Vol. 6, No. 3, 2024

Yaroslav Sokolovskyy [1], Tetiana Samotii [2]

[1] Computer Aided Design Department, Lviv Polytechnic National University, Ukraine, Lviv, 12 S.Bandery Street, E-mail: yaroslav.i.sokolovskyi@lpnu.ua, ORCID 0000-0003-4866-2575

[2]Software Engineering Department, Ukrainian National Forestry University, Ukraine, Lviv, 103 H.Chuprinky Str., E-mail: t.samotiy@nltu.edu.ua, ORCID 0000-0002-1317-8786

ADAPTIVE FRACTIONAL NEURAL ALGORITHM FOR MODELING HEAT-AND-MASS TRANSFER

Received: November 12, 2024 / Revised: November 20, 2024 / Accepted: November 25, 2024

https://doi.org/

Abstract. A fractional neural network with an adaptive learning rate has been proposed for modeling the dynamics of non-isothermal heat and mass transfer in capillary-porous materials, taking into account the memory effect and spatial nonlocality. The proposed approach employs a decoupled neural network architecture based on loss functions that reflect the physical characteristics of the investigated process. A stepwise training method is utilized to reduce sensitivity to errors and disruptions. The network structure has been analyzed, its parameters optimized, and appropriate activation functions and regularization methods selected to achieve high accuracy and reliability in modeling results.

Keywords: Fractional Derivatives, Machine Learning in Physical Processes, Neural Network Modeling, Adaptive Learning Rate, Capillary-Porous Materials

Introduction

At the current stage of scientific development, an important task is the creation of mathematical models to describe nonequilibrium physical processes, especially in fractal systems, which exhibit a complex spatiotemporal structure. These systems are characterized by phenomena such as memory, self-organization, and spatial non-locality, requiring the use of fractal dimension geometry for their adequate modeling. This approach is widely applied in various fields of science and technology, including mechanics, physics, electrical engineering, medicine, economics, and sociology. The foundation of such models is fractional order integrodifferential equations, which account for historical effects that determine the behavior of systems in time and space.

The fractional calculus of integration and differentiation allows for the modeling of memory systems, considering how previous states influence the system's dynamics. The time fractional exponent determines the degree of the system's "memory": whether it retains all information about past states or loses it over time. Spatial fractional derivatives, in turn, reflect the self-similarity and heterogeneity of the internal structure of the modeled environment. This enables a more accurate description of processes in fractal media, where parts of the system exhibit the same structural properties as the system as a whole, and interactions can occur over large distances. This approach provides a deeper understanding of the nature of physical phenomena and allows for effective solutions to complex problems.

One of the current directions for solving fractional integrodifferential equations is the application of neural network approaches, which demonstrate significant potential in modeling complex systems. Currently, architectures of neural networks are being developed that adapt to the specificities of fractional problems. Studying the structure of such networks and analyzing their performance is crucial for achieving high accuracy in results. Further research in this direction will not only improve computational accuracy but also optimize the speed of problem-solving, opening new perspectives for the practical application of fractional models in various fields.

In this work, we apply an adapted neural network method based on the fPINN (Fractional Physics-Informed Neural Networks) architecture [1], [2] to solve a mathematical model of non-isothermal heat and moisture exchange, taking into account fractional derivatives, and investigate the structure of the neural network. This is important for enhancing the efficiency of modeling processes in environments with fractal structures.

The object of this study is the physical processes considering the heterogeneity of environments and memory effects occurring in capillary-porous materials during heat and moisture exchange.

The subject of the study is the model of a fractal neural network, its characteristics, and methods that ensure the accuracy of modeling these processes.

The main goal of the work is to analyze the fractal neural network model for predicting the evolution of temperature and moisture fields. The investigation of the network structure, optimization of its parameters, selection of activation functions, and regularization methods are aimed at achieving high accuracy and reliability in the modeling results.

The practical significance of the work lies in determining the optimal values of the key parameters of the neural network, which take into account the features of heat and moisture exchange in environments with fractal heterogeneity. These parameters include the number of layers, the number of neurons in each layer, the selection of activation functions, and regularization methods. The obtained parameter values can be used to construct efficient neural networks capable of ensuring high prediction accuracy for processes in capillary-porous materials. This allows for the optimization of modeling processes for their effective practical application in various fields and creates opportunities for further research in the direction of modeling complex multiphysical processes occurring in heterogeneous environments.

Problem Statement

Methods of fractional derivatives and integrals are gaining increasing popularity in modeling physical processes in fields such as mechanics, physics, electrical engineering, medicine, and economics, as they allow for the consideration of memory effects and the fractal structure of materials. Such processes include, in particular, transport processes in amorphous semiconductors, aerogels, and porous media [3-7]. While a significant number of scientific works are dedicated to the development of analytical methods for implementing such models [8-10], numerical approaches have become the most widespread [11-13]. However, given the complexity of implementation, instability of numerical methods, and the need for significant computational resources to solve such problems, neural network approaches are attracting increasing attention as an effective alternative in this area.

For example, in [14], a method was proposed for solving ordinary differential equations and partial differential equations using artificial neural networks. Works [15,16] introduced two main ideas that allow the integration of physical laws into neural networks while avoiding the need for spatial discretization. Today, these approaches are widely used for the numerical solution of partial differential equations (PDEs) using neural networks. However, at the time, their practical application was limited due to technological barriers, particularly the insufficient computational power of computers.

The situation changed with the emergence of computational platforms with GPU acceleration and high-performance machine learning frameworks, which rekindled interest in these methods. Special attention has been given to data-driven training methods for partial differential equations. One of the most influential approaches that sparked a renewed interest in solving PDEs is the Physics-Informed Neural Networks (PINNs) methodology, proposed by the authors in 2019 [17]. This approach demonstrated great potential in various application scenarios [18,19]. Specialized deep learning libraries, such as DeepXDE [20], were also developed to solve differential equations.

The PINN architecture is based on a fully connected neural network. However, due to some drawbacks of the basic PINN architecture, many researchers have sought to extend its capabilities through various approaches. This led to the creation of variants such as conservative PINN (cPINN) and nonlocal PINN (nPINN) [21,22]. Moreover, some authors aimed to improve performance and achieve more accurate results in modeling tasks by combining PINN with other types of neural networks, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [23,24]. In 2022, a neural network method based on Fractional Physics-Informed Neural Networks (fPINNs) was proposed, enabling efficient modeling of physical processes in fractal media [25]. The authors of [26] developed a series expansion approach for more accurate and efficient solutions to linear partial differential equations of higher order. Work [27] introduced an enhanced fPINN variant combining neural networks with the spectral collocation method, reducing the required discrete points for fractional operators, improving computational efficiency and accuracy. Additionally, [28] proposed an improved Monte Carlo method for fractional and tempered fractional partial differential equations, utilizing Gaussian quadrature to minimize variance and errors, effectively solving high-dimensional problems. Overall, neural network approaches for fractional-order problems remain in an early stage, with limited research currently available.

