse problems, inferring unknown parameters, making them well-suited for nonlinear CRD equations and other PDEs in fields like fluid dynamics, heat transfer, and biomechanics (Karniadakis et al. 2021; Jagtap et al. 2022; Sahli Costabal et al. 2020).

Their ability to reduce computational complexity and handle high-dimensional domains has made PINNs a modern alternative to conventional solvers, enabling solutions to previously intractable problems (Hennigh et al. 2021). Open-source libraries like DeepXDE (Lu et al. 2021) and SimNet (Hennigh et al. 2021) have further supported their adoption, driving advancements in scientific computing across disciplines.

The exponential finite difference method is a robust numerical technique for solving PDEs, particularly in cases where conventional finite difference methods struggle. First introduced by Bhattacharya for solving heat conduction problems (Bhattacharya 1985), this method has been extended to tackle more complex scenarios, including nonlinear systems, steep gradients, and stiff equations (Bahadir 2005; Handschuh and Keith 1992). Unlike standard finite difference schemes, which use linear approximations for derivatives, the exponential finite difference method incorporates an exponential formulation to better capture rapid variations in the solution. This approach enhances both the accuracy and stability of the method, making it highly effective for problems characterized by sharp transitions or highly nonlinear behaviors. By delivering accurate approximations while maintaining computational efficiency, the exponential finite difference method has emerged as a robust and versatile alternative to traditional numerical approaches. In this paper, we aim to enhance the accuracy and reliability of PINNs by introducing an additional residual term derived from a variation of the exponential finite difference scheme. This modification embeds the strengths of the exponential scheme into the standard PINN framework, allowing the network to better capture the behavior of the underlying equations. By combining the traditional PDE residual with the exponential scheme-based residual, the proposed approach seeks to improve solution accuracy and ensure closer alignment with analytical results.

The study conducted by Ali et al. (2022) represents an important contribution to the numerical analysis of nonlinear convection-reaction-diffusion equations, utilizing the Galerkin Finite Element Method (GFEM) to address four representative cases: Burgers Equation, Fisher's Equation, Burgers-Huxley Equation, and the Newell-Whitehead-Segel Equation. Their research established critical benchmarks for solution accuracy and computational performance, providing a foundational reference for subsequent methodologies. Building on this foundation, Hasan et al. (2024) investigated the application of the traditional PINN to the same set of equations, showcasing its potential to surpass the GFEM in accuracy.

In this work, we build on these previous studies by considering the same test problems as analyzed in Ali et al. (2022) and Hasan et al. (2024), using identical initial and boundary conditions, as well as model parameter values. This consistency provides a solid foundation for comparing our results with those obtained from the GFEM and the traditional PINN framework. By integrating an additional residual term based on a modified exponential finite difference scheme, we develop an improved PINN approach. The goal is to improve solution accuracy by applying the modified framework to the same nonlinear equations and comparing its effectiveness with previously reported results.

## 2. Materials and Methods

Without loss of generality, we consider a simplified case of the general convection-reaction-diffusion equation, where the variable of interest is a scalar u=u defined in a one-dimensional spatial domain $\nabla=\partial/\partial\_x$. The diffusivity coefficient is assumed to be a scalar constant D=d, and the velocity field v can be a constant, a function of space and time, or may depend on the solution u itself, as in the case of the Burgers' equation, one of our test problems. The source or sink term S(u,x,t), also a scalar function, which may depend on u, space, and time. Under these assumptions, the governing equations 2-4 can be expressed as

$$\partial u/\partial t = d\,(\partial^2 u)/(\partial x^2) - v(u,x,t)\,\partial u/\partial x + S(u,x,t) \qquad (2)$$

subject to the initial and boundary conditions given by

$$u(x,0) = u_0(x), x \in \Omega \qquad (3)$$

$$B[u] = b(x,t), (x,t) \in \partial\Omega \times R_+. \qquad (4)$$

Here, B[u] represents the boundary operator, which may take different forms depending on the type of boundary condition. The term b(x,t) represents the boundary condition and can vary based on the physical problem, taking forms such as zero (for homogeneous or no-flux conditions) or a function of the spatial variable x and the initial condition $u_0(x)$ provides the initial distribution of the variable u across the domain.

In the following, we explain how the exponential scheme is applied to solve the governing equation (2) along with the initial (3) and boundary conditions (4). Next, we outline the PINN framework, which embeds the physical laws of the problem into the neural network training process. Finally, we combine the exponential scheme with the PINN framework to create a unified computational approach that takes advantage of both methods to solve the problem effectively and accurately.

### 2.1. Exponential Scheme for CRD Equation

To apply the exponential scheme to the one component one dimensional CRD equation (2), we proceed as follows. The governing equation is transformed into a logarithmic form to facilitate numerical stability and better handling of the nonlinearities. Let F(u)=lnu, so that $\partial F/\partial u = 1/u$. Multiplying the CRD equation 5 by $\partial F/\partial u$results in

$$1/u\,\partial u/\partial t = d\,1/u\,(\partial^2 u)/(\partial x^2) - v(u,x,t)\,1/u\,\partial u/\partial x + 1/u\,S(u,x,t) \qquad (5)$$

Here, the term $1/u\,\partial u/\partial t$ represents the temporal evolution in the transformed space, with the remaining terms scaled by 1/u. For temporal discretization, a forward Euler scheme is employed, where the time derivative $\partial u/\partial t$is approximated as (equation 6);

$$1/u^k\,\partial u/\partial t \approx (\ln u^{(k+1)} - \ln u^k)/\Delta t. \qquad (6)$$

Substituting this into the transformed equation, we get $(\ln u^{(k+1)} - \ln u^k)/\Delta t = d\,1/u^k\,(\partial^2 u^k)/(\partial x^2) - v(u^k,x,t)\,1/u^k\,(\partial u^k)/\partial x + 1/u^k\,S(u^k,x,t)$.

Multiplying through by Δt and taking the exponential of both sides isolates $u^{(k+1)}$ as (equation 7);

$$u^{(k+1)} = u^k \exp(\Delta t\,\Phi^k(u^k)) \qquad (7)$$

where the term inside the exponential is denoted as $\Phi^k(u^k) = d\,1/u^k\,(\partial^2 u^k)/(\partial x^2) - v(u^k,x,t)\,1/u^k\,(\partial u^k)/\partial x + 1/u^k\,S(u^k,x,t)$.

To avoid numerical instabilities when $u^k$ approaches 0, regularization techniques are commonly applied. One widely used method replaces $u^k$ in the denominator with $\max(u^k,\delta)$, $u^k+\delta$, or $\sqrt{(u^k)^2+\delta^2}$, where δ>0 is a small positive constant. The last approach, particularly suitable when u is non-negative, provides smooth and differentiable behavior near $u^k = 0$. These techniques prevent division by extremely small values of $u^k$, which could otherwise result in computational errors. Additionally, regularization stabilizes nonlinear terms like $v(u,x,t)$ and $S(u,x,t)$, which might otherwise become undefined or excessively large for small $u^k$. Depending on the problem's specific requirements and the solution's properties, alternative regularization strategies may also be employed (Tarantola 2005; Tikhonov et al. 1977).

In the following sections, this method will be integrated into the PINN framework, combining the advantages of numerical techniques and machine learning to address nonlinear CRD problems.

