# Two-dimensional Klein-Gordon and Sine-Gordon numerical solutions based on deep neural network

**Soumaya Nouna[1], Assia Nouna[1], Mohamed Mansouri[1], Ilyas Tammouch[2], Boujamaa Achchab[1]**
[1]Laboratory LAMSAD, ENSA Berrechid, Hassan First University, Settat, Morocco
[2]Laboratory of Telecommunications Systems and Decision Engineering, Faculty of Science, Ibn Tofail University, Kenitra, Morocco

**Article Info**

***Article history:***

Received Feb 25, 2024
Revised Oct 30, 2024
Accepted Nov 14, 2024

***Keywords:***

Deep learning
Feedforward neural network
Nonlinear Klein-Gordon equations
Partial differential equations
Sine-Gordon equations

**ABSTRACT**

Due to the well-known dimensionality curse, developing effective numerical techniques to resolve partial differential equations proved a complex problem. We propose a deep learning technique for solving these problems. Feedforward neural networks (FNNs) use to approximate a partial differential equation with more robust and weaker boundaries and initial conditions. The framework called PyDEns could handle calculation fields that are not regular. Numerical experiments on two-dimensional Sine-Gordon and Klein-Gordon systems show the provided frameworks to be sufficiently accurate.

***Corresponding Author:***

Soumaya Nouna
Laboratory LAMSAD, ENSA Berrechid, Hassan First University
Km. 10 Idiroko Road, Canaan Land, Ota, Ogun Settat, Morocco
Email: s.nouna@uhp.ac.ma

## 1. INTRODUCTION

Partial differential equations (PDEs) can be used to simulate nonlinear processes seen in many scientific domains, including plasma physics, the physics of solids, chemical kinetics, the dynamics of fluids, hydrodynamics, and the biology of mathematics. Many numerical and analytical solutions have been employed to solve these issues. Various studies of phenomena in physics across multiple domains of engineering and science have recently been conducted [1], [2]. This research focuses on the Klein-Gordon and Sine-Gordon systems, which are significant families of PDEs used to explain many diverse physical phenomena.

The Sine-Gordon system [3] was developed in the 19th century during research into various differential geometry issues. It is an intermediate hyperbolic differential equation widely applied in describing and modeling physical phenomena in various engineering and scientific fields, including fluxions propagation, nonlinear waves, and metal dislocation. This equation has garnered significant interest due to its numerous soliton solutions [4]. Similarly, the Klein-Gordon equation [5] has also received considerable attention for studying solitons, soliton interactions in collisionless plasmas, and nonlinear wave systems [6]. The Klein-Gordon equation represents a motion equation for a scalar or pseudoscalar field, significant for several scientific applications, including nonlinear optics theory and solid-state physics.

Various numerical approaches have been proposed to solve Sine-Gordon and Klein-Gordon systems, such as the exp-function model [7], the adomian decomposition method [8], [9], the reduced differential transform approach [10], [11], the variational iteration approach [12], [13], the homotopy perturbation approach [14], and the homotopy analysis approach [15], [16]. These traditional numerical approaches use a discrete domain over several finite elements or points, requiring finer meshes for better solutions. This strategy's computing cost can be greatly increased, hindering its use for highly-dimensional systems. To address these challenges, many deep learning methods have been proposed for solving high-dimensional problems.

In machine learning, deep learning methods use multiple hidden layers in neural networks. The benefit of adding hidden layers within the neural network process is that, for certain implementations, computing costs are reduced exponentially, as is the amount of data required for training [17]. Consequently, these networks can, in principle, find any solutions to differential equations within low error boundaries. Extensive research has been conducted on applying neural networks to solve differential equations. For example, the research in [18], [19] used artificial neural networks and unsupervised neural networks for resolving differential systems. Other works have also applied deep learning methods or deep neural networks (DNNs) to solve PDEs [20]–[22], demonstrating that deep learning-based approaches effectively solve mathematical models, including high-dimensional PDEs.

This paper uses a deep learning technique to resolve the 2-dimensional Sine-Gordon and Klein-Gordon equations. This research is partly motivated by the concepts presented in [23], where feedforward neural networks (FNNs) are employed to solve partial differential systems under both boundary and initial conditions. The rest of this paper is structured as follows: section 2 discusses the theory behind FNNs, section 3 introduces our approach to solving PDEs using these networks, and section 4 presents numerical simulations. Finally, section 5 concludes the article.

## 2. DEEP LEARNING: FEEDFORWARD NEURAL NETWORK

FNNs [24] are the most basic form of artificial neural network and have many applications in machine learning. FNNs are also called multilayer neuron networks (MLNs). These model networks are known as feedforward the information in the neural network propagates only forward, first via the input nodes, then through the single or many hidden layers, and ultimately through the output nodes. FNNs appear to be a potential method for solving PDEs. Figure 1 shows an example of an FNN with two hidden layers.

