MATHEMATICAL MODELING AND COMPUTING, Vol. 11, No. 2, pp. 505–511 (2024)

Embedding physical laws into Deep Neural Networks for solving generalized Burgers–Huxley equation

Hariri I.[1], Radid A.[1], Rhofir K.[2]

[1]*LMFA, FSAC, Hassan II University of Casablanca*
[2]*LASTI, ENSAK, University of Sultan Moulay Slimane*

(Received 27 December 2023; Accepted 25 May 2024)

Among the difficult problems in mathematics is the problem of solving partial differential equations (PDEs). To date, there is no technique or method capable of solving all PDEs despite the large number of effective methods proposed. One finds in the literature, numerical methods such as the methods of finite differences, finite elements, finite volumes and their variants, semi-analytical methods such as the Variational Iterative Method, New Iterative Method and others. In recent years, we have witnessed the introduction of neural networks in solving PDEs. In this work, we will propose an adaptation of the method of embedding some physical laws into neural networks for solving Burgers–Huxley equation and revealing the dynamic behavior of the equation directly from spatio-temporal data. We will combine our technique with the Residual-based Adaptive Refinement method to improve its accuracy. We will give a comparison of the proposed method with those obtained by the New Iterative Method.

Keywords: *deep learning; physics-informed neural networks; generalized Burgers–Huxley equation; residual-based adaptive refinement.*

2010 MSC: 68T07, 68Txx, 35K57 **DOI:** 10.23939/mmc2024.02.505

1. Introduction

In this work, we studied the generalized Burgers–Huxley equation. It was first introduced thanks to the works of Bateman [1, 2], Burgers [3] for the Burgers equation and Hodgkin and Huxley [4] for the Huxley equation. The generalized non-linear partial differential Burgers–Huxley equation is given by:

$$\frac{\partial u}{\partial t} + \mu u^{\theta}\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} = \xi u(1-u^{\theta})(u^{\theta}-\Gamma), \tag{1}$$

for all $x \in [0,1]$ and $t \in [0,T]$, with the given boundary conditions:

$$\begin{cases} u(0,t) = \left[\dfrac{\Gamma}{2} + \dfrac{\Gamma}{2}\tanh\left[-\sigma\Gamma\left(\dfrac{\Gamma\mu}{1+\theta} - \dfrac{(1+\theta-\Gamma)(\rho-\mu)}{2(1+\theta)}\right)t\right]\right]^{1/\theta}, \\ u(1,t) = \left[\dfrac{\Gamma}{2} + \dfrac{\Gamma}{2}\tanh\left[\sigma\Gamma\left(1 - \left\{\dfrac{\Gamma\mu}{1+\theta} - \dfrac{(1+\theta-\Gamma)(\rho-\mu)}{2(1+\theta)}\right\}t\right)\right]\right]^{1/\theta}, \end{cases} \tag{2}$$

and initial condition:

$$u(x,0) = \left[\frac{\Gamma}{2} + \frac{\Gamma}{2}\tanh(\sigma\Gamma x)\right]^{1/\theta}. \tag{3}$$

The exact solution is given by [5]:

$$u(x,t) = \left[\frac{\Gamma}{2} + \frac{\Gamma}{2}\tanh\left[\sigma\Gamma\left(x - \left\{\frac{\Gamma\mu}{1+\theta} - \frac{(1+\theta-\Gamma)(\rho-\mu)}{2(1+\theta)}\right\}t\right)\right]\right]^{1/\theta}, \tag{4}$$

In above equations, $T > 0$, $\mu \in \mathbb{R}$ is the advection coefficient, $\xi \geqslant 0$ is the reaction coefficient, $\Gamma \in (0,1)$, $\theta > 0$, $\sigma = \theta(\rho-\mu)/4(1+\theta)$ and $\rho = \sqrt{\mu^2 + 4\xi(1+\theta)}$.