### 2.2. Physics-Informed Neural Networks

PINNs are a class of machine learning models designed to solve PDEs by incorporating the governing physics directly into the training process. Unlike traditional numerical methods, PINNs take advantage of the flexibility of artificial neural networks to approximate solutions while satisfying PDE constraints, initial and boundary conditions, during the optimization process. A neural network, $\mathcal{NN}$, is a parameterized nonlinear mapping function that maps input variables to output variables through a systematic arrangement of layers. Mathematically, it can be expressed as (equation 8);

$$\mathcal{NN}(x,t;\theta) = \mathcal{F}^{(L)} \circ \mathcal{F}^{(L-1)} \circ \cdots \circ \mathcal{F}^{(1)}(x,t) \qquad (8)$$

where $x$ and $t$ are the input variables, $\mathcal{F}^{(l)}$ denotes the function applied by the $l$-th layer, $L$ is the total number of layers, and $\theta$ denotes the trainable parameters (weights and biases) of the network. Each hidden layer performs a combination of linear transformations and nonlinear activation functions, expressed as (equation 9);

$$\mathcal{F}_j^{(l)} = \sigma^{(l)}\left(\sum_{i=1}^{n_u^{(l-1)}} W_{ji}^{(l)} \mathcal{F}_i^{(l-1)} + b_j^{(l)}\right), \forall l \in \{1, \dots, L-1\}. \qquad (9)$$

Here, $W^l$ and $b^l$ denote the weight matrix and bias vector for the $l$-th layer, respectively, such that $\theta^{(l)} = \{W^{(l)}, b^{(l)}\}$ represents the trainable parameters of the $l$-th layer. The activation function $\sigma^{(l)}$, is applied element-wise to the output of the linear transformation at the $l$-th layer, with the final layer using a linear activation function, i.e., $\sigma^{(L)}(z) = z$. The input layer initializes the process as $\mathcal{F}^{(0)}(x,t) = (x,t)$, while the output of the final layer provides the predicted solution $u(x,t)$.

In the PINN framework, the neural network $\mathcal{NN}(x,t;\theta)$is trained to approximate the solution of a PDE defined over a spatiotemporal domain $\Omega \times [0,T]$. This predicted solution is denoted by (equation 10);

$$u(x,t) = \mathcal{NN}(x,t;\theta). \qquad (10)$$

A key feature of PINNs is their ability to enforce the governing PDE and boundary/initial conditions as either soft or hard constraints during training. This is accomplished by designing a loss function that includes the residual of the PDE, boundary conditions, initial conditions, and additional data loss when available. The total loss function is expressed as

$\mathcal{L}(\theta) = w_{PDE}\mathcal{L}_{PDE} + w_{BC}\mathcal{L}_{BC} + w_{IC}\mathcal{L}_{IC} + w_{Data}\mathcal{L}_{Data}$,

where $w_{PDE}$, $w_{BC}$, $w_{IC}$, $w_{Data}$ are weights balancing the contribution of each term. The individual components are defined as follows:

- PDE Residual Loss: Enforces the governing equation at collocation points $\mathcal{T}_f \subset \Omega \times [0,T]$

$$\mathcal{L}_{PDE} = \frac{1}{\mathcal{T}_f}\sum_{x\in\mathcal{T}_f} \parallel \frac{\partial u}{\partial t} - d\frac{\partial^2 u}{\partial x^2} + v(u,x,t)\frac{\partial u}{\partial x} - S(u,x,t) \parallel^2.$$

- Boundary Condition Loss: Imposes the boundary conditions $B[u] = b(x,t) on \partial\Omega$

$$\mathcal{L}_{BC} = \frac{1}{\mathcal{T}_b}\sum_{x\in\mathcal{T}_b} \parallel B[u] - b(x,t) \parallel^2.$$

- Initial Condition Loss: Imposes the boundary conditions at $t = 0$

$$\mathcal{L}_{IC} = \frac{1}{\mathcal{T}_0}\sum_{x\in\mathcal{T}_0} \parallel u(x,0) - u_0(x) \parallel^2.$$

- Data Loss: Ensures that the predicted solution $u$matches the given data at specific points $\mathcal{X}_{data}$

$$\mathcal{L}_{Data} = \frac{1}{N_{data}}\sum_{i=1}^{N_{data}} \parallel u(x_{data}^{(i)}, t_{data}^{(i)}) - u_{data}^{(i)} \parallel^2.$$

Here, $\mathcal{X}_{data} = \{(x_{data}^{(i)}, t_{data}^{(i)}, u_{data}^{(i)})\}_{i=1}^{N_{data}}$ denotes the training set containing information about the solution $u_{data}$ at the input locations $x_{data}^{(i)}$ and $t_{data}^{(i)}$. Additionally, $\mathcal{T}_f$, $\mathcal{T}_b$, and $\mathcal{T}_0$ refer to collocation points within the domain, on the boundary, and at the initial time, respectively. Specifically, the PINN ensures boundary conditions by comparing the model's predictions at boundary points directly to the prescribed boundary values. The differences between the predictions and these known conditions are minimized during training. As a result, the trained solution accurately respects the physical constraints defined at the boundaries . A similar approach is applied when enforcing initial conditions and data losses, where predictions are evaluated at the respective points and differences are minimized to ensure consistency with the initial conditions and available data.

The aim is to minimize the total loss $\mathcal{L}(\theta)$, allowing the network to approximate the solution $u(x,t)$ while satisfying the underlying physical constraints using Automatic Differentiation (AD). AD is crucial in PINNs, as it enables the computation of neural network derivatives with respect to inputs through backpropagation (Baydin et al. 2018). Libraries such as TensorFlow (Abadi et al. 2015) and PyTorch (Paszke et al. 2019) provide robust AD tools, simplifying the implementation and application of PINNs. The training objective is to determine the optimal parameters $\theta^*$ using gradient-based optimization methods, such as stochastic gradient descent (SGD) and adaptive algorithms like Adam (Kingma and Ba 2014). The minimization process is expressed as (equation 11);

$$\theta^* = \underset{\theta}{argmin}\mathcal{L}(\theta). \quad (11)$$

This PINN framework operates iteratively. It begins by sampling spatial and temporal points from the domain. These inputs are fed into the neural network to predict $u(x,t)$, which is then used to compute derivatives and evaluate the loss terms. The network parameters are updated iteratively until convergence criteria are met, such as achieving a target loss value or reaching the maximum number of iterations. This process enables PINNs to approximate solutions while balancing physical constraints and available data, offering a strong and flexible alternative to traditional numerical methods.

### 2.3. Integrated PINN Framework with Exponential Schemes

In this section, we propose an integrated framework that combines the strengths of the exponential scheme with the PINN framework. The motivation for this integration comes from the well-established benefits of exponential schemes in numerical methods, particularly their ability to achieve high accuracy and stable solutions for nonlinear CRD-like equations. Both explicit and implicit exponential finite difference methods have proven to be computationally efficient while delivering solutions that closely match analytical results, often outperforming standard numerical techniques in terms of both accuracy and stability (Inan and Bahadır 2013; Tian and Dai 2007).

The PINN framework traditionally minimizes the residuals of the governing PDE, boundary conditions, and initial conditions. By integrating the exponential scheme, we introduce an additional residual term derived from the exponential update formulation. This term has the potential to improve the training process by enhancing the accuracy of the approximated solution. As a result, the integrated framework combines the numerical advantages of exponential schemes with the data-driven learning capabilities of PINNs.

The final equation derived from the exponential scheme (7) requires the solution at the next time step, $u^{k+1}$, which poses a challenge when integrating this directly into the PINN framework. As $u^{k+1}$ cannot be determined without explicitly evaluating the neural network at the $k$-th time step, the exponential finite difference scheme needs to be adjusted to account for the temporal dynamics within the PINN framework. As a result, $u^{k+1}$ in the exponential scheme is replaced with its second-order Taylor expansion around $u^k$ (equation 12);

$$u^{k+1} \approx u^k + \Delta t\frac{\partial u^k}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u^k}{\partial t^2}, \quad (12)$$

where $\Delta t$ denotes the time step size. Substituting this into equation 13, we have

$$u^k + \Delta t \frac{\partial u^k}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u^k}{\partial t^2} = u^k exp(\Delta t \Phi^k) \qquad (13)$$

This equation holds true for any given time step $k$. As a result, all terms in the equation are evaluated at the same time level $k$, ensuring consistency. Therefore, the use of superscripts to denote the time step becomes unnecessary, and we omit them for simplicity, highlighting that the equation is valid for all time values (equation 13);

$$u + \Delta t \frac{\partial u}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u}{\partial t^2} = uexp\left(\Delta t \Phi(u)\right) \qquad (13)$$

This scheme introduces an additional residual term involving the second temporal derivative $\partial^2 u/\partial t^2$, alongside the existing spatial and temporal gradients. During training, the modified scheme contributes to the total loss by incorporating the constraints of the exponential formulation while ensuring consistency with the physical residuals. This integrated approach strengthens the enforcement of physical laws and enhances the accuracy and stability of the predicted solution.