Main Material Presentation

The mathematical model for describing the behavior of the dynamic heat and moisture exchange process in one-dimensional space can be represented using a system of fractional differential equations as follows:

$$\frac{\partial^{\alpha} T(x,t)}{\partial t^{\alpha}} = a\frac{\partial^{\beta} T(x,t)}{\partial x^{\beta}} + \frac{\varepsilon\rho_0 r}{c\rho}\frac{\partial^{\alpha} U(x,t)}{\partial t^{\alpha}}, \quad (1)$$

$$\frac{\partial^{\alpha} U(x,t)}{\partial t^{\alpha}} = d\frac{\partial^{\beta} U(x,t)}{\partial x^{\beta}} + d\delta\frac{\partial^{\beta} T(x,t)}{\partial x^{\beta}}, \quad (2)$$

The equations (1) and (2) must be satisfied with the following established initial and boundary conditions:

$$T(x,0) = T_i(x), U(x,0) = U_i(x), \quad (3)$$

$$\lambda\frac{\partial^{\gamma} T(t,0)}{\partial x^{\gamma}} + \sigma\beta\left(U(t,0) - U_p\right) = \alpha^*\left(T(t,0) - t_c\right), \quad (4)$$

$$\lambda\frac{\partial^{\gamma} T(t,l)}{\partial x^{\gamma}} + \sigma\beta\left(U(t,l) - U_p\right) = \alpha^*\left(T(t,l) - t_c\right), \quad (5)$$

$$d\frac{\partial^{\gamma} U(t,0)}{\partial x^{\gamma}} + d\delta\frac{\partial^{\gamma} T(t,0)}{\partial x^{\gamma}} = \beta\left(U_p - U(t,0)\right), \quad (6)$$

$$d\frac{\partial^{\gamma} U(t,l)}{\partial x^{\gamma}} + d\delta\frac{\partial^{\gamma} T(t,l)}{\partial x^{\gamma}} = \beta\left(U_p - U(t,l)\right), \quad (7)$$

where $(t,x)\in\Omega, \Omega=[0,T]\times[0,l]$; t — is the time axis, and x —is the spatial axis; $U(t,\boldsymbol{x})-$is the moisture function, which is to be determined, $T(t,\boldsymbol{x})-$is the temperature function, which is to be determined; a – is the temperature conductivity coefficient; $\lambda(T,U)-$is the moisture conductivity coefficient; $d(T,U)-$ moisture conductivity coefficient; $c(T,U)-$is the specific heat capacity; $\sigma=\rho_0(1-\varepsilon), \rho_0 -$ is the density; U_p — is the equilibrium moisture content, which is a function of temperature t_c and the relative moisture of the surrounding environment; U_0 — is the initial moisture content; T_0 —is the initial temperature; β are the moisture exchange coefficients; α^* — are the heat exchange coefficients; α is the fractional time derivative order $(0<\alpha\leq 1)$; β are the fractional spatial derivative orders $(1<\beta\leq 2)$; γ — is another fractional order parameter $(0<\gamma\leq 1)$.

The fractional derivative, as defined in the model (1)-(6), is specified in the Caputo interpretation and is given as follows [29]:

$$\partial_t^\alpha f(x,t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-\tau)^{-\alpha} f'(t) \partial\tau, \; 0<\alpha\le 1, \tag{8}$$

here, $\Gamma(\cdot)$ — denotes the gamma function, and α represents the fractional time derivative.

The fractional-order parameters β (where $(1<\beta\le 2)$) and γ (where $(0<\gamma\le 1)$), which are derivatives with respect to spatial coordinates, are defined in the Grunwald-Letnikov interpretation and are expressed as follows [28]:

$$\partial_x^\beta f(x,t) == \lim_{N\to\infty} \frac{1}{\Gamma(-\beta)} \left(\frac{x-a}{N}\right)^{-\beta} \sum_{k=0}^{N-1} \frac{\Gamma(k-\beta)}{\Gamma(k+1)} f\left(x - k\frac{x-\alpha}{N}\right), \tag{9}$$

The Grunwald-Letnikov derivative captures not only the local variations of a function but also accounts for distant interrelations between points, enabling the modeling of the effect of self-similar heterogeneity.

Methodology. Physics-Informed Neural Networks (PINNs) [17] incorporate physical laws in the form of partial differential equations into the loss function, enabling the model to learn hidden physical patterns. For partial differential equations, derivatives of any order with respect to spatial or temporal coordinates are computed using automatic differentiation, ensuring high computational accuracy. However, calculating fractional derivatives is more complex and requires alternative approaches.

To address this, the fPINNs method [25] was developed, utilizing various numerical methods to approximate fractional derivative operators in the time and spatial domains.

In this work, time-fractional and space-fractional differential operators are approximated using the corresponding numerical schemes (10) and (11), which are subsequently integrated into the loss functions of the fractional neural network [2].

$$\partial_t^\alpha f(x,t_k) = \frac{\omega^{-\alpha}}{\Gamma(1-\alpha)} \left[b_0 f(x,t_k) - b_{k-1} f(x,t_0) + \sum_{i=1}^{k-1} (b_i - b_{i-1}) f(x,t_{k-i}) \right], \tag{10}$$

where $b_i = (i+1)^{1-\alpha} - i^{1-\alpha}, i = 0,1,\ldots,K-1;\ \omega = \frac{T}{N},\ t_k = k\omega, k = 0,\ldots,K.$

$$\partial_x^\beta f(x,t) \approx \frac{1}{h^\beta} \sum_{i=0}^{n} \upsilon_i^\beta f(x_{n+1-i}, t), \tag{11}$$

where $\upsilon_i = \frac{\Gamma(i-\beta)}{\Gamma(i+1)\Gamma(-\beta)},\ h = x_n / i, n = 0,\ldots,N.$

The structure of this network is illustrated in Figure 1. As shown in the figure, the architecture consists of two independent fully connected feedforward neural networks operating in parallel. In each individual network, every neuron in the current layer is connected to all neurons in the next layer, with the output values of one layer serving as the input data for the next.

The network model implements a transformation from the input layer $y_0 = z(x,t),\ y_0 \in \Omega \subset \mathbb{R}^2$, which is shared by both networks, to the output layer $y_L \in \mathbb{R}^2$, which combines the outputs of the two networks:

$$y_L := (y_L^1, y_L^2) = \begin{cases} y_L^1 = N_U(z,\theta^1) = \sum_{l=1}^{L} a_l^1 (W_l^1 y_{l-1}^1 + b_l^1) \\ y_L^2 = N_T(z,\theta^2) = \sum_{l=1}^{L} a_l^2 (W_l^2 y_{l-1}^2 + b_l^2) \end{cases} \tag{12}$$

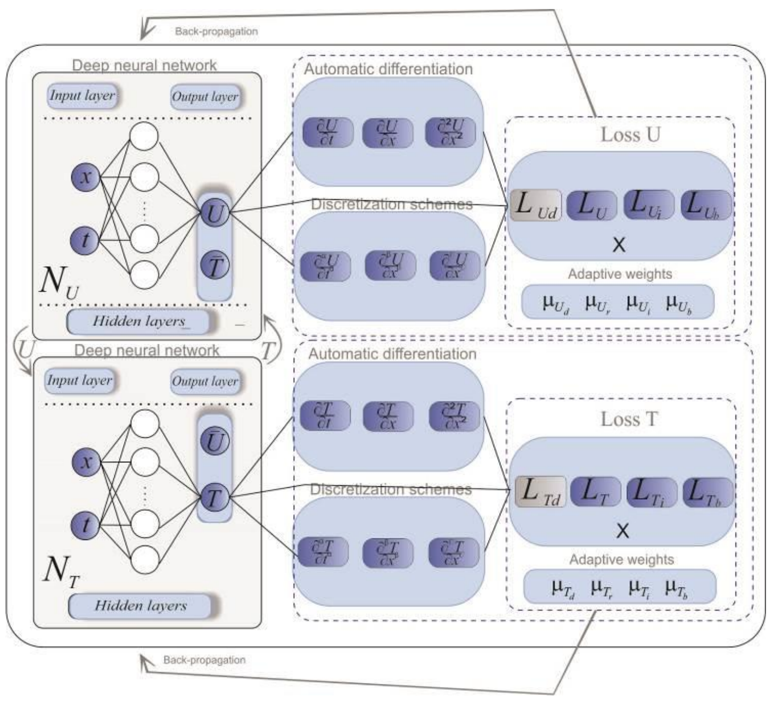


Fig 1. Structure of a neural network.

Here, y_l represents a layer located between the input and output layers, $l = 1, 2, ..., L-1$; $\theta^1 := (W_l^1, b_l^1), \theta^2 := (W_l^2, b_l^2) -$ are the parameters of the respective networks, depending on the weight coefficients w_l^i and biases b_l^i, $i = 1, 2$; a_l^i -is the activation function that performs component-wise processing; N_U and N_T are the outputs of the respective individual neural networks.