The Burgers–Huxley equation is used to model the complex interplay between various physical phenomena, such as reaction mechanisms, diffusion transport and convection effects. It finds its application in various fields such as biology, chemistry, combustion, mathematics and engineering [6].

Due to its wide applications, the researchers applied many numerical methods to approximate the solution such as finite difference method [7]. Others used semi-analytical methods such as the New Iterative Method (NIM) [5], the Variational Iteration Method (VIM) [8], the Adomian Decomposition Method (ADM) [9] and the differential transform method (DTM) [10].

Over the last few decades, neural networks have been achieving spectacular success in many machine learning fields such as speech recognition [11], image recognition [12, 13] and natural language processing [14]. Physics-informed neural networks (PINNs) have been used to solve forward and inverse differential problems. Unlike classical numerical schemes for solving partial differential equations, PINNs are non-data-driven and mesh free models that satisfy the given initial and boundary conditions as well as the governing PDE. Many authors has used this approach in order to solve stiff PDEs like the Burgers–Huxley equation [15–17].

In the current work, we will evaluate the performance of PINN in solving Generalized Burgers–Huxley equation and compare it with the performance of the New Iterative Method. The structure of this paper is as follows: an explanation of the PINN methodology will be presented in Section 2. In Section 3, the performances of the PINN in solving four Burgers-Huxley problems will be investigated. Conclusions are presented in Section 4.

2. Methodology

In this section, we will discussed the PINNs methodology for solving partial differential equations and we will also present briefly the residual-based adaptive refinement method to improve the PINNs accuracy.

2.1. Deep neural networks (DNN)

In the PINN framework, the training model is based on a fully connected neural network (FCNN) parameterized by a set of parameters θ.

The network consists of an input layer, an output layer and n hidden layers. Each of these hidden layer takes $X = [x_1, x_2, \ldots, x_i]$ as an input and outputs $Y = [y_1, y_2, \ldots, y_i]$ through a nonlinear differentiable activation function $\sigma(\cdot)$ such as:

$$y_i = \sigma(w_i x_i + b_i) \quad 1 \leqslant i \leqslant n, \tag{5}$$

where trainable hyperparameters w_i and b_i represent the weight and bias of the ith layer of the neural network that will be updated during the training phase.

2.2. Physics-informed neural networks

Physics-informed neural networks [18] are deep neural networks trained to solve forward and inverse differential problems while respecting the physical laws given by the PDE. They solve PDEs expressed in the given form:

$$\begin{cases} \mathcal{D}(u(x,t);\lambda) = f(x,t) & x \in \Omega \subset \mathbb{R}^d,\ t \in [0,T],\ T > 0, \\ u(x,0) = g(x), & \\ u(x,t) = h(x,t) & x \in \partial\Omega. \end{cases} \tag{6}$$

Here, u represents the unknown solution of the PDE, $\mathcal{D}$ represents the non-linear differential operator, g and h represent the initial and boundary functions and λ are the parameters related to the physics.

In this case, the PINN must learn to approximate the solution of the PDE through finding the hyperparameters θ defining the network by minimizing a weighted loss function given by [19]:

$$\mathcal{L}oss(\theta, \mathcal{T}) = \omega_{\mathcal{D}}\, \mathcal{L}oss_{\mathcal{D}}(\theta, \mathcal{T}_d) + \omega_{\mathcal{BC}}\, \mathcal{L}oss_{\mathcal{BC}}(\theta, \mathcal{T}_{bc}) + \omega_{\mathcal{IC}}\, \mathcal{L}oss_{\mathcal{IC}}(\theta, \mathcal{T}_{ic}), \tag{7}$$

where:

$$\mathcal{L}oss_{\mathcal{D}}(\theta, \mathcal{T}_d) = \frac{1}{|\mathcal{T}_d|} \sum_{(x,t)\in\mathcal{T}_d} \|\mathcal{D}(\hat{u}_\theta(x,t);\lambda) - f(x,t)\|_2^2, \tag{8}$$