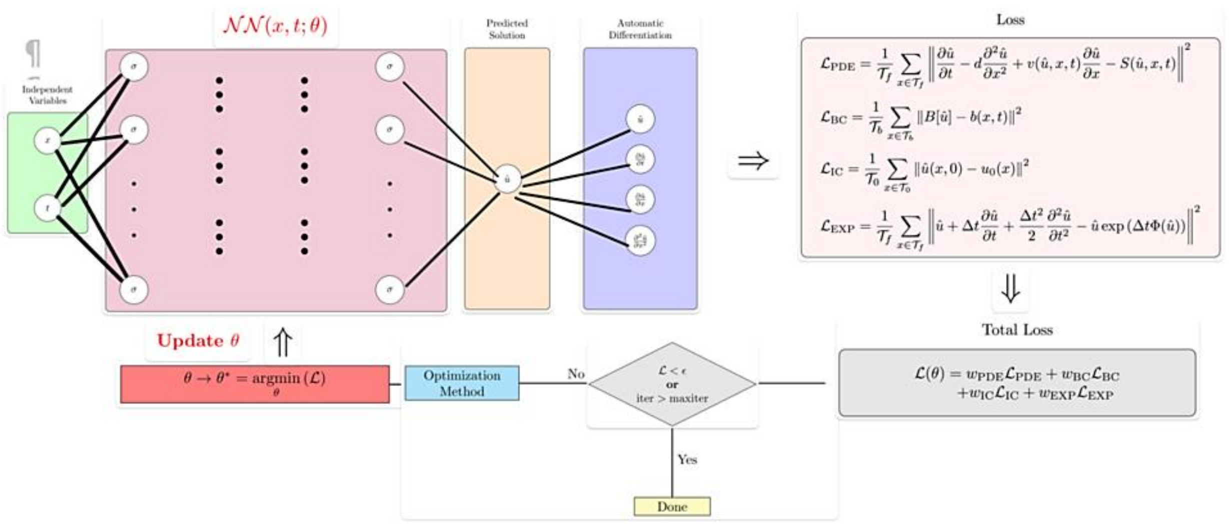
The overall procedure of the modified PINN framework is depicted in Figure 1. In addition to the standard loss terms, the modified framework includes an additional residual term based on the exponential scheme, which strengthens the numerical robustness and accuracy of the learning process. The total loss function is expressed as

$\mathcal{L}(\theta) = w_{PDE}\mathcal{L}_{PDE} + w_{BC}\mathcal{L}_{BC} + w_{IC}\mathcal{L}_{IC} + w_{EXP}\mathcal{L}_{EXP}$,

where $\mathcal{L}_{EXP}$ is the residual associated with the exponential scheme (equation 14);

$$\mathcal{L}_{EXP} = \frac{1}{\mathcal{T}_f}\sum_{x\in\mathcal{T}_f} \parallel u + \Delta t \frac{\partial u}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u}{\partial t^2} - uexp\left(\Delta t \Phi(u)\right) \parallel^2. \qquad (14)$$

By integrating the residual of the exponential scheme into the training process, the modified PINN framework effectively combines the strengths of numerical techniques with the adaptability of neural networks, leading to improved solution accuracy and stability.



$\mathcal{NN}(x,t;\theta)$
Independent Variables
Predicted Solution
Automatic Differentiation
Loss
$\mathcal{L}_{PDE} = \frac{1}{\mathcal{T}_f}\sum_{x\in\mathcal{T}_f}\left\|\frac{\partial \hat{u}}{\partial t} - d\frac{\partial^2 \hat{u}}{\partial x^2} + v(\hat{u},x,t)\frac{\partial \hat{u}}{\partial x} - S(\hat{u},x,t)\right\|^2$
$\mathcal{L}_{BC} = \frac{1}{\mathcal{T}_b}\sum_{x\in\mathcal{T}_b}\|B[\hat{u}] - b(x,t)\|^2$
$\mathcal{L}_{IC} = \frac{1}{\mathcal{T}_0}\sum_{x\in\mathcal{T}_0}\|\hat{u}(x,0) - u_0(x)\|^2$
$\mathcal{L}_{EXP} = \frac{1}{\mathcal{T}_f}\sum_{x\in\mathcal{T}_f}\left\|\hat{u} + \Delta t\frac{\partial \hat{u}}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 \hat{u}}{\partial t^2} - \hat{u}\exp(\Delta t\Phi(\hat{u}))\right\|^2$
Total Loss
$\mathcal{L}(\theta) = w_{PDE}\mathcal{L}_{PDE} + w_{BC}\mathcal{L}_{BC} + w_{IC}\mathcal{L}_{IC} + w_{EXP}\mathcal{L}_{EXP}$
Update $\theta$
$\theta \to \theta^* = \underset{\theta}{\text{argmin}}(\mathcal{L})$
Optimization Method
No
$\mathcal{L} < \epsilon$ or iter > maxiter
Yes
Done

**Figure 1.** Illustration of the integrated framework combining PINNs with the modified exponential scheme, showcasing the processing of inputs, solution prediction, and computation of loss terms for optimization.

## 3. Results

In this study, we implement our modified PINN approach on four nonlinear PDEs that are widely used in various scientific and engineering applications: the nonlinear Burgers' equation (Burgers 1948), Fisher's equation (Fisher 1937), the Burgers-Huxley equation (Burgers 1948; Hodgkin and Huxley 1952), and the Newell-Whitehead-Segel equation (Newell and Whitehead 1969). To ensure a meaningful comparison with previously obtained results presented in Hasan et al. (2024), we adopt the same neural network architecture, test models, their parameter values and their corresponding initial and boundary conditions.

For the implementation, a fully connected feedforward neural network is used, consisting of two input neurons (representing $x$ and $t$), four hidden layers with 20 neurons each, and one output neuron predicting the solution $u(x,t)$. The network employs the *tanh* activation function, the Glorot uniform initializer, and L2 regularization to reduce overfitting. Training is conducted using the Adam optimizer with a learning rate of $1 \times 10^{-3}$. The training dataset includes 10000 collocation points sampled from the spatiotemporal domain using Latin Hypercube Sampling, while boundary and initial conditions are enforced using 50uniformly sampled points each.

After the initial training phase with the Adam optimizer, we employ the LBFGS optimizer (Liu and Nocedal 1989) as a secondary step to refine the solution further. The use of LBFGS represents the only methodological difference in the optimization process compared to the prior study.

Even though it is possible to assign different weights to each loss component, in this study, we apply equal weights $w = 1$ to all loss terms in the loss function. This approach simplifies the training process by avoiding the need to tune multiple weight parameters. It also ensures that no single loss term dominates the training, resulting in balanced contributions from each part. However, this assumes each loss component has roughly equal importance, which might not hold true if some terms inherently have larger magnitudes or significantly greater influence on the solution accuracy. In such cases, adjusting weights might further improve performance.

The model is trained by minimizing the total loss function, which ensures that the predicted solution $u(x,t)$ aligns with the governing PDE, initial and boundary conditions, and the residuals of the modified exponential scheme.

In PINNs, accurately satisfying initial and boundary conditions is critical to ensure the physical validity of the computed solutions. A recent study by Lu et al. (2021) highlighted the advantages of modifying the network architecture to incorporate hard constraints. This approach allows the network to strictly enforce initial and boundary conditions. This also improves the stability and convergence of the training process, and reduces the computational cost.