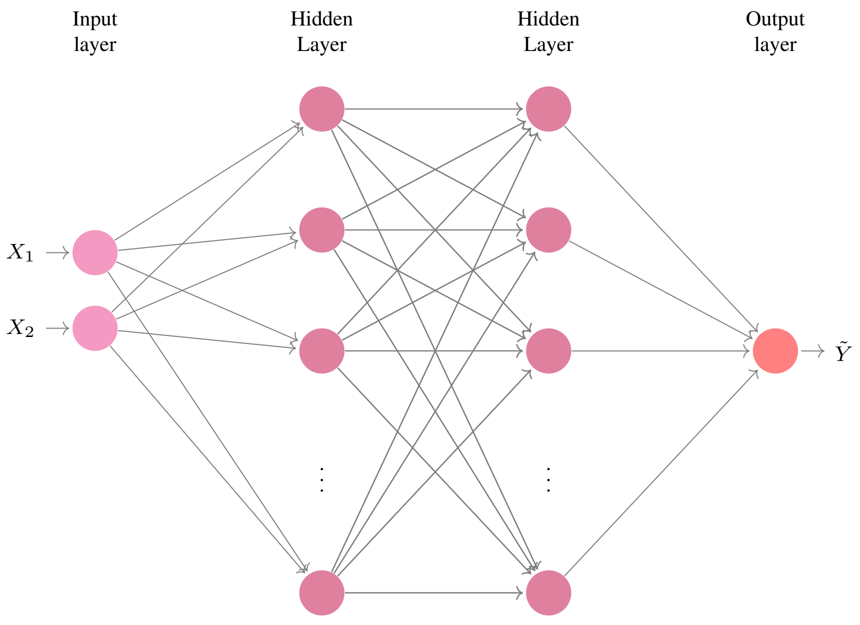


Figure 1. FNN with two hidden layers

Assume an FNN consisting of $(M + 1)$ layers, wherein 0 represents an entry layer, and $M$ represents an exit layer. Hidden layers consist of the intermediary layers between an input and output layer. $R_m$ neurons are found in the $m$-th layer, $m = 1, 2, ..., M$. Specifically, $R_M = 1$, indicating that the exit layer contains just a single neuron. Also, every node of the current layer $m$ takes as input each output of layer $m - 1$ and passes its proper output onto every node of the following layer. A weighted entry at every node $i$, $i = 1, 2, ..., R_m$, in layer $m$, for $m = 1, 2, ..., M$, is provided by the recursive formula:

$$k_i^m = \sum_{s=1}^{R_{m-1}} w_{is}^m \varphi_{m-1}(k_s^{m-1}) + b_i^m \tag{1}$$

Where $\varphi_m$ represents the activation function of layer $m$, $b_i^m$ denotes a bias of node $i$ in layer $m$ and a weight between node $s$ in layer $m-1$ and node $i$ in layer $m$ is $w_{is}^m$. For clarity, we replace $R_0$ with $r$. If the vector input is $X = (X_1, X_2, ..., X_r) \in \mathbb{R}^r$, so:

$$\varphi_0(k_s^0) = X_s, \qquad s = 1, 2, ..., r \tag{2}$$

Therefore, a neural network's output is provided by:

$$\tilde{Y} = N(X, \mathbf{q}) = k^M(X, \mathbf{q}) \tag{3}$$

Where $\mathbf{q} \in \mathbb{R}^{r_q}$ is a vector of all tunable parameters $w_{is}^m$ and $b_i^m$, where $r_q$ represents the number of total parameters, and $k^M$ comes from the (1). We assumed in (1) a unique activating functional utilized inside $m$-th layer with $m = 1, 2, ...M-1$.

Giving an entry or a collection of entries, a node's activation function defines the node's output. A neural network would be a linear transformation without the activation function. Also, any neural network having one hidden layer with a sufficient number of neurons can approximate every function continuously according to Theorem 2..1. There are many types of activation functions [25], such as the Relu function, which is defined as follows:

$$\varphi(X) = max\ (0, X) \tag{4}$$

The Relu activation can also be employed for second or higher-order PDEs. The purpose of FNN training is to minimize a loss function. Lesser the loss, the better the match between the analytical and predicted solution. We use optimizer algorithms to determine the best weights and collection biases to minimize the loss function. There are various optimization approaches, including Broyden-Fletcher-Goldfarb-Shanno (BFGS) [26], Levenberg-Marquardt [27], limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) [28], adaptive moment estimation (ADAM) [29], and stochastic gradient descent (SGD) [30].

**Theorem 2..1 (Theorem of universal approximation)** *The neural networks having the Relu activation function are the universal approximators. Note that universal approximators are defined in the Definition 2..1.*

**Definition 2..1** *Suppose $\Omega$ is a topology space with $\varphi : \mathbb{R} \to \mathbb{R}$. We state by saying that any neural network having activating functions $\varphi$ would be a universal approximator on $\Omega$ only if $\sum_n(\varphi)$ would be dense on $C(\Omega)$. Where $C(\Omega)$ is all the continuous functions from $\Omega$ through to $\mathbb{R}$ and the collection $\sum_n$ is all tasks that may be computed using a neural network with one hidden layer with activating functions $\varphi$.*

## 3. DESCRIPTION OF THE APPROACH

This study uses the FNN method to solve PDEs. The approach includes reformulating the problem into a loss function incorporating the initial and boundary conditions. This loss function is then minimized using advanced optimization techniques. FNNs are particularly well suited to this task because of their universal approximation capability [31], as demonstrated in the Theorem 3..1. This property allows neural networks to accurately approximate any continuous function, which is particularly advantageous for solving complex PDEs without resorting to traditional discretization methods. This section describes in detail the experimental procedure, the neural network configurations, the validation criteria, and the optimization techniques employed.

Assume that the first non-linear system is