$$\mathcal{L}oss_{\mathcal{BC}}(\theta, \mathcal{T}_{bc}) = \frac{1}{|\mathcal{T}_{bc}|} \sum_{(x,t)\in\mathcal{T}_{bc}} \|\hat{u}_\theta(x,t) - h(x,t)\|_2^2, \tag{9}$$

$$\mathcal{L}oss_{\mathcal{IC}}(\theta, \mathcal{T}_{ic}) = \frac{1}{|\mathcal{T}_{ic}|} \sum_{x \in \mathcal{T}_{ic}} \|\hat{u}_\theta(x, 0) - g(x)\|_2^2, \tag{10}$$

in which $\mathcal{L}oss_{\mathcal{D}}$ represents the residual of the governing PDE, $\mathcal{L}oss_{\mathcal{BC}}$ is the residual of the boundary conditions and $\mathcal{L}oss_{\mathcal{IC}}$ is the residual of the initial condition. $\mathcal{T}_d$ is the set of points inside the domain Ω, $\mathcal{T}_{bc}$ is the set of points on the boundary $\partial\Omega$ and $\mathcal{T}_{ic}$ is the set of initial points. $\omega_{\mathcal{D}}$, $\omega_{\mathcal{BC}}$ and $\omega_{\mathcal{IC}}$ represent the weighting coefficients of $\mathcal{L}oss_{\mathcal{D}}$, $\mathcal{L}oss_{\mathcal{BC}}$ and $\mathcal{L}oss_{\mathcal{IC}}$ respectively.

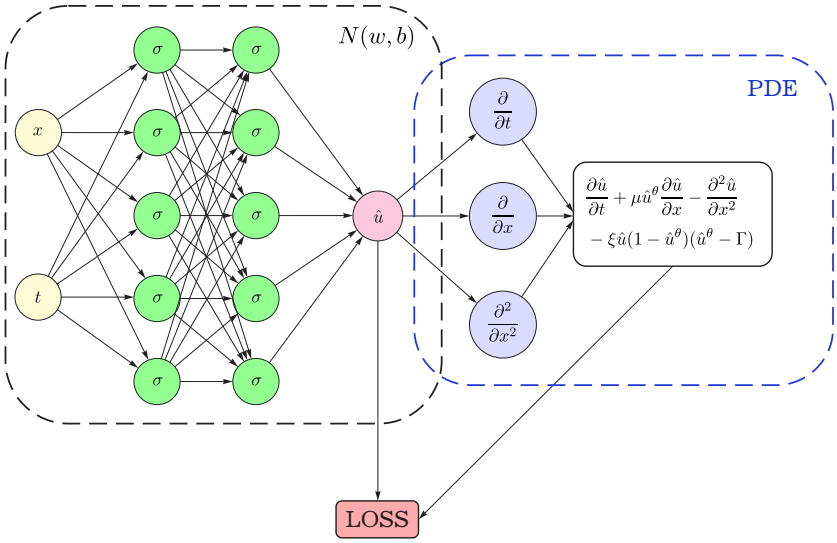


Fig. 1. Solving the Burgers-Huxley equation with PINN.

2.3. Residual-based adaptive refinement

In general, PINNs residual points $\mathcal{T}_d$ are randomly distributed in the domain Ω, therefore the authors in [20] introduced the Residual-based Adaptive Refinement (RAR) method for improving the distribution of training residual points and consequently improving the accuracy and training efficiency of the PINN. The main idea of the RAR method is to adaptively add more residual points in the locations where the PDE residual is large during the training of the network.

The RAR method works as follows: we first start by training our PINN on the training set $\mathcal{T}$ for a certain number of iterations. Then we compute the PDE residual at random points in the domain. We then start adding m new points to the training set where the residual is the largest. We repeat those steps for a number n of iterations.