Building on this concept, we develop an output transformation framework tailored to our implementation. During training, the model approximates the solution $u(x,t)$ across the spatial-temporal domain. To ensure adherence to both initial and boundary conditions, an output transform, denoted as $\tilde{u}(x,t)$, is applied. This transform effectively enforces problem-specific constraints on the predicted solution $u(x,t)$, ensuring that the initial condition at $t = 0$ and the specified boundary conditions are satisfied. Mathematically, the transform $\tilde{u}(x,t)$ can be described as

$$\tilde{u}(x,t) = u_c(x,t) + t g(x) u(x,t), \tag{14}$$

where $u_c(x,t)$represents the constrained term that satisfies the initial and boundary conditions. Specifically, $u_c(x,t)$ satisfies the initial condition at $t = 0$, such that $u_c(x,0) = u_0(x)$ for $x \in \Omega$, and supports the boundary conditions at the boundary points, ensuring $B[u_c] = b(x,t), (x,t) \in \partial\Omega \times \mathbb{R}_+$.

The term $g(x)$ is the boundary basis function, designed to vanish at the domain boundaries. By incorporating this output transform, the predicted solution $u(x,t)$is naturally adjusted to satisfy both the initial and boundary conditions, ensuring physically consistent and accurate predictions throughout the domain. In this study, hard constraints are incorporated directly into the PINN by transforming the neural network's output, ensuring these constraints are exactly satisfied by construction, rather than being enforced through additional penalty terms in the loss function.

Through the optimization process, the model learns to approximate the solution $u(x,t)$ accurately across the spatial-temporal domain. The model's performance was evaluated using mean absolute error, $MAE$, which quantifies the average difference between the true solution $u(x_i,t_i)$ and the predicted solution $u(x_i,t_i)$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|u(x_i,t_i) - u(x_i,t_i)\right|,$$

where $N$ represents the total number of evaluation points.

### 3.1. Numerical Experiments and Comparative Analysis

In this section, we aim to evaluate the effectiveness of the proposed enhanced PINN framework by applying it to four nonlinear CRD systems. These systems include Burgers Equation, Fisher′s Equation, the Burgers-Huxley Equation, and the Newell-Whitehead-Segel Equation, which represent a broad spectrum of nonlinear dynamics. The numerical experiments aim to demonstrate the framework's accuracy in solving these equations, while comparisons will emphasize the improvements achieved over the GFEM and traditional PINN approaches.

#### 3.1.1. Test problem: Burgers' equation

Burgers' equation, first introduced by J. M. Burgers in 1948 (Burgers 1948), is a fundamental nonlinear partial differential equation that has become a cornerstone in mathematical physics and applied mathematics. Celebrated for its simplicity and versatility, the equation captures the interaction between nonlinear convection and viscous diffusion, making it an essential tool for analyzing complex dynamical systems. While Burgers' equation is analytically solvable in certain cases, its inherent nonlinearity often requires advanced computational methods to address more general scenarios. The specific form of Burgers' equation studied here is expressed as

$$\frac{\partial u(x,t)}{\partial t} = \frac{1}{Re}\frac{\partial^2 u(x,t)}{\partial x^2} - u(x,t)\frac{\partial u(x,t)}{\partial x},$$

where $Re > 0$ denotes the Reynolds number, a dimensionless parameter that quantifies the balance between viscous and inertial forces. The problem is accompanied by the following initial and boundary conditions.

$$u(x,0) = \frac{2\pi sin(\pi x)}{Re(\sigma + cos(\pi x))}, x \in \Omega = [0,1]$$

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

where $\sigma > 1$ is a given parameter. In this study, we adopt the parameter values $Re = 1$and $\sigma = 2$, consistent with those used in Hasan et al. (2024). An analytical solution to this problem, corresponding to the specified initial and boundary conditions and the parameter values, is available and is expressed as

$$u(x,t) = \frac{2\pi e^{-\pi^2 t/Re} sin(\pi x)}{Re\left(\sigma + e^{-\pi^2 t/Re} cos(\pi x)\right)},$$

which provides a benchmark for assessing the accuracy of numerical solutions.

In Figure 2, we present a comparison of the analytical solution and the predicted solution for Burgers' Equation using the modified PINN framework. The figure includes the analytical and predicted solutions as 2D plots, the MAE between them, and a 3D visualization of the predicted solution. As depicted, it is safe to say that there is a good agreement between analytical expression and modified PINN based prediction.

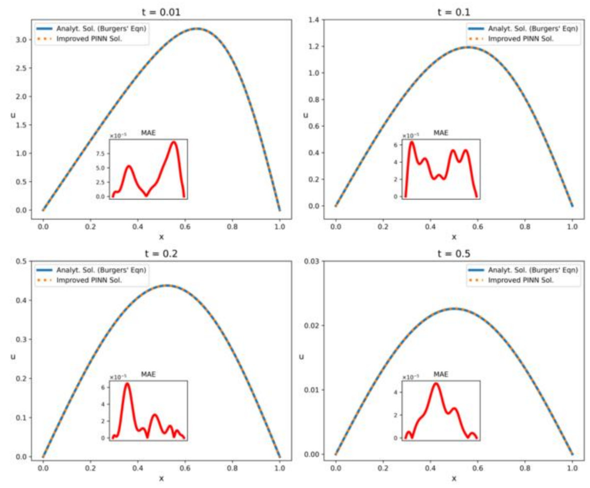


**Figure 2.** Analytical solution (top-left), improved PINN solution (top-right), mean absolute error (bottom-left), and 3D view of the modified PINN solution (bottom-right) for Burgers' equation.

In Figure 3, we present a detailed comparison of the analytical and predicted solutions for Burgers' Equation at four distinct time values: $t = 0.01$, $t = 0.1$, $t = 0.2$and $t = 0.5$. Each subplot depicts the spatial evolution of the solutions, enabling a direct visual comparison between the analytical and modified PINN predictions. Insets within each subplot show the $MAE$ for the corresponding time value, emphasizing the discrepancies between the solutions. The first immediate observation from this figure is that our model gives excellent agreement with the analytical solutions, with $MAE$ values remaining consistently low across all time values.

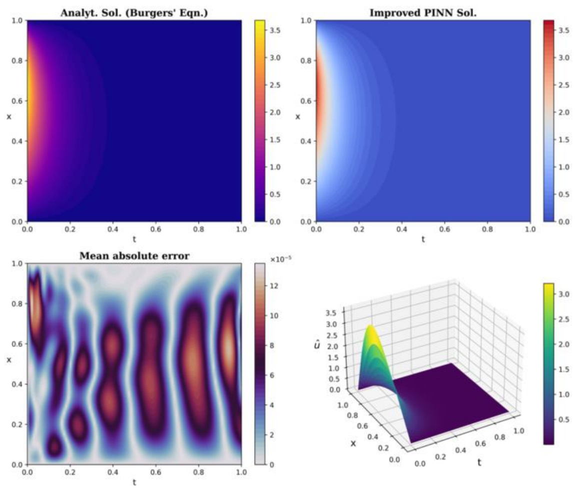


**Figure 3.** Comparison of analytical and predicted solutions for Burgers' Equation at various time values, along with the mean absolute error.

Table 1 presents a quantitative comparison of the mean absolute errors ($MAE$) between the analytical solution and three considered methods—GFEM, the traditional PINN framework, and the proposed modified PINN framework—at two specific time steps, $t = 0.02$ and $t = 0.04$. The modified PINN consistently outperforms GFEM and the traditional PINN, achieving significantly lower errors across all spatial points for both time values.