According to the described approach, the functions $U(t, \boldsymbol{x})$ and $T(t, \boldsymbol{x})$, to be determined, are modeled using a fully connected neural network (12). The outputs of the network, N_U and N_T, serve as approximations of the sought functions for moisture content and temperature:

$$\begin{cases} N_U = \tilde{U}(z) \\ N_T = \tilde{T}(z) \end{cases}. \tag{13}$$

In this model, the loss functions for the networks N_U and N_T are formulated as mean squared error (MSE), incorporating the residual losses of the equations L_{U_r}, L_{T_r}, the boundary condition losses L_{U_b}, L_{T_b}, the initial condition losses L_{U_i}, L_{T_i}, and the losses between the training data L_{U_d}, L_{T_d} and the network predictions N_U and N_T. The loss functions are expressed as follows:

$$\begin{cases} L_{N_U}\left(\theta^1\right) = \mu_{U_d} L_{U_d}\left(\theta^1\right) + \mu_{U_r} L_{U_r}\left(\theta^1\right) + \mu_{U_i} L_{U_i}\left(\theta^1\right) + \mu_{U_b} L_{U_b}\left(\theta^1\right), \\ L_{N_T}\left(\theta^2\right) = \mu_{T_d} L_{T_d}\left(\theta^2\right) + \mu_{T_r} L_{T_r}\left(\theta^2\right) + \mu_{T_i} L_{T_i}\left(\theta^2\right) + \mu_{T_b} L_{T_b}\left(\theta^2\right), \end{cases} \tag{14}$$

here, the expressions $L_{U_r}, L_{T_r}, L_{U_i}, L_{T_i}, L_{U_b}, L_{T_b}$ – are defined based on equations (1)–(7), considering the numerical discretization schemes (10) and (11), and are presented by formulas (15)–(19), while $\mu_{T_d}, \mu_{U_d}, \mu_{U_r}, \mu_{T_r}, \mu_{U_b}, \mu_{U_i}, \mu_{T_b}, \mu_{T_i}$ represents the weighting parameters.

$$L_U = \frac{1}{N_r} \times \sum_{k=1}^{K} \sum_{n=1}^{N-1} \left[\frac{\omega^{-\alpha}}{\Gamma(2-\alpha)} [b_0 U_{k,n} - b_{k-1} U_{0,n} + \sum_{i=1}^{k-1} \left(b_i - b_{i-1}\right) U_{k-i,n}] - \frac{d}{h_1^\beta} \sum_{i=0}^{n} \upsilon_i^\beta U_{k,n+1-i} - \frac{d\delta}{h_1^\beta} \sum_{i=0}^{n} \upsilon_j^\beta T_{k,n+1-i}, \right]^2 \tag{15}$$

$$L_T = \frac{1}{N_r} \times \sum_{k=1}^{K} \sum_{n=1}^{N-1} \left[\begin{array}{l} \frac{\omega^{-\alpha}}{\Gamma(2-\alpha)} [b_0 T_{k,n} - b_{k-1} T_{0,n} + \sum_{i=1}^{k-1} (b_i - b_{i-1}) T_{k-i,n}] - \frac{a}{h^{\beta}} \sum_{i=0}^{n} \upsilon_i^{\beta} T_{k,n+1-i} - \\ - \frac{\omega^{-\alpha} \varepsilon \rho_0 r}{c\rho \cdot \Gamma(2-\alpha)} [b_0 U_{k,n} - b_{k-1} U_{0,n} + \sum_{i=1}^{k-1} (b_i - b_{i-1}) U_{k-i,n}] \end{array} \right]^2 \tag{16}$$

$$L_{T_i} = \frac{1}{N_i} \sum_{n=0}^{N} \left(T_{0,n} - T_i\left((x)_n \right) \right)^2,\ L_{U_{ic}} = \frac{1}{N_i} \sum_{n=0}^{N} \left(U_{0,n} - U_i\left((x)_n \right) \right)^2, \tag{17}$$

$$L_{U_b} = \frac{1}{N_b} \times \left[\begin{array}{l} \sum_{k=1}^{K} \left(\frac{d}{h^{\gamma}} \upsilon_0^{\gamma} U_{k,1} + \frac{d\delta}{h^{\gamma}} \upsilon_0^{\gamma} T_{k,1} - \beta\left(U_p - U_{k,0} \right) \right)^2 + \\ + \sum_{k=1}^{K} \left(\frac{d}{h^{\gamma}} \sum_{i=0}^{N} \upsilon_i^{\gamma} U_{k,N+1-i} - \frac{d\delta}{h^{\gamma}} \sum_{i=0}^{N} \upsilon_i^{\gamma} T_{k,N+1-i} - \beta\left(U_p - U_{k,N} \right) \right)^2 \end{array} \right], \tag{18}$$

$$L_{T_b} = \frac{1}{N_b} \times \left[\begin{array}{l} \sum_{k=1}^{K} \left(\frac{\lambda}{h^{\gamma}} \upsilon_0^{\gamma} T_{k,1} - \sigma\beta_1 \left(U_{k,0} - U_p \right) - \alpha^* \left(T_{k,0} - t_c \right) \right)^2 + \\ + \sum_{k=1}^{K} \left(\frac{\lambda}{h^{\gamma}} \sum_{i=0}^{N} \upsilon_i^{\gamma} T_{k,N+1-i} - \sigma\beta\left(U_{k,N} - U_p \right) - \alpha^* \left(T_{k,N} - t_c \right) \right)^2 \end{array} \right], \tag{19}$$

The training process is organized so that the loss functions of the networks N_U and N_T are minimized through sequential optimization aimed at finding the optimal parameters θ^1, θ^2. The training is conducted in stages: first, the parameters of network N_U are optimized while keeping the parameters of network N_T fixed, and then vice versa—the parameters of network N_T are optimized with the parameters of N_U fixed. This cycle is repeated until the predefined number of iterations is completed or the desired level of accuracy is achieved [2].

The network parameters, denoted as θ^1, θ^2, are updated at each step using the gradient descent method according to the following formulas:

$$\begin{cases} \theta^1{}_{k+1} = \theta^1{}_k - \eta \nabla_{\theta^1} L_{N_U}\left(\theta^1{}_k \right) = \theta^1{}_k - \eta\left(\nabla_{\theta^1} L_{U_d}\left(\theta^1 \right) + \nabla_{\theta^1} L_{U_r}\left(\theta^1 \right) + \nabla_{\theta^1} L_{U_i}\left(\theta^1 \right) + \nabla_{\theta^1} L_{U_b}\left(\theta^1 \right) \right), \\ \theta^2{}_{k+1} = \theta^2{}_k - \eta \nabla_{\theta^2} L_{N_T}\left(\theta^2{}_k \right) = \theta^2{}_k - \eta\left(\nabla_{\theta^2} L_{T_d}\left(\theta^2 \right) + \nabla_{\theta^2} L_{T_r}\left(\theta^2 \right) + \nabla_{\theta^2} L_{T_i}\left(\theta^2 \right) + \nabla_{\theta^2} L_{T_b}\left(\theta^2 \right) \right), \end{cases} \tag{20}$$

where η is the learning rate, and $\nabla_{\theta^1} L_{U_d}, \nabla_{\theta^1} L_{U_r}, \nabla_{\theta^1} L_{U_i}, \nabla_{\theta^1} L_{U_b}, \nabla_{\theta^2} L_{T_d}, \nabla_{\theta^2} L_{T_r}, \nabla_{\theta^2} L_{T_i}, \nabla_{\theta^2} L_{T_b}$ represents the gradients of the loss functions with respect to the equations (1), (2), boundary condition losses (4)-(7), initial condition losses (3), and the loss between the training data L_{U_d}, L_{T_d} and the network predictions for N_U and N_T.

In standard physics-informed neural networks, the coefficients of different terms in the loss function are typically set to 1 or have fixed values. However, the gradient imbalance between these terms can cause the network to focus on specific aspects, such as boundary or initial conditions, training data, or governing equations, which increases the risk of getting stuck in local minima.