3. Results and discussions

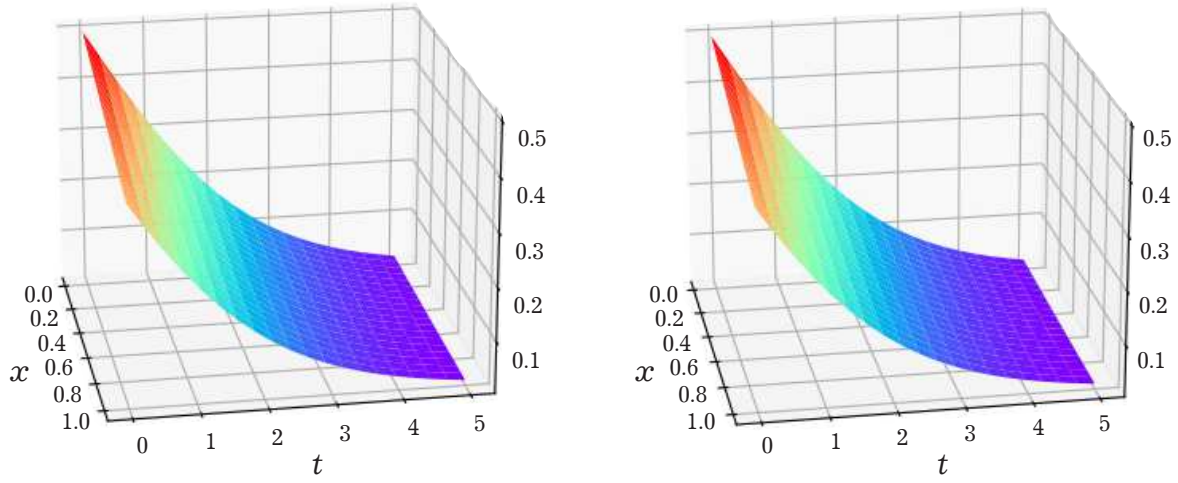
We test the efficiency of our PINN on four different cases. We implemented our PINN using Google Collaboratory notebooks `https://colab.research.google.com/?hl=fr` on its GPU T4. We sample the training points using the Latin Hypercube Sampling (LHS) and we choose "tanh(·)" as the activation function for the four cases.

Case 1. We choose, in this case, $\mu = -1$ and $\xi = \Gamma = \theta = 1$. We choose 2000 training residual points inside the domain, and 500 training points sampled on its boundary and 800 initial residual points for the initial conditions. We use a FCNN of depth 4 (i.e., 3 hidden layers) with 50 neurons on each layer, we trained our NN with 30000 iterations using the optimizer Adam then we continue to train our NN using the optimizer L-BFGS to achieve a smaller loss. We present the results obtained by our network in Table 1. The exact and the predicted solution were plotted using our PINN. As shown in Figure 2, our proposed framework produced good predictions.

To test the ability of the RAR method in improving the accuracy of PINN, we compare the absolute errors of our PINN before and after 10 iterations of RAR. The results are presented in Table 2. It can be observed that after using the RAR method, the errors has decreased which shows its efficiency.

Table 1. Absolute errors for case 1 using PINN solution.

x	t	Exact value	Predicted value	Absolute error
0.1	1	0.31002552	0.31004667	2.11596×10^{-5}
	2	0.17508627	0.17506838	1.78962×10^{-5}
	3	0.09112296	0.09109049	3.24696×10^{-5}
	4	0.04521747	0.04525789	4.04194×10^{-5}
	5	0.02188127	0.02182179	5.94798×10^{-5}
0.3	1	0.28905050	0.28906330	1.28149×10^{-5}
	2	0.16110895	0.16109467	1.42902×10^{-5}
	3	0.08317270	0.08314499	2.77161×10^{-5}
	4	0.04109128	0.04112641	3.51294×10^{-5}
	5	0.01984031	0.01980489	3.54219×10^{-5}
0.9	1	0.23147522	0.23148110	5.88595×10^{-6}
	2	0.12455336	0.12454838	4.97698×10^{-6}
	3	0.06297336	0.06296322	1.01327×10^{-5}
	4	0.03076886	0.03079020	2.13421×10^{-5}
	5	0.01477403	0.01474961	2.44211×10^{-5}



a (Predicted solution) ***b*** (Exact solution)

Fig. 2. Predicted and exact solution for case 1.