**Table 1.** Absolute error comparison for Burgers' equation at $t = 0.02$ and $t = 0.04$ between the analytical solution and predictions from GFEM, traditional PINN, and modified PINN

| | $t = 0.02$ | | | $t = 0.04$ | | |
|---|---|---|---|---|---|---|
| $x$ | GFEM | PINN | PRESENT | GFEM | PINN | PRESENT |
| 0.1 | $4.20 \times 10^{-3}$ | $5.09 \times 10^{-4}$ | $1.99 \times 10^{-5}$ | $1.21 \times 10^{-2}$ | $3.87 \times 10^{-4}$ | $1.57 \times 10^{-5}$ |
| 0.2 | $9.10 \times 10^{-3}$ | $3.37 \times 10^{-4}$ | $2.23 \times 10^{-5}$ | $2.50 \times 10^{-2}$ | $8.00 \times 10^{-5}$ | $1.78 \times 10^{-6}$ |
| 0.4 | $2.16 \times 10^{-2}$ | $2.42 \times 10^{-4}$ | $2.55 \times 10^{-5}$ | $5.07 \times 10^{-2}$ | $2.92 \times 10^{-4}$ | $2.87 \times 10^{-5}$ |
| 0.6 | $2.51 \times 10^{-2}$ | $8.70 \times 10^{-5}$ | $6.15 \times 10^{-5}$ | $4.83 \times 10^{-2}$ | $2.63 \times 10^{-4}$ | $8.30 \times 10^{-5}$ |
| 0.8 | $6.83 \times 10^{-3}$ | $3.90 \times 10^{-5}$ | $1.24 \times 10^{-4}$ | $9.19 \times 10^{-3}$ | $1.62 \times 10^{-4}$ | $1.30 \times 10^{-4}$ |
| 1.0 | $0.00 \times 10^{0}$ | $1.17 \times 10^{-3}$ | $2.3 \times 10^{-16}$ | $0.00 \times 10^{0}$ | $3.90 \times 10^{-5}$ | $3.8 \times 10^{-16}$ |

### 3.1.2. Test problem: Fisher's equation

Fisher's equation, introduced by R. A. Fisher in 1937 (Fisher 1937), models the spread of advantageous genetic traits within a population. A fundamental equation in mathematical biology, it captures the interplay between diffusion and logistic growth, describing phenomena such as traveling wavefronts. Although analytical solutions are available for specific cases, the nonlinearity of the equation necessitates advanced numerical methods to handle complex scenarios with varied initial and boundary conditions. Mathematically, Fisher's equation is expressed as

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2} + 6u(x,t)(1 - u(x,t)),$$

where $u(x,t)$represents the population density or another relevant quantity. The initial condition considered in this work is specified as

$u(x,0) = (1 + e^{x})^{-2}, x \in [0,1]$

and the boundary conditions are

$u(0,t) = (1+e^{-5t})^{-2}, u(1,t) = (1+e^{1-5t})^{-2}$.

The exact analytical solution to Fisher's equation, corresponding to the specified initial and boundary conditions, is expressed as

$u(x,t) = (1+e^{x-5t})^{-2}$,

which provides a benchmark for verifying the accuracy of numerical methods. This expression is used to compare with the predictions of our modified PINN framework to assess its performance. In Figure 4, we present a detailed visualization of the solutions for Fisher's Equation. As illustrated, the predictions of the modified PINN model are in excellent agreement with the analytical solution, demonstrating its capability to accurately capture the spatiotemporal dynamics of Fisher's equation.

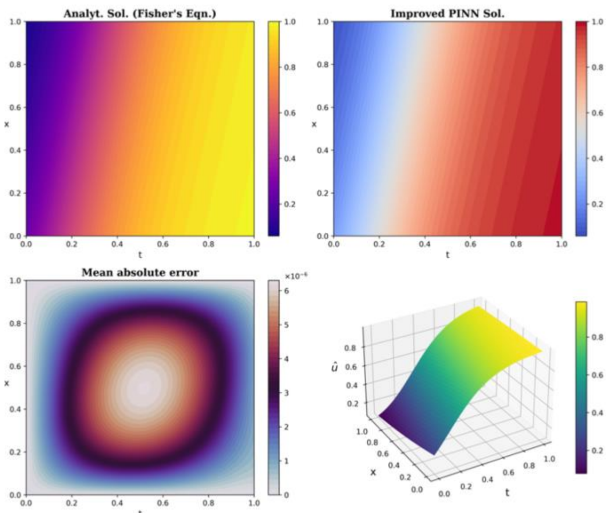


**Figure 4.** Analytical solution (top-left), improved PINN solution (top-right), mean absolute error (bottom-left), and 3D view of the modified PINN solution (bottom-right) for Fisher's equation.

In Figure 5, we present a comparison of the analytical and predicted solutions for Fisher's Equation at four distinct time steps: $t = 0.25$, $t = 0.5$, $t = 0.75$, and $t = 1$. Each subplot depicts the spatial evolution of the solutions, allowing for a direct visual comparison between the analytical and improved PINN results. Across all time steps, the $MAE$ demonstrates a consistent trend, with the smallest errors occurring at the boundaries. This behavior aligns with other test cases we consider, where the use of hard constraints plays a pivotal role.

Table 2 compares the absolute errors between the analytical solution of Fisher's equation and results obtained using GFEM, the traditional PINN, and the modified PINN at $t = 0.05$ and $t = 0.1$. The results clearly demonstrate that the modified PINN consistently achieves the lowest errors across all spatial points for considered time values.

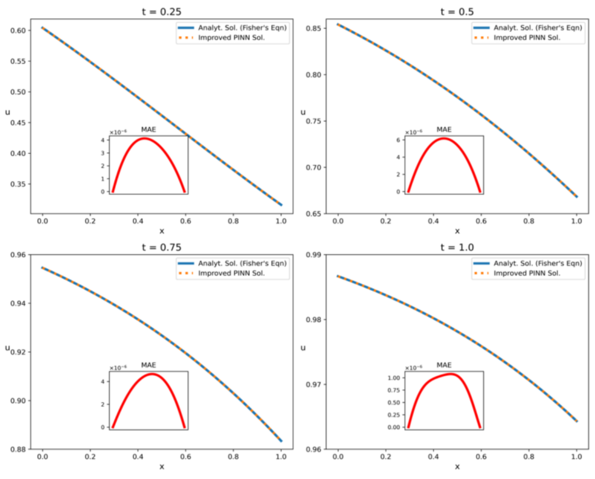


**Figure 5.** Comparison of analytical and predicted solutions for Fisher's Equation at $t = 0.25$, $t = 0.5$, $t = 0.75$, and $t = 1$.

**Table 2.** Absolute error comparison for Fisher's equation at $t = 0.05$ and $t = 0.1$ between the analytical solution and predictions from GFEM, traditional PINN, and modified PINN

| | $t = 0.05$ | | | $t = 0.1$ | | |
|---|---|---|---|---|---|---|
| $x$ | GFEM | PINN | PRESENT | GFEM | PINN | PRESENT |
| 0.1 | $3.02 \times 10^{-3}$ | $3.67 \times 10^{-4}$ | $4.28 \times 10^{-7}$ | $1.33 \times 10^{-3}$ | $4.19 \times 10^{-4}$ | $8.36 \times 10^{-7}$ |
| 0.2 | $7.91 \times 10^{-4}$ | $2.27 \times 10^{-4}$ | $6.88 \times 10^{-7}$ | $2.78 \times 10^{-3}$ | $3.35 \times 10^{-4}$ | $1.36 \times 10^{-6}$ |
| 0.4 | $1.35 \times 10^{-3}$ | $3.40 \times 10^{-5}$ | $8.40 \times 10^{-7}$ | $6.27 \times 10^{-3}$ | $8.70 \times 10^{-5}$ | $1.70 \times 10^{-6}$ |
| 0.6 | $1.51 \times 10^{-3}$ | $1.13 \times 10^{-4}$ | $7.08 \times 10^{-7}$ | $5.79 \times 10^{-3}$ | $2.90 \times 10^{-5}$ | $1.48 \times 10^{-6}$ |
| 0.8 | $3.87 \times 10^{-4}$ | $1.48 \times 10^{-4}$ | $4.35 \times 10^{-7}$ | $2.71 \times 10^{-3}$ | $1.76 \times 10^{-4}$ | $9.25 \times 10^{-7}$ |
| 1.0 | $9.67 \times 10^{-18}$ | $1.32 \times 10^{-4}$ | $0.00 \times 10^{+0}$ | $6.32 \times 10^{-18}$ | $2.34 \times 10^{-4}$ | $0.00 \times 10^{+0}$ |