To reduce the gradient imbalance between different terms in the loss function (12), adaptive weight parameters $\mu_{T_d}, \mu_{U_d}, \mu_{U_r}, \mu_{T_r}, \mu_{U_b}, \mu_{U_i}, \mu_{T_b}, \mu_{T_i}$ are introduced for each term in the loss function. These parameters are defined as follows:

$$\begin{aligned}&\forall\varepsilon_1>0,\left|\left(\mu_{U_r}\nabla_{\theta^1}L_{U_r}\left(\theta^1\right)-\mu_{U_i}\nabla_{\theta^1}L_{U_i}\left(\theta^1\right)-\mu_{U_b}\nabla_{\theta^1}L_{U_b}\left(\theta^1\right)\right)-\mu_{U_d}\nabla_{\theta^1}L_{U_d}\left(\theta^1\right)\right|<\varepsilon_1,\\&\forall\varepsilon_2>0,\left|\left(\mu_{T_r}\nabla_{\theta^2}L_{T_r}\left(\theta^2\right)-\mu_{T_i}\nabla_{\theta^2}L_{U_i}\left(\theta^2\right)-\mu_{T_b}\nabla_{\theta^2}L_{T_b}\left(\theta^2\right)\right)-\mu_{T_d}\nabla_{\theta^2}L_{T_d}\left(\theta^2\right)\right|<\varepsilon_2,\end{aligned}\tag{21}$$

$$\begin{aligned}&\mu_{U_d}=\frac{\left|\overline{\nabla_{\theta^1}L_{U_r}\left(\theta^1\right)}\right|}{\left|\overline{\nabla_{\theta^1}L_{U_d}\left(\theta^1\right)}\right|},\ \mu_{U_r}=\frac{\left|\overline{\nabla_{\theta^1}L_{U_r}\left(\theta^1\right)}\right|}{\left|\overline{\nabla_{\theta^1}L_{U_r}\left(\theta^1\right)}\right|}=1,\ \mu_{U_i}=\frac{\left|\overline{\nabla_{\theta^1}L_{U_r}\left(\theta^1\right)}\right|}{\left|\overline{\nabla_{\theta^1}L_{U_i}\left(\theta^1\right)}\right|},\ \mu_{U_b}=\frac{\left|\overline{\nabla_{\theta^1}L_{U_r}\left(\theta^1\right)}\right|}{\left|\overline{\nabla_{\theta^1}L_{U_b}\left(\theta^1\right)}\right|},\\&\mu_{T_d}=\frac{\left|\overline{\nabla_{\theta^2}L_{T_r}\left(\theta^2\right)}\right|}{\left|\overline{\nabla_{\theta^2}L_{T_d}\left(\theta^2\right)}\right|},\ \mu_{T_r}=\frac{\left|\overline{\nabla_{\theta^2}L_{T_r}\left(\theta^2\right)}\right|}{\left|\overline{\nabla_{\theta^2}L_{T_r}\left(\theta^2\right)}\right|}=1,\ \mu_{T_i}=\frac{\left|\overline{\nabla_{\theta^2}L_{T_r}\left(\theta^2\right)}\right|}{\left|\overline{\nabla_{\theta^2}L_{T_i}\left(\theta^2\right)}\right|},\ \mu_{T_b}=\frac{\left|\overline{\nabla_{\theta^2}L_{T_r}\left(\theta^2\right)}\right|}{\left|\overline{\nabla_{\theta^2}L_{T_b}\left(\theta^2\right)}\right|},\end{aligned}\tag{22}$$

where $\left|\overline{\cdot}\right|$ is the average gradient value for the corresponding term in the loss function (12).

These coefficients help balance the importance of different parts of the loss function during training, contributing to a more stable optimization process.

It is worth noting that more efficient learning is achieved when μ is set

$$\begin{aligned}&\mu_{U_d}=\frac{\max\left|\nabla_{\theta^1}L_{U_r}\left(\theta^1\right)\right|}{\left|\overline{\nabla_{\theta^1}L_{U_d}\left(\theta^1\right)}\right|},\ \mu_{U_i}=\frac{\max\left|\nabla_{\theta^1}L_{U_r}\left(\theta^1\right)\right|}{\left|\overline{\nabla_{\theta^1}L_{U_i}\left(\theta^1\right)}\right|},\ \mu_{U_b}=\frac{\max\left|\nabla_{\theta^1}L_{U_r}\left(\theta^1\right)\right|}{\left|\overline{\nabla_{\theta^1}L_{U_b}\left(\theta^1\right)}\right|},\\&\mu_{T_d}=\frac{\max\left|\nabla_{\theta^2}L_{T_r}\left(\theta^2\right)\right|}{\left|\overline{\nabla_{\theta^2}L_{T_d}\left(\theta^2\right)}\right|},\ \mu_{T_i}=\frac{\max\left|\nabla_{\theta^2}L_{T_r}\left(\theta^2\right)\right|}{\left|\overline{\nabla_{\theta^2}L_{T_i}\left(\theta^2\right)}\right|},\ \mu_{T_b}=\frac{\max\left|\nabla_{\theta^2}L_{T_r}\left(\theta^2\right)\right|}{\left|\overline{\nabla_{\theta^2}L_{T_b}\left(\theta^2\right)}\right|},\end{aligned},\tag{23}$$

This approach allows the network to adjust the learning rate based on the magnitude of the gradients. By applying this strategy, a reduction in the loss values for the training data and boundary conditions is observed, which improves the model's convergence compared to using a fixed learning rate.

Numerical Examples and Results

In this section, a study is conducted that demonstrates how the proposed fractal neural method can be applied to numerically solve the mathematical model of heat and mass transfer processes (1)-(7) with fractional derivatives in time and space. To evaluate the method's performance, indicators such as the the relative L2-norm, defined as, were used.

$$\varepsilon_{rel}=\sqrt{\sum_{i=1}^{N_d}\frac{\left|y_L-y_i^*\right|^2}{\left|y_i^*\right|^2}}.,\tag{23}$$

where y_i^* represents the training data and y_L denotes the approximate (calculated) value.

The numerical experiments were conducted on a platform with an Intel(R) Core(TM) i7-8750H processor, 16 GB of RAM, and the Windows 10 operating system. The model implementation, consisting of two separate fully connected neural networks, was carried out using TensorFlow and Keras libraries with automatic differentiation support. The Adam optimization algorithm was used to minimize the loss function, and parameter initialization was performed using the Xavier method. The learning rate was set to 1×10^{-4} , with 8 hidden layers, 40 neurons in each layer, and 4000 iterations.

The input layer of the model processes the spatial and temporal coordinates, which serve as input parameters, while the output layer of each neural network generates a single result used for approximating the moisture or temperature functions.The experiments were conducted using a capillary-porous material—wood with a density of $\rho=450$ kg/m³. The model was set up with the following initial parameters: sample temperature $T_0=20^\circ C$, ambient temperature $t_c=80^\circ C$, initial moisture content $U_0=0.5$ (kg/kg), relative air moisture $\varphi = 65\%$, and drying agent velocity $v = 2$ (m/s). Values $\alpha = 0.9, \beta = 1.9, \gamma = 0.95$ for fractional parameters were used.

To evaluate the effectiveness of the proposed method, results were compared with other approaches. Each experiment was repeated 10 times, and the final evaluation was based on the average of the obtained errors. A description of the models used in the study is provided below:

- *fPINN (fixed)*. This model uses discretization of fractional derivatives to bypass the limitations of automatic differentiation. The loss function consists of four components with fixed weights for each of them. Fractional derivatives are discretized using schemes (10) and (11), and the tanh activation function is used.
- *fPINN (adaptive)*. This model implements a dynamic weight distribution between the components of the loss function based on gradient statistics, which reduces the imbalance between the different components. All layers of the network use the swish activation function. Other parameters remain the same as in the fPINN (fixed) model.

Graphical representations of the simulation results for temperature and moisture dynamics using the fractal neural network method are presented in Figure 2.