Case 2. For this case, we have $\mu = \xi = 1$, $\Gamma = 0.001$ and $\theta = 1$. We choose 20000 training residual points sampled inside the domain, and 1500 training points sampled on the boundary. We also include 1800 initial residual points for the initial conditions. We use a FCNN of depth 4 (i.e., 3 hidden layers) with 123 neurons on each layer, we trained our NN with 15000 iterations using Adam then we continue to train the NN using the optimizer L-BFGS to achieve a smaller loss. Next, we improve the accuracy of our network by using the RAR method for 20 iterations. In Table 3, a comparison between our PINN and the NIM method [5] is given. From the table, it is noticed that our method gives accurate solutions in comparison to the New Iterative Method.

Case 3. In this case, we choose $\mu = \xi = 1$, $\Gamma = 0.01$ and $\theta = 2$. We choose 30000 training points inside the domain, 1500 training points sampled on its boundary and 1500 initial residual points for the initial conditions. We use a FCNN of depth 4 with 128 neurons on each layer, we trained our network for 15000 iterations using Adam then we train our network using L-BFGS in order to acheive a smaller loss. In Table 4, we compare our PINN with the NIM method [5]. From the comparison it is observed that proposed technique produced accurate solution.

Case 4. We choose the same values as case 3 but with $\theta = 4$. We implemented the same NN used in case 3. The results obtained with our model are compared with the NIM method [5] in Table 5. According to these results, the accuracy of the PINN solution is comparable to that of the NIM solution.

Table 2. Absolute errors for case 1 before and after using RAR method.

x	t	Absolute error before RAR	Absolute error after RAR
0.1	1	2.11596×10^{-5}	1.12653×10^{-5}
	2	1.78962×10^{-5}	1.59442×10^{-6}
	3	3.24696×10^{-5}	3.21120×10^{-6}
	4	4.04194×10^{-5}	1.37091×10^{-5}
	5	5.94798×10^{-5}	2.60230×10^{-5}
0.3	1	1.28149×10^{-5}	1.00732×10^{-5}
	2	1.42902×10^{-5}	2.19047×10^{-6}
	3	2.77161×10^{-5}	1.63913×10^{-7}
	4	3.51294×10^{-5}	9.97633×10^{-6}
	5	3.54219×10^{-5}	2.36239×10^{-5}
0.5	1	7.62939×10^{-6}	1.11461×10^{-5}
	2	1.04457×10^{-5}	4.30644×10^{-6}
	3	2.16365×10^{-5}	3.65823×10^{-6}
	4	3.03574×10^{-5}	6.06850×10^{-6}
	5	2.14968×10^{-5}	2.24095×10^{-5}
0.9	1	5.88595×10^{-6}	1.90884×10^{-5}
	2	4.97698×10^{-6}	6.02752×10^{-6}
	3	1.01327×10^{-5}	1.25393×10^{-5}
	4	2.13421×10^{-5}	7.78586×10^{-7}
	5	2.44211×10^{-5}	2.51625×10^{-5}

Table 3. Absolute errors for case 2 using PINN and NIM solutions.

x	t	Our PINN	NIM [5]
0.1	0.05	1.74622×10^{-10}	1.87405×10^{-8}
	0.1	5.12227×10^{-9}	3.74811×10^{-8}
	1	4.48781×10^{-8}	3.74811×10^{-7}
0.5	0.05	7.56699×10^{-9}	1.87405×10^{-8}
	0.1	5.47152×10^{-9}	3.74811×10^{-8}
	1	9.03382×10^{-8}	3.74811×10^{-7}
0.9	0.05	2.85217×10^{-8}	1.87405×10^{-8}
	0.1	2.78814×10^{-8}	3.74811×10^{-8}
	1	8.95815×10^{-8}	3.74811×10^{-7}