### 3.1.3. Test problem: Burgers-Huxley equation

The Burgers-Huxley equation combines the dynamics of Burgers' and Huxley's equations to model reaction-diffusion processes in excitable media. t plays a critical role in understanding the interplay of convection, diffusion, and reaction phenomena, with applications in different fields such as neuroscience, population genetics, and chemical kinetics. The equation is particularly significant for its capacity to capture traveling wave solutions and complex spatiotemporal behaviors, making it a cornerstone in the study of nonlinear systems and numerical methods (Keener and Sneyd 2009; Whitham 2011). Mathematically, the equation is given by

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - \phi\frac{\partial u}{\partial x} + \epsilon u(x,t)(u(x,t)-1)(\tau - u(x,t)),$$

where $\phi$, $\epsilon$, and $\tau$ are real valued parameters. The initial condition for this equation is defined as

$$u(x,0) = \frac{3}{2} + \frac{1}{2}tanh\left(\frac{x}{2}\right), x \in \Omega = [-25,17]$$

and the boundary conditions are given as

$$\frac{\partial u}{\partial x}|_{x=-25} = \frac{\partial u}{\partial x}|_{x=17} = 0, t > 0,$$

indicating a no-flux condition at the spatial boundaries. The analytical solution for this configuration for the parameter values $\phi = -1$, $\epsilon = 1$, and $\tau = 2$, is expressed as

$$u(x,t) = \frac{3}{2} + \frac{1}{2} tanh\left(\frac{1}{2}(x + 3t)\right).$$

This analytical result serves as a reference to evaluate the accuracy and effectiveness of the our modified PINN method in capturing the equation's dynamics.

Figure 6 presents contour plots of the Burgers-Huxley equation, illustrating the spatiotemporal evolution of the predicted solution alongside the analytical solution. Additionally, the figure includes a visualization of the $MAE$ and a 3D plot of the modified PINN solution. The results reveal that the error is notably minimized near the boundaries, as expected.

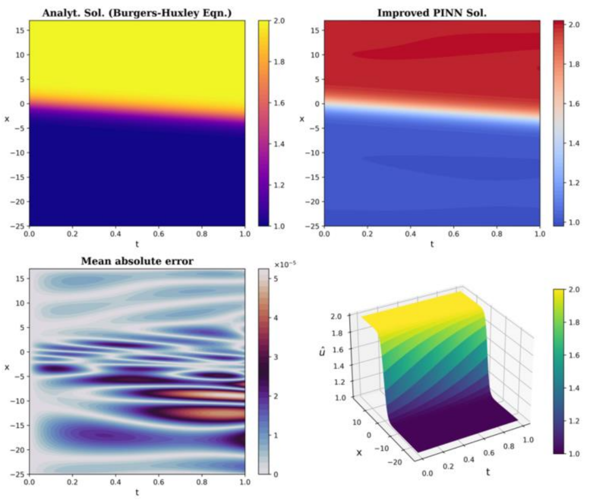


**Figure 6.** Contour plots of the analytical and predicted solutions, the MAE, and a 3D plot of the predicted solution for the Burgers-Huxley equation.

In Figure 7, the analytical and predicted solutions for the Burgers-Huxley equation are presented for four time values: $t = 0.05$, $t = 0.5$, $t = 0.75$, and $t = 1$. The results show that the modified PINN framework closely matches the analytical solution, effectively capturing the dynamics of the Burgers-Huxley equation across the domain and over time.

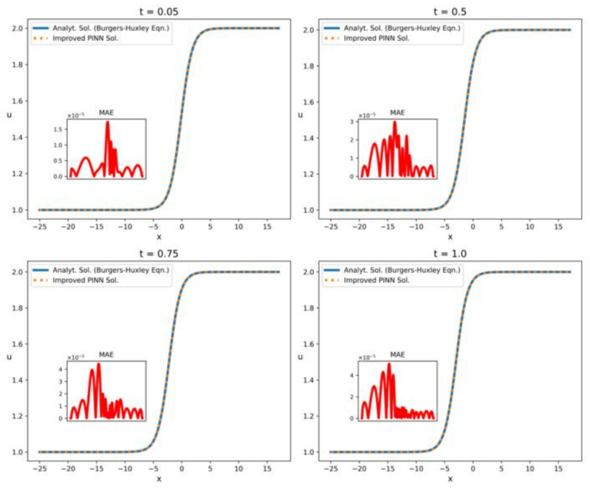


**Figure 7 .** Evolution of the analytical and predicted solutions for the Burgers-Huxley equation at selected time values $t = 0.05$, $t = 0.5$, $t = 0.75$, and $t = 1$, along with insets displaying the corresponding mean absolute errors (MAE) for each time values.

Table 3 presents the absolute error between the analytical solution and three methods—GFEM, Traditional PINN, and Modified PINN—for the Burgers-Huxley equation at $t = 0.10$ and $t = 0.20$. The Modified PINN consistently achieves the lowest error values across most spatial locations and time instances, outperforming both GFEM and Traditional PINN. The exception occurs near the spatial boundaries, where GFEM shows slightly better performance. However, at these boundary locations, the analytical solution is known to be fixed, making the observed differences in errors marginal in practical significance.

**Table 3.** Absolute error comparison for Burgers-Huxley equation at $t = 0.10$ and $t = 0.20$ between the analytical solution and predictions from GFEM, traditional PINN, and modified PINN

| | $t = 0.1$ | | | $t = 0.2$ | | |
|---|---|---|---|---|---|---|
| $x$ | GFEM | PINN | PRESENT | GFEM | PINN | PRESENT |
| -25.00 | $4.73 \times 10^{-13}$ | $4.27 \times 10^{-3}$ | $4.86 \times 10^{-12}$ | $3.84 \times 10^{-12}$ | $5.38 \times 10^{-3}$ | $1.14 \times 10^{-11}$ |
| -20.80 | $1.86 \times 10^{-10}$ | $7.27 \times 10^{-4}$ | $2.40 \times 10^{-5}$ | $4.09 \times 10^{-11}$ | $1.00 \times 10^{-3}$ | $4.24 \times 10^{-5}$ |
| -12.40 | $8.26 \times 10^{-7}$ | $2.53 \times 10^{-3}$ | $2.86 \times 10^{-5}$ | $1.82 \times 10^{-7}$ | $3.47 \times 10^{-3}$ | $1.61 \times 10^{-5}$ |
| -4.00 | $3.49 \times 10^{-3}$ | $5.37 \times 10^{-3}$ | $1.92 \times 10^{-4}$ | $7.18 \times 10^{-4}$ | $6.09 \times 10^{-3}$ | $1.54 \times 10^{-4}$ |
| 0.20 | $3.50 \times 10^{-3}$ | $6.89 \times 10^{-4}$ | $3.22 \times 10^{-5}$ | $2.67 \times 10^{-3}$ | $1.81 \times 10^{-3}$ | $7.05 \times 10^{-5}$ |
| 4.40 | $1.22 \times 10^{-3}$ | $5.41 \times 10^{-3}$ | $1.88 \times 10^{-5}$ | $7.54 \times 10^{-4}$ | $4.19 \times 10^{-3}$ | $3.60 \times 10^{-5}$ |
| 12.80 | $2.79 \times 10^{-7}$ | $3.04 \times 10^{-3}$ | $4.47 \times 10^{-5}$ | $1.78 \times 10^{-7}$ | $2.08 \times 10^{-3}$ | $5.52 \times 10^{-5}$ |
| 17.00 | $8.48 \times 10^{-9}$ | $6.30 \times 10^{-3}$ | $1.07 \times 10^{-8}$ | $4.11 \times 10^{-9}$ | $5.22 \times 10^{-3}$ | $1.87 \times 10^{-8}$ |