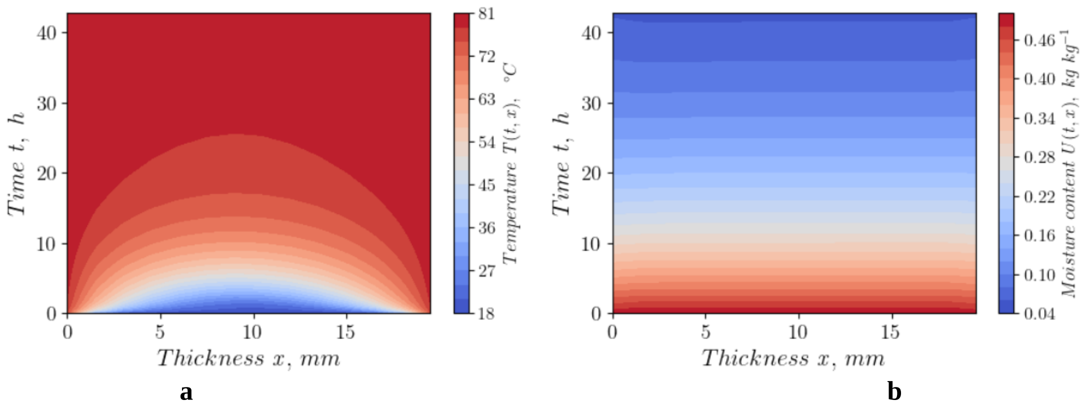


Fig. 2. The results obtained using the network model: (a) Variations in temperature fields; (b) Variations in moisture content.

In the course of the study, the fPINN (fixed) model was evaluated by dividing the data into three sets: training, validation, and test sets. The model was trained on the training set, with parameter optimization performed using the loss function (12). The validation set was used for hyperparameter tuning and overfitting control. After training was completed, the model was evaluated on the test set, which allowed for a final assessment of its generalization ability. The analysis of the loss functions, presented in Figure 3, helped determine the optimal number of epochs (≈ 4000), which contributed to avoiding overfitting.

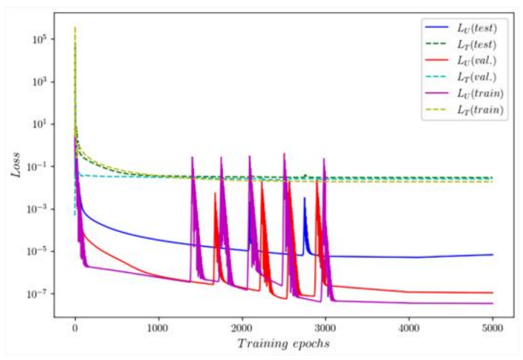
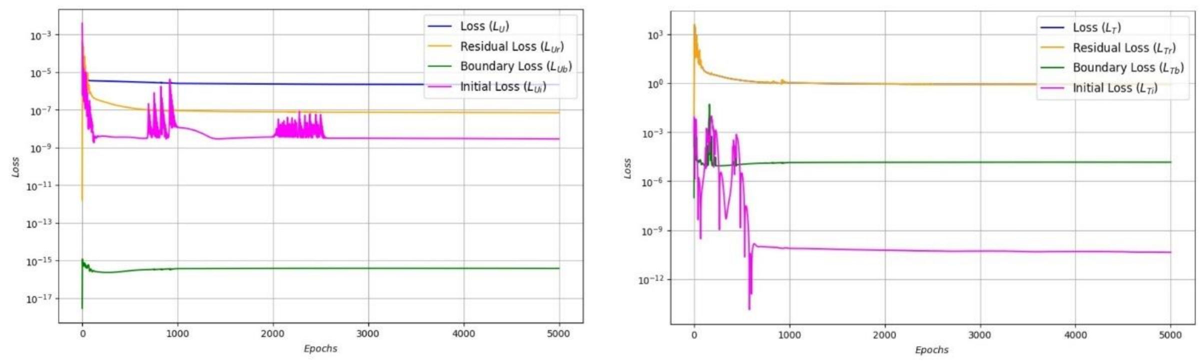


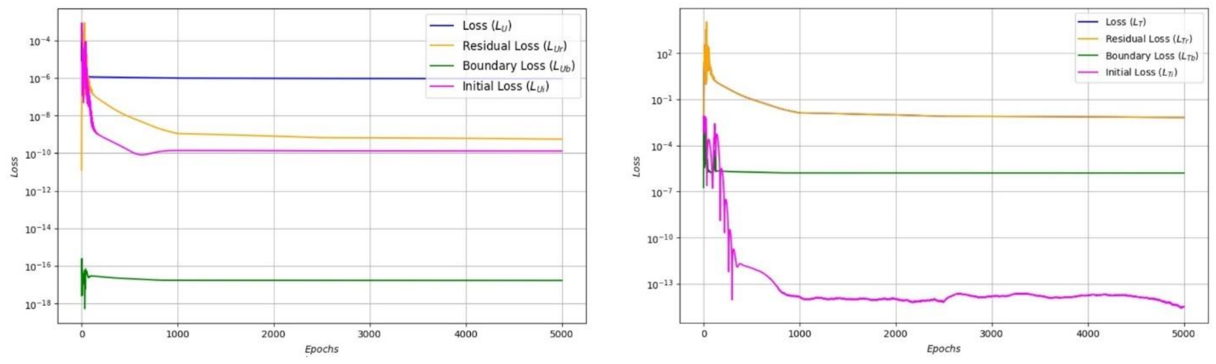
Fig. 3. Evolution of the loss functions for the training, validation, and test datasets

A study was also conducted to investigate the impact of regularization methods on the model. L1 and L2 regularization [29] were sequentially introduced into the training process. The results of the experiment are presented in Figures 4 and 5.



a **b**

Fig. 4. The effect of the L1 regularization method on the model: a) The effect of L1 regularization on the loss function L_{N_U} and its components in the fPINNs model with fixed weights; b) The effect of L1 regularization on the loss function L_{N_T} and its components in the fPINNs model with fixed weights.



a **b**

Fig. 5. The impact of the L2 regularization method on the model: a) The effect of L2 regularization on the loss function L_{N_U} and its components in the fPINNs model with fixed weights; b) The effect of L2 regularization on the loss function L_{N_T} and its components in the fPINNs model with fixed weights.

The application of L1 regularization leads to an increase in the L2 relative norm, as L1 regularization adds a penalty to the loss function for the absolute values of the network's weight coefficients. This can result in some increase in the variation of the coefficient magnitudes compared to L2 regularization. At the same time, L1 regularization has the property of creating sparse models, which allows ignoring less important features.

In contrast, L2 regularization stabilizes the training process by reducing the magnitude of the network's weight coefficients, helping to prevent overfitting and making the model more robust to noise in the data. Due to the penalty in the loss function for large coefficients, L2 regularization helps reduce model complexity and improves its generalization ability by decreasing the variability in model parameters.

L2 regularization demonstrates more stable loss reduction over time, while L1 regularization may exhibit a more erratic behavior during the initial iterations. L1 regularization promotes the "zeroing out" of model weights, which can explain the greater fluctuations during the early iterations. L2 regularization ensures faster convergence of the loss, particularly for the residual error, as evidenced by the smoother decrease in the graph.

When comparing the impact of L1 and L2 regularization on model accuracy, it is particularly important to consider stability and error levels. The graphs presented in Figure 6, which analyze the relative L2-norm of errors for the variables $U(t, \boldsymbol{x})$ and $T(t, \boldsymbol{x})$, provide a clearer understanding of this difference.

As seen in the first graph, L1 regularization has both strengths and weaknesses. When applied, the error ε_{rel} for the variable $U(t, \boldsymbol{x})$ remains stable at a relatively low level—around 0.05–0.075. The variable $T(t, \boldsymbol{x})$ exhibits sharp fluctuations, sometimes reaching quite high values (up to 0.225), indicating instability in the model. Only at the end of the iterations does this process stabilize, but at a relatively high level ($\varepsilon_{rel} \approx 0.125$). This behavior can be attributed to the nature of L1 regularization, which, through its tendency to "sparsify" the weights, may cause erratic optimization.

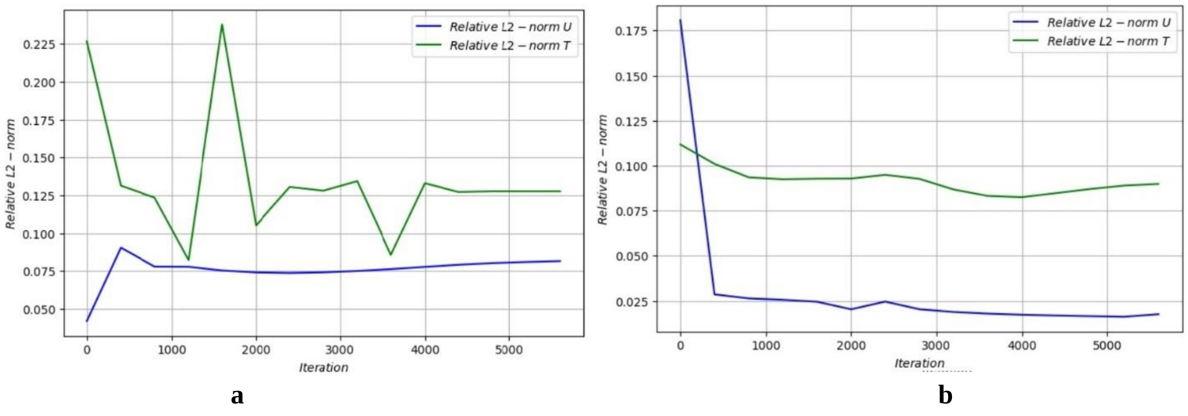


Fig. 6. Comparison of the relative L2 norm of errors for L1 and L2 regularizations: a) L1 Regularization b) L2 Regularization

On the other hand, L2 regularization has a more favorable effect on the model. As shown in graph b, the error $\varepsilon_{rel}$for the moisture content variable decreases rapidly to a low level ($\varepsilon_{rel} \approx 0.025$) within the first 1000 iterations and remains stable throughout the rest of the training. For the variable $T(t, \boldsymbol{x})$, a smooth decrease in the error ε_{rel} is observed, reaching a level of 0.075–0.1 without any spikes or instability. This indicates that L2 regularization better controls the optimization process and weight distribution.

The study involved analyzing the impact of different types of activation functions. Figure 7 illustrates the dynamics of the loss functions during the training of a neural network with various activation functions: ReLU, Tanh, Swish, and Sigmoid [30].

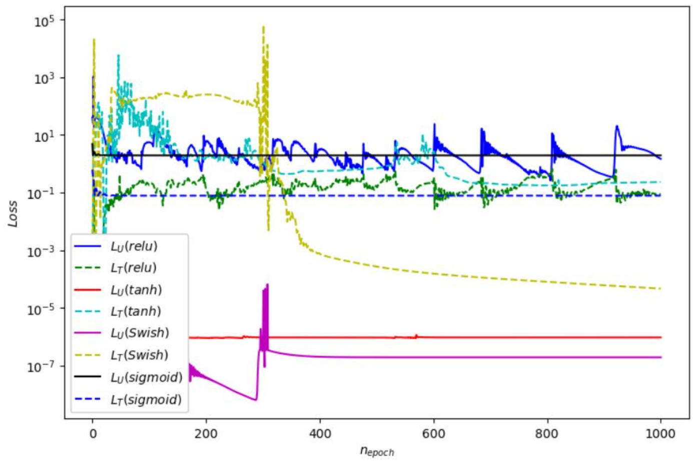
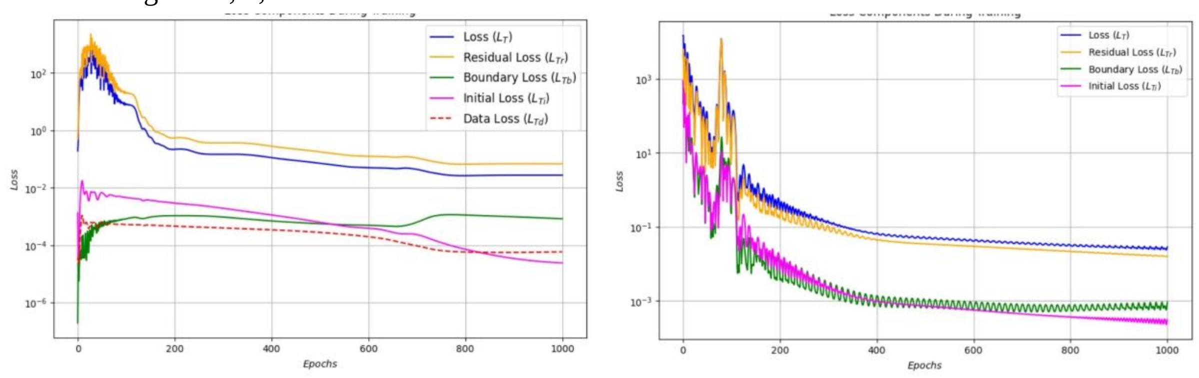


Fig . 7. Impact of various types of activation functions

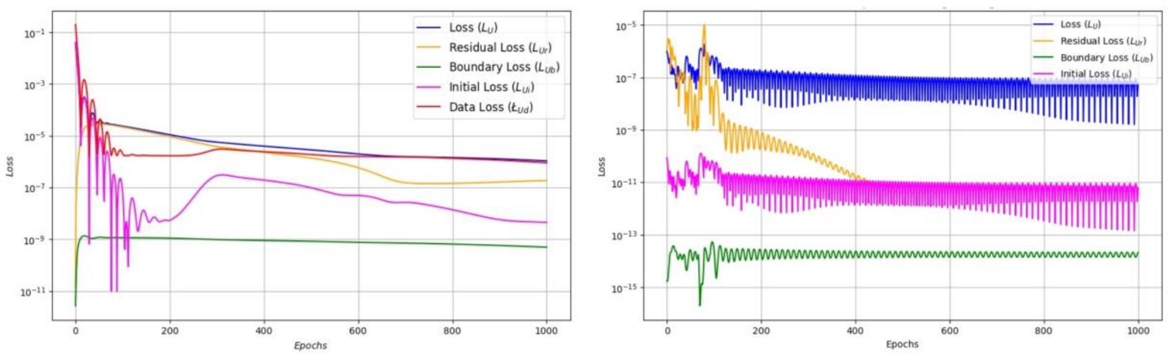
The model with the ReLU activation function exhibits unstable behavior, with the loss function values fluctuating even after many epochs of training. This may indicate difficulties with gradient stability. The Tanh activation function shows a more stable decrease in the loss functions. The training progresses smoothly, with the values of L_{N_U} and L_{N_T} decreasing to low levels, indicating balanced and stable performance for both variables of the system. The Tanh activation function is popular in neural networks for solving partial differential equations, and its effectiveness has been confirmed by numerous experiments. However, this study establishes that the Swish activation function delivers the best results. The loss functions L_{N_U} and L_{N_T} decrease rapidly and reach minimal values compared to the other activation functions. After approximately 200 epochs of training, the losses stabilize at a very low level, indicating effective optimization for the variables $U(t,\boldsymbol{x})$ and $T(t,\boldsymbol{x})$. The Sigmoid activation function is the least effective for this model, as it demonstrates a slow reduction in the loss.

The adaptive assignment of weights to the components of the loss function, based on gradient statistics, helps to reduce imbalances between different components, thereby facilitating more efficient neural network training. This approach ensures that the network pays equal attention to all loss function components during backpropagation, preventing the dominance of any single component. We integrated this concept of adaptive learning rate adjustment into the architecture of the fractal network. The results are presented in Figures 8, 9, and 10.



a **b**

Fig . 8. Effect of Adaptive Weights on Loss Function Convergence in fPINN fixed model: a) Without adaptive weights ; b) With adaptive weights



a **b**

Fig . 9. Effect of Adaptive Weights on Loss Function Convergence in fPINN fixed model: a) Without adaptive weights; b) With adaptive weights

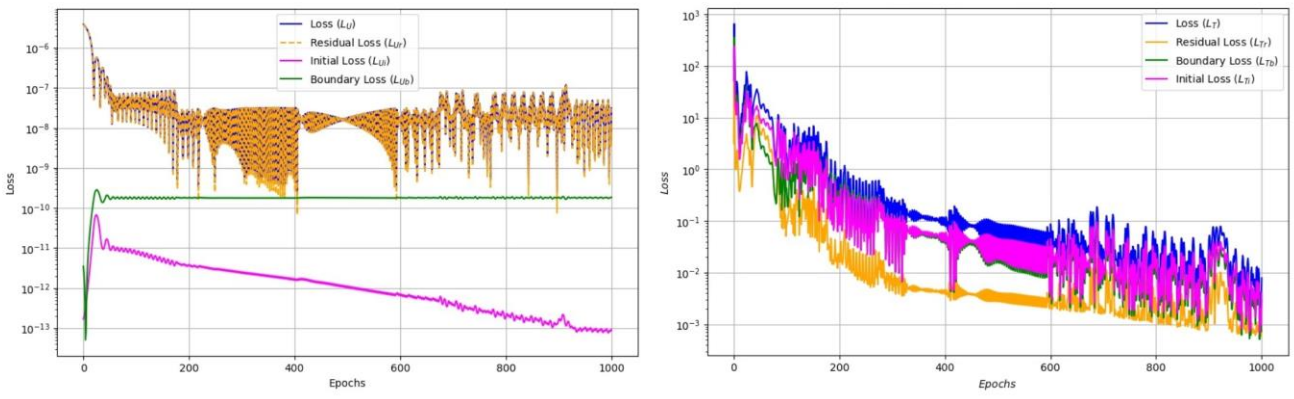
The presented graphs (Figure 8, 9) illustrate the changes in the total loss function and its components during model training.