Table 4. Absolute errors for case 3 using PINN and NIM solutions.

x	t	Our PINN	NIM [5]
0.1	0.1	2.19047×10^{-6}	5.51552×10^{-5}
	0.2	7.39097×10^{-6}	1.10340×10^{-4}
	0.3	1.52140×10^{-5}	1.65525×10^{-4}
	0.4	2.13459×10^{-5}	2.2070×10^{-4}
	0.5	2.58684×10^{-5}	2.75945×10^{-4}
0.3	0.1	2.47210×10^{-5}	5.51380×10^{-5}
	0.2	3.52114×10^{-5}	1.10291×10^{-4}
	0.3	4.39211×10^{-5}	1.65455×10^{-4}
	0.4	5.09247×10^{-5}	2.20632×10^{-4}
	0.5	5.63114×10^{-5}	2.75830×10^{-4}
0.5	0.1	3.38554×10^{-5}	5.51131×10^{-5}
	0.2	4.45619×10^{-5}	1.10244×10^{-4}
	0.3	5.34951×10^{-5}	1.65400×10^{-4}
	0.4	6.07296×10^{-5}	2.20541×10^{-4}
	0.5	6.63548×10^{-5}	2.75714×10^{-4}

Table 5. Absolute errors for case 4 using PINN and NIM solutions.

x	t	Our PINN	NIM [5]
0.1	0.1	2.17854×10^{-5}	2.17787×10^{-4}
	0.2	7.80820×10^{-6}	4.35691×10^{-4}
	0.3	3.51071×10^{-5}	6.53717×10^{-4}
	0.4	5.99026×10^{-5}	8.71833×10^{-4}
	0.5	8.20457×10^{-5}	1.09050×10^{-3}
0.3	0.1	9.10162×10^{-5}	2.17548×10^{-4}
	0.2	1.21890×10^{-4}	4.35228×10^{-4}
	0.3	1.50350×10^{-4}	6.53010×10^{-4}
	0.4	1.76250×10^{-4}	8.70915×10^{-4}
	0.5	1.99380×10^{-4}	1.08891×10^{-3}
0.5	0.1	1.29700×10^{-4}	2.17320×10^{-4}
	0.2	1.60370×10^{-4}	4.34745×10^{-4}
	0.3	1.88650×10^{-4}	6.52311×10^{-4}
	0.4	2.14310×10^{-4}	8.69963×10^{-4}
	0.5	2.37260×10^{-4}	1.08755×10^{-3}

4. Conclusions

In this study, we test the efficiency of PINNs in solving the nonlinear Burgers–Huxley equation. Four cases of the equation were investigated using the proposed framework. In the two first cases, we apply the RAR method to our PINN to improve its accuracy.

We compared the results obtained by PINN with those obtained by the NIM method [5]. It was found that the solutions predicted by PINN are very accurate and better than those of NIM.

For future work, we will compare the performance of PINN with other neural network techniques, such as Recurrent Neural Networks (RNN).

[1] Bateman H. Some recent researches on the motion of fluids. Monthly Weather Review. **43** (4), 163–170 (1915).

[2] Whitham G. B. Linear and Nonlinear Waves. Wiley, New York (2011).

[3] Burgers J. M. A mathematical model illustrating the theory of turbulence. Advances in Applied Mechanics. **1**, 171–199 (1948).

[4] Hodgkin A. L., Huxley A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of Physiology. **117** (4), 500–544 (1952).

[5] Batiha B., Ghanim F., Batiha K. Application of the New Iterative Method (NIM) to the Generalized Burgers–Huxley Equation. Symmetry. **15** (3), 688 (2023).

[6] Satsuma J. Topics in soliton theory and exactly solvable nonlinear equations. World Scientifc, Singapore (1987).

[7] Inan B. Finite difference methods for the generalized Huxley and Burgers–Huxley equations. Kuwait Journal of Science. **44** (3), 20–27 (2017).

[8] Batiha B., Noorani M. S. M., Hashim I. Application of variational iteration method to the generalized Burgers–Huxley equation. Chaos, Solitons & Fractals. **36** (3), 660–663 (2008).

[9] Hashim I., Noorani M., Al-Hadidi M. S. Solving the generalized Burgers–Huxley equation using the Adomian decomposition method. Mathematical and Computer Modelling. **43** (11–12), 1404–1411 (2006).

[10] Biazar J., Mohammadi F. Application of Differential Transform Method to the Generalized Burgers-Huxley Equation. Applications and Applied Mathematics: An International Journal. **05** (10), 1726-1740 (2011).

[11] Zhang S., Chen M., Chen J., Li Y.-F., Wu Y., Li M., Zhu C. Combining cross-modal knowledge transfer and semi-supervised learning for speech emotion recognition. Knowledge-Based Systems. **229**, 107340 (2021).

[12] Gao Y., Mosalam K. M. Deep Transfer Learning for Image-Based Structural Damage Recognition. Computer-Aided Civil and Infrastructure Engineering. **33** (9), 748–768 (2018).

[13] Yang X., Zhang Y., Lv W., Wang D. Image recognition of wind turbine blade damage based on a deep learning model with transfer learning and an ensemble learning classifier. Renewable Energy. **163**, 386–397 (2021).

[14] Ruder S., Peters M. E., Swayamdipta S., Wolf T. Transfer Learning in Natural Language Processing. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials. 15–18 (2019).

[15] Wen Y., Chaolu T. Study of Burgers–Huxley Equation Using Neural Network Method. Axioms. **12** (5), 429 (2013).

[16] Panghal S., Kumar M. Approximate Analytic Solution of Burger Huxley Equation Using Feed-Forward Artificial Neural Network. Neural Processing Letters. **53**, 2147–2163 (2021).

[17] Kumar H., Yadav N., Nagar A. K. Numerical solution of Generalized Burger–Huxley & Huxley's equation using Deep Galerkin neural network method. Engineering Applications of Artificial Intelligence. **115**, 105289 (2022).

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

[19] Cuomo S., Di Cola V. S., Giampaolo F., Rozza G., Raissi M., Piccialli F. Scientific Machine Learning Through Physics-Informed Neural Networks: Where we are and What's Next. Journal of Scientific Computing. **92**, 88 (2022).

[20] Wu C., Zhu M., Tan Q., Kartha Y., Lu L. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering. **403** (A), 115671 (2023).

Вбудовування фізичних законів у глибоку нейронну мережу для розв'язування узагальненого рівняння Бюргерса-Гакслі

Харірі І.[1], Радід А.[1], Рофір К.[2]

[1] *LMFA, FSAC, Університет Хасана II Касабланки*
[2] *LASTI, ENSAK, Університет Султана Мулая Слімана*

До складних задач математики належить задача розв'язування диференціальних рівнянь із частинними похідними (PDE). На сьогоднішній день не існує техніки чи методу, здатного розв'язати всі PDE, незважаючи на велику кількість запропонованих ефективних методів. У літературі можна знайти чисельні методи, такі як методи скінченних різниць, скінченних елементів, скінченних об'ємів та їх варіанти, напіваналітичні методи, такі як варіаційний ітеративний метод, новий ітеративний метод та інші. В останні роки ми стали свідками впровадження нейронних мереж у розв'язуванні PDE. У цій роботі пропонуємо адаптацію методу вбудовування деяких фізичних законів у нейронні мережі для розв'язання рівняння Бюргерса–Гакслі та виявлення динамічної поведінки рівняння безпосередньо з просторово-часових даних. Поєднуємо запропоновану техніку з методом адаптивного уточнення на основі нев'язок, щоб підвищити його точність. Наведено порівняння запропонованого методу з отриманими за допомогою нового ітераційного методу.

Ключові слова: *глибоке навчання; фізичні нейронні мережі; узагальнене рівняння Бюргерса–Хакслі; адаптивне уточнення на основі залишків.*