### 3.1.5. Test problem: Newell-Whitehead-Segel equation

The Newell-Whitehead-Segel equation models spatially periodic phenomena near instability points, making it essential for understanding pattern formation and bifurcation in symmetric systems. It is widely applied in fluid dynamics, reaction-diffusion systems, and population dynamics, capturing behaviors like wave propagation and emergent patterns (Cross and Hohenberg 1993; Murray 2007). The equation is mathematically expressed as

$$\frac{\partial u}{\partial t} = \eta \frac{\partial^2 u}{\partial x^2} + \mu u(x,t) - \nu u(x,t)^{\rho},$$

where $\eta, \mu, \nu > 0$ are real valued parameters and $\rho$ is a positive integer. The initial condition is given by

$$u(x,0) = \left(1 + e^{x/\sqrt{6}}\right)^{-2}, x \in \Omega = [0,1].$$

The boundary conditions are defined as non-homogeneous Neumann conditions

$$\frac{\partial u}{\partial x}|_{x=0} = \frac{2e^{-5t/6}}{\sqrt{6}(1+e^{-5t/6})^3}, \frac{\partial u}{\partial x}|_{x=1} = -\frac{2e^{(5t)/6}}{\sqrt{6}(1+e^{(5t)/6})^3}.$$

For this equation, the analytical solution is known and serves as a benchmark to evaluate the numerical methods. For the parameter values $\eta = 1$, $\mu = 1$, $\nu = 1$, and $\rho = 2$, the analytical solution is

$$u(x,t) = \left(1 + e^{\frac{x}{\sqrt{6}} - \frac{5t}{6}}\right)^{-2}.$$

We now apply our modified PINN framework to the Newell-Whitehead-Segel equation to evaluate its effectiveness in capturing the spatiotemporal dynamics of the solution. A detailed visualization of the solution's behaviour over time and space is presented in Figure 8. The results demonstrate that the modified PINN framework aligns closely with the analytical solution.

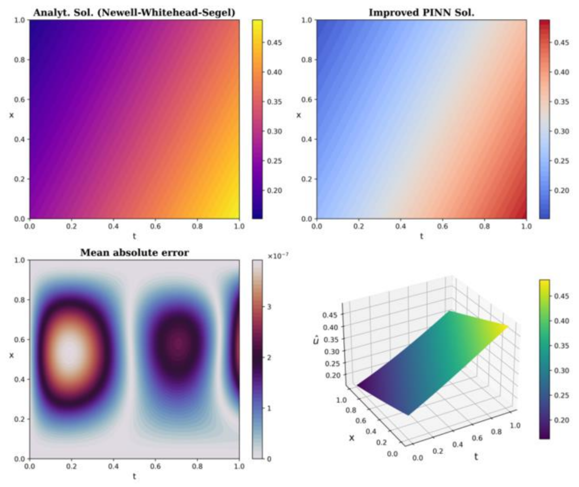


**Figure 8.** Visualizations for the Newell-Whitehead-Segel equation showing the analytical and predicted solutions as contour plots, a contour plot of the mean absolute error (MAE) across the domain, and a 3D view of the predicted solution's spatiotemporal evolution.

Figure 9 depicts the evolution of the solution to the Newell-Whitehead-Segel equation at specific time values, $t = 0.05, 0.1, 0.2,$ *and* $0.5$. The modified PINN framework produces results that closely match the analytical solution.

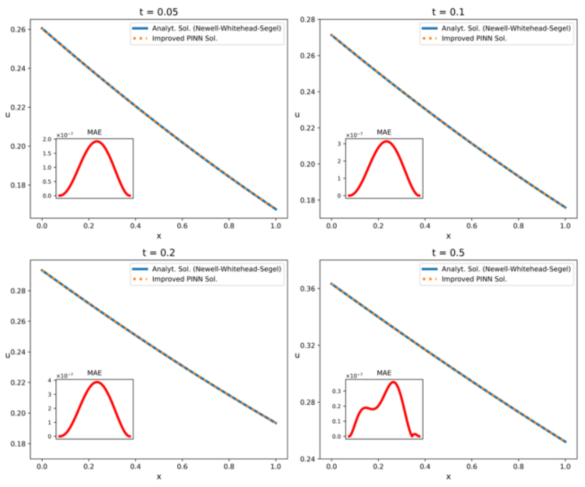


**Figure 9 .** Comparison of analytical and predicted solutions for Newell-Whitehead-Segel Equation at various time values, with mean absolute error.

Table 4 shows the absolute error between the analytical solution and three methods—GFEM, traditional PINN, and the modified PINN—for the Newell-Whitehead-Segel equation at two time instances, $t = 0.03$ and $t = 0.05$, across various spatial points. For all spatial points considered, the modified PINN gives the smallest absolute errors.

**Table 4.** Absolute error comparison for Newell-Whitehead-Segel equation at $t = 0.03$ and $t = 0.05$between the analytical solution and predictions from GFEM, traditional PINN, and modified PINN

| | $t = 0.03$ | | | $t = 0.05$ | | |
|---|---|---|---|---|---|---|
| $x$ | GFEM | PINN | PRESENT | GFEM | PINN | PRESENT |
| 0.1 | $1.48 \times 10^{-6}$ | $9.10 \times 10^{-6}$ | $1.01 \times 10^{-8}$ | $3.06 \times 10^{-5}$ | $2.55 \times 10^{-5}$ | $1.54 \times 10^{-8}$ |
| 0.2 | $1.88 \times 10^{-5}$ | $1.77 \times 10^{-5}$ | $3.78 \times 10^{-8}$ | $5.45 \times 10^{-5}$ | $3.18 \times 10^{-5}$ | $5.80 \times 10^{-8}$ |
| 0.4 | $3.57 \times 10^{-5}$ | $3.15 \times 10^{-5}$ | $1.06 \times 10^{-7}$ | $7.97 \times 10^{-5}$ | $3.78 \times 10^{-5}$ | $1.64 \times 10^{-7}$ |
| 0.6 | $4.34 \times 10^{-5}$ | $3.24 \times 10^{-5}$ | $1.19 \times 10^{-7}$ | $9.55 \times 10^{-5}$ | $3.50 \times 10^{-5}$ | $1.84 \times 10^{-7}$ |
| 0.8 | $6.45 \times 10^{-5}$ | $2.47 \times 10^{-5}$ | $5.56 \times 10^{-8}$ | $1.28 \times 10^{-4}$ | $2.31 \times 10^{-5}$ | $8.60 \times 10^{-8}$ |
| 1.0 | $1.32 \times 10^{-4}$ | $1.03 \times 10^{-5}$ | $0.00 \times 10^{+0}$ | $2.03 \times 10^{-4}$ | $1.38 \times 10^{-5}$ | $0.00 \times 10^{+0}$ |

## 5. Discussion

This study presented an improved PINN framework, incorporating an additional residual term based on an exponential finite difference scheme. The approach was developed to overcome challenges faced by traditional PINNs in solving nonlinear CRD equations, particularly those with complex dynamics, steep gradients, and strong nonlinearities. Numerical tests were conducted on four nonlinear PDEs: Burgers Equation, Fisher′s Equation, Burgers-Huxley Equation, and the Newell-Whitehead-Segel Equation, with results compared against both traditional PINNs and the GFEM.

The modified PINN framework demonstrated significant improvements in both accuracy and robustness, highlighting its potential as a versatile tool for solving a wide range of nonlinear PDEs. By incorporating residuals from the exponential scheme, the framework effectively captured the intricate dynamics of nonlinear PDEs. It consistently reduced absolute errors compared to traditional PINNs and, in most cases, outperformed GFEM in accuracy.

Results from Tables 1–4 demonstrate that the proposed modifications consistently improved model accuracy and robustness. In particular, the enhanced PINN achieved lower absolute errors across all test cases compared to traditional PINNs and generally outperformed the GFEM as well. These results highlight the effectiveness of incorporating the enhanced residual term, which more rigorously enforces the governing equations during training, leading to more accurate and physically consistent solutions.

Furthermore, while GFEM remains a reliable and well-established numerical method, it requires significant computational resources, particularly for problems in higher dimensions or with irregular geometries. In contrast, the mesh-free architecture of the PINN framework offers a scalable and flexible alternative, well-suited to modern scientific and engineering applications.