In graph a of Figure 8, 9 (without adaptive weights), the total loss function (L_{N_U} and L_{N_T}) decreases gradually; however, the process is accompanied by significant oscillations, particularly during the initial stages of training. This indicates that the model struggles to effectively balance the various constraints of the problem, which slows down convergence.

In contrast, in graph b of Figure 8, 9 (with adaptive weights), the loss function decreases more steadily. The reduction in losses occurs faster, and the amplitude of oscillations significantly decreases, even in the early stages of training. This suggests that adaptive weights enable the model to better focus on critical aspects of the problem at each epoch.

The loss components also exhibit notable differences between the two approaches. Without adaptive weights (graph a), the components decrease unevenly—some converge slowly, while others remain nearly unchanged throughout training. This highlights the model's inability to account for the diversity of problem conditions effectively. However, with adaptive weights (graph b), the loss components exhibit consistent reductions. Adaptive weights allow the model to automatically adjust the importance of each component based on its contribution to the total loss. As a result, balance across all constraints is achieved, significantly improving the training process.

The graphs further show that training with adaptive weights is significantly more stable. Without adaptation, the loss functions exhibit persistent oscillations, even at later stages of training, indicating challenges for the model in achieving optimal solutions. Conversely, with adaptive weights, oscillations are markedly reduced, particularly after the first few hundred epochs, facilitating faster and more stable convergence.



a **b**

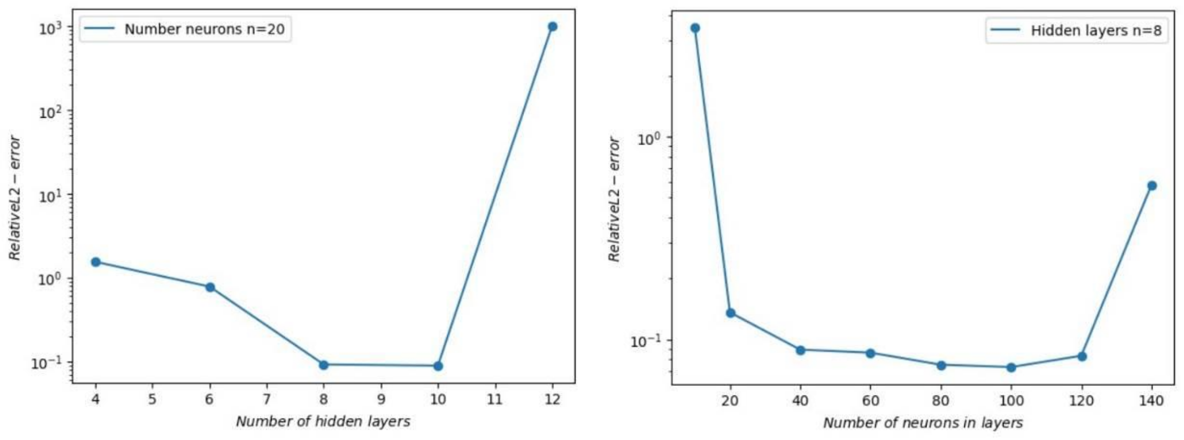
Fig . 10. Impact of adaptive weights on loss functions in fPINN adaptive model:

a) Impact on loss L_{N_U} ; b) Impact on loss L_{N_T}

The use of the Swish activation function combined with adaptive weights for the components of the loss function demonstrates a significant improvement in the model's training quality. As observed from the graph, the overall loss functions decrease steadily and rapidly, reaching values close to zero with minimal oscillations throughout the training process. The components of the loss function decrease synchronously, ensuring a balanced contribution of each to the model's training. This indicates that the use of Swish facilitates effective interaction between components, optimizing convergence. Such an approach can enable faster and more stable convergence compared to using adaptive weights alone, making it a promising strategy for application in fractal neural networks.

The analysis results demonstrate that the use of adaptive weights provides substantial advantages over the approach without their application.

A study was also conducted on the impact of the neural network architecture, specifically the number of hidden layers and neurons in each layer, on the model's accuracy, which is evaluated using the relative L2 norm ε_{rel}. The results of this study are presented in Figure 11.



a **b**

Fig . 11. Impact of network width and depth on model performance:
a) Impact of network depth; b) Impact of network width

Graph a in Figure 11 shows that with the increase in the number of hidden layers, there is initially a gradual decrease in the error, which reaches a minimum at 8-10 layers. However, beyond this point, there is a sharp increase in error, indicating the inefficiency of the network with further increases in depth. This suggests that the optimal number of hidden layers for this task is around 8, and adding extra layers does not improve but rather worsens the results.

Graph b in Figure 11 demonstrates that with the increase in the number of neurons in the layers, the relative L2 norm ε_{rel} decreases up to a certain point (approximately 100 neurons per layer), after which the accuracy sharply deteriorates. This phenomenon indicates the difficulty in optimizing the model with an excessive number of parameters. The optimal number of neurons in the layers for this model is around 100, which ensures the best accuracy. However, the training time for the model with 100 neurons significantly exceeds that of the model with 40 neurons in each layer. The gain in accuracy is marginal, which allows the best model parameters to be determined as 40 neurons and 8 hidden layers.

Conclusions

The research successfully applied a fractional neural network with an adaptive learning rate for modeling the dynamics of non-isothermal heat and mass transfer in capillary-porous materials. This approach accounted for memory effects and spatial nonlocality, which are crucial for accurately replicating physical processes in such environments.

A staged approach was used to optimize the model, reducing sensitivity to errors and failures, ensuring the stability of the training process and high accuracy of the results. Additionally, the neural network architecture was fine-tuned, and optimal activation functions and regularization methods were chosen, improving the modeling quality and ensuring the reliability of the obtained results.

Thus, the developed approach not only demonstrated its effectiveness but also opens new perspectives for further research and improvement of numerical modeling methods in the fields of heat and mass transfer, as well as in other areas of scientific investigation.

Thanks to the optimization of network parameters and the correct choice of activation functions and regularization, the model demonstrated high accuracy and stability. This confirmed the effectiveness of using fractional neural networks for complex physical models of heat and mass transfer in capillary-porous materials and opens new possibilities for further research and applications in materials science and engineering.

References

[1] Y. Sokolovskyy, T. Samotii, I. Kroshnyy, "Physics-Informed Neural Network for Modeling the Process of Heat-and-Mass Transfer Based on the Apparatus of Fractional Derivatives," Proceedings of the 2023 IEEE 17th

International Conference on the Experience of Designing and Application of CAD Systems (CADSM), Lviv, Ukraine, February 22–25, 2023, pp. 30–35.

[2] Y. Sokolovskyy, K. Drozd, T. Samotii, I. Boretska, "Fractional-Order Modeling of Heat and Moisture Transfer in Anisotropic Materials Using a Physics-Informed Neural Network," Materials, vol. 17, p. 4753, 2024. https://doi.org/10.3390/ma17194753.

[3] B. Golodenko, "Estimation of the adequacy of the fractal model of the atomic structure of amorphous silicon," Semiconductors, vol. 44, pp. 84–88, 2010. https://doi.org/10.1134/S1063782610010148.

[4] Y. Sokolovskyy, I. Boretska, B. Gayvas, I. Kroshnyy, V. Shymanskyi, M. Gregus, "Mathematical modeling of heat transfer in anisotropic biophysical materials, taking into account the phase transition boundary," Proceedings of the 2nd International Workshop on Informatics and Data-Driven Medicine, IDDM–CEUR, vol. 2488, pp. 121–132, 2019.

[5] S. Aman, Q. Al-Mdallal, I. Khan, "Heat transfer and second order slip effect on MHD flow of fractional Maxwell fluid in a porous medium," Journal of King Saud University - Science, vol. 32, pp. 450–458, 2020.

[6] Z.-Y. Li, H. Liu, X.-P. Zhao, W.-Q. Tao, "A multi-level fractal model for the effective thermal conductivity of silica aerogel," Journal of Non-Crystalline Solids, vol. 430, pp. 43–51, 2015. https://doi.org/10.1016/j.jnoncrysol.2015.09.023.

[7] Y. Sokolovskyy, M. Levkovych, O. Mokrytska, Y. Kaspryshyn, "Mathematical Modeling of Nonequilibrium Physical Processes, Taking into Account the Memory Effects and Spatial Correlation," Proceedings of the 9th International Conference on Advanced Computer Information Technologies, ACIT 2019, 2019, pp. 56–59. doi:10.1109/ACITT.2019.8780011.

[8] Z.-H. Wu, A. Debbouche, J. Guirao, X.-J. Yang, "On local fractional Volterra integral equations in fractal heat transfer," Thermal Science, vol. 20, pp. 202–202, 2016. https://doi.org/10.2298/TSCI151217202W.

[9] Yan S., "Local fractional Laplace series expansion method for diffusion equation arising in fractal heat transfer," Thermal Science, vol. 19, no. suppl. 1, pp. 131–135, 2015.

[10] X.-J. Yang, F. R. Zhang, "Local Fractional Variational Iteration Method and Its Algorithms," Adv. Comput. Math. Appl., vol. 1, no. 3, pp. 139–145, 2012.

[11] Y. Sokolovskyy, V. Shymanskyi, M. Levkovych, V. Yarkun, "Mathematical modeling of heat and moisture transfer and rheological behavior in materials with fractal structure using the parallelization of predictor-corrector numerical method," Proceedings of the 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv, 2016, pp. 108–111.

[12] M. M. Meerschaert, C. Tadjeran, "Finite difference approximations for fractional advection–dispersion flow equations," Journal of Computational and Applied Mathematics, vol. 172, pp. 65–77, 2004.

[13] E. N. Akimova, M. A. Sultanov, V. E. Misilov, Y. Nurlanuly, "Parallel Algorithm for Solving the Inverse Two-Dimensional Fractional Diffusion Problem of Identifying the Source Term," Fractal Fract., vol. 7, p. 801, 2023. https://doi.org/10.3390/fractalfract7110801.

[14] A. Lagaris, A. Likas, D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," IEEE Transactions on Neural Networks, vol. 9, pp. 987–1000, 1998.

[15] T. Chen, H. Chen, "Approximations of continuous functionals by neural networks with application to dynamic systems," IEEE Transactions on Neural Networks, vol. 4, no. 6, pp. 910–918, 1993.

[16] T. Chen, H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," IEEE Transactions on Neural Networks, vol. 6, no. 4, pp. 911–917, 1995.

[17] M. Raissi, P. Perdikaris, G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," Journal of Computational Physics, vol. 378, pp. 686–707, 2019. https://doi.org/10.1016/j.jcp.2018.10.045.

[18] D. Zhang, L. Lu, L. Guo, G. E. Karniadakis, "Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems," Journal of Computational Physics, vol. 397, pp. 1–19, 2019.

[19] D. Jagtap, K. Kawaguchi, G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," Journal of Computational Physics, vol. 404, pp. 1–23, 2020.

[20] L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," SIAM Review, vol. 63, no. 1, pp. 208–228, 2021.

[21] D. Jagtap, E. Kharazmi, G. E. Karniadakis, "Conservative physics-informed neural networks on discrete

domains for conservation laws: Applications to forward and inverse problems," Comput. Methods Appl. Mech. Eng., vol. 365, pp. 113028, 2020.

[22] G. Pang, M. D'Elia, M. Parks, G. E. Karniadakis, "nPINNs: Nonlocal physics-informed neural networks for a parametrized nonlocal universal Laplacian operator," Journal of Computational Physics, vol. 422, pp. 109760, 2020.

[23] M. Lahariya, F. Karami, C. Develder, G. Crevecoeur, "Physics-informed Recurrent Neural Networks for The Identification of a Generic Energy Buffer System," Proceedings of the 2021 IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS), Suzhou, China, 2021, pp. 1044–104.

[24] Z. Fang, "A High-Efficient Hybrid Physics-Informed Neural Networks Based on Convolutional Neural Network," IEEE Transactions on Neural Networks and Learning Systems, vol. 33, pp. 5514–5526, 2021.

[25] G. Pang, L. Lu, G. E. Karniadakis, "fPINNs: Fractional physics-informed neural networks," SIAM Journal on Scientific Computing, vol. 41, pp. A2603–A2626, 2019.

[26] F. Rostami, A. Jafarian, "A new artificial neural network structure for solving high-order linear fractional differential equations," International Journal of Computational Mathematics, vol. 95, pp. 528–539, 2018. https://doi.org/10.1080/00207160.2017.1291932.

[27] S. Wang, H. Zhang, X. Jiang, "Fractional physics-informed neural networks for time-fractional phase field models," Nonlinear Dynamics, vol. 110, no. 4, pp. 2715–2739, 2022. https://doi.org/10.1007/s11071-022-07746-3.

[28] A. Atangana, A. Secer, "A note on fractional order derivatives and table of fractional derivatives of some special functions," Abstract and Applied Analysis, vol. 2013, Article ID 279681, 2013.

[29] V. V. Uchaikin, *Method drobovykh pokhidnykh* [*Method of Fractional Derivatives*], Ulyanovsk: Artishok Publ., Russia, 2008.

[30] A. V. Morozov, V. L. Levkivskyi, D. D. Plechystyy, "Activation functions in neural networks: Overview and comparison," Scientific Notes of V.I. Vernadsky Taurida National University. Series: Technical Sciences, vol. 35 (74), no. 3, 2024. https://doi.org/10.32782/2663-5941/2024.3.1/22.

Ярослав Соколовський [1], Тетяна Самотій[2]

[1] Кафедра систем автоматизованого проектування, Національний університет Львівська політехніка, Україна, Львів, вул.С.Бандери 12, E-mail: yaroslav.i.sokolovskyi@lpnu.ua, ORCID 0000-0003-4866-2575

[2] Кафедра інженерії програмного забезпечення, Національний лісотехнічний університет України, Україна, Львів, вул. Г.Чупринки 103, E-mail: t.samotiy@nltu.edu.ua, ORCID 0000-0002-1317-8786

АДАПТИВНИЙ ФРАКЦІЙНИЙ НЕЙРОННИЙ АЛГОРИТМ ДЛЯ МОДЕЛЮВАННЯ ТЕПЛОВОЛОГОПЕРЕНЕСЕННЯ

Отримано: Листопад 12, 2024 / Переглянуто: Листопад 20, 2024 / Прийнято: Листопад 25, 2024

Анотація. Запропоновано фракційну нейронну мережу з адаптивним темпом навчання для моделювання динаміки неізотермічного тепло- та масоперенесення в капілярно-пористих матеріалах з урахуванням ефекту пам'яті та просторової нелокальності. Використано архітектуру нейронної мережі з роз'єднаною структурою, яка базується на функціях втрат, що враховують фізичні особливості досліджуваного процесу. Для навчання мережі використано поетапний підхід, що дозволяє зменшити чутливість до помилок та збоїв. Досліджено структуру мережі, оптимізовано її параметри, а також обрано відповідні активаційні функції та методи регуляризації з метою досягнення високої точності та достовірності результатів моделювання.

Ключові слова: дробові похідні, машинне навчання у фізичних процесах, моделювання нейронними мережами, адаптивний темп навчання, капілярно-пористі матеріали