The results of this study highlight several promising directions for future research. Extending the proposed framework to high-dimensional problems and more complex geometries would provide further insights into its scalability and applicability. Moreover, integrating other advanced numerical schemes could enhance its robustness and adaptability to even more challenging PDE scenarios, such as those involving discontinuities or stochastic effects. Additionally, optimizing the training process through advanced loss-weighting strategies or adaptive optimization techniques could further improve its performance and efficiency.

**Author Contributions**

The percentages of the author' contributions are presented below. The author reviewed and approved the final version of the manuscript.

| | B.B. |
|---|---|
| C | 100 |
| D | 100 |
| S | 100 |
| DCP | 100 |
| DAI | 100 |
| L | 100 |
| W | 100 |
| CR | 100 |
| SR | 100 |
| PM | 100 |
| FA | 100 |

C=Concept, D=design, S=supervision, DCP=data collection and/or processing, DAI=data analysis and/or interpretation, L=literature search, W=writing, CR=critical review, SR=submission and revision, PM=project management, FA= funding acquisition.

**Conflict of Interest**

The author declared that there is no conflict of interest.

**Ethical Consideration**

Ethics committee approval was not required for this study because of there was no study on animals or humans.

## References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Xiao Z, Monga R, Moore S, Murray D, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Mountain View, CA: TensorFlow.

Ali H, Kamrujjaman M, Islam MS. 2022. An advanced Galerkin approach to solve the nonlinear reaction–diffusion equations with different boundary conditions. J Math Res, 14(1): pp: 30–45.

Bahadır AR. 2005. Exponential finite-difference method applied to Korteweg–de Vries equation for small times. Appl Math Comput, 160(3): pp: 675–682.

Barth T, Jespersen D. 1989. The design and application of upwind schemes on unstructured meshes. 27th Aerospace Sci Meet, Paper No: 89-0366.

Baydin AG, Pearlmutter BA, Radul AA, Siskind JM. 2018. Automatic differentiation in machine learning: A survey. J Mach Learn Res, 18(153): pp: 1–43.

Bejan A. 2013. Convection heat transfer. John Wiley & Sons, Hoboken, NJ, USA, pp: 688.

Bezekci B. 2025. Deep learning-enhanced regularization of irregular traveling pulses in the FitzHugh–Nagumo model. SN Comput Sci, 6: pp: 206.

Bezekci B. 2025. The efficacy of Haar wavelets in addressing discontinuities of McKean equations with Heaviside functions. Osmaniye Korkut Ata Univ J Inst Sci, 8(1): pp: 200–210.

Bhattacharya MC. 1985. An explicit conditionally stable finite difference equation for heat conduction problems. Int J Numer Methods Eng, 21(2): pp: 239–265.

Burgers JM. 1948. A mathematical model illustrating the theory of turbulence. Adv Appl Mech, 1: pp: 171–199.

Costa R, Clain S, Loubère R, Machado GJ. 2018. Very high-order accurate finite volume scheme on curved boundaries for the two-dimensional steady-state convection–diffusion equation with Dirichlet condition. Appl Math Model, 54: pp: 752–767.

Cross MC, Hohenberg PC. 1993. Pattern formation outside of equilibrium. Rev Mod Phys, 65(3): pp: 851.

Dong X, Li W, Liu Q, Wang H. 2022. Research on convection-reaction-diffusion model of contaminants in fracturing flowback fluid in non-equidistant fractures with arbitrary inclination of shale gas development. J Pet Sci Eng, 208: 109479.

Fisher RA. 1937. The wave of advance of advantageous genes. Ann Eugen, 7(4): pp: 355–369.

Handschuh RF, Keith TG Jr. 1992. Applications of an exponential finite-difference technique. Numer Heat Transfer A, 22(3): pp: 363–378.

Harten A. 1997. High resolution schemes for hyperbolic conservation laws. J Comput Phys, 135(2): pp: 260–278.

Hasan F, Ali H, Arief HA. 2024. From mesh to neural nets: A multi-method evaluation of physics-informed neural networks and Galerkin finite element method for solving nonlinear convection–reaction–diffusion equations. arXiv preprint arXiv:2411.09704.

Hennigh O, Narasimhan S, Nabian MA, Subramaniam A, Tangsali K, Fang Z, Rietmann M, Byeon W, Choudhry S, Ferguson M, Pfaff T, Tompson J. 2021. NVIDIA SimNet™: An AI-accelerated multi-physics simulation framework. In: Int Conf Comput Sci, Springer, pp: 447–461.

Hodgkin AL, Huxley AF. 1952. A quantitative description of membrane current and its application to conduction and excitation in nerve. J Physiol, 117(4): pp: 500.

Inan B, Bahadir AR. 2013. Numerical solution of the one-dimensional Burgers' equation: Implicit and fully implicit exponential finite difference methods. Pramana, 81: pp: 547–556.

Jagtap AD, Mao Z, Adams N, Karniadakis GE. 2022. Physics-informed neural networks for inverse problems in supersonic flows. J Comput Phys, 466: 111402.

John V, Schmeyer E. 2008. Finite element methods for time-dependent convection–diffusion–reaction equations with small diffusion. Comput Methods Appl Mech Eng, 198(3–4): pp: 475–494.

Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. 2021. Physics-informed machine learning. Nat Rev Phys, 3: pp: 422–440.

Keener J, Sneyd J. 2009. Mathematical physiology: II: Systems physiology. Springer, New York, NY, USA, pp: 574.

Kerner BS. 1999. The physics of traffic. Phys World, 12(8): pp: 25.

Kingma DP, Ba J. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Koren B. 1993. A robust upwind discretization method for advection, diffusion and source terms (Vol. 45). Centrum voor Wiskunde en Informatica, Amsterdam, Netherlands, pp: 56.

Liu DC, Nocedal J. 1989. On the limited memory BFGS method for large scale optimization. Math Program, 45(1): pp: 503–528.

Lu L, Meng X, Mao Z, Karniadakis GE. 2021. DeepXDE: A deep learning library for solving differential equations. SIAM Rev, 63(1): pp: 208–228.

Morton KW. 1996. Numerical solution of convection-diffusion problems. Chapman & Hall, London, UK, pp: 232.

Murray JD. 2007. Mathematical biology: I. An introduction (Vol. 17). Springer, New York, NY, USA, pp: 551.

Nekhamkina OA, Nepomnyashchy AA, Rubinstein BY, Sheintuch M. 2000. Nonlinear analysis of stationary patterns in convection-reaction-diffusion systems. Phys Rev E, 61(3): pp: 2436.

Newell AC, Whitehead JA. 1969. Finite bandwidth, finite amplitude convection. J Fluid Mech, 38(2): pp: 279–303.

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A. 2019. PyTorch: An imperative style, high-performance deep learning library. In: Adv Neural Inf Process Syst, pp:32.

Peddavarapu S, Srinivasan R. 2021. Local maximum entropy approximation-based streamline upwind Petrov–Galerkin meshfree method for convection–diffusion problem. J Braz Soc Mech Sci Eng, 43(6): 326.

Phongthanapanich S, Dechaumphai P. 2008. A characteristic-based finite volume element method for convection-diffusion-reaction equation. Trans Can Soc Mech Eng, 32(3–4): pp: 549–560.

Raissi M, Perdikaris P, Karniadakis GE. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys, 378: pp: 686–707.

Sahli Costabal F, Yang Y, Perdikaris P, Hurtado DE, Kuhl E. 2020. Physics-informed neural networks for cardiac activation mapping. Front Phys, 8: 42.

Seinfeld JH, Pandis SN. 1998. From air pollution to climate change. Atmospheric chemistry and physics. John Wiley & Sons, New York, NY, USA, pp: 1326.

Tarantola A. 2005. Inverse problem theory and methods for model parameter estimation. SIAM, Philadelphia, PA, USA, pp: 342.

Tian ZF, Dai SQ. 2007. High-order compact exponential finite difference methods for convection–diffusion type problems. J Comput Phys, 220(2): pp: 952–974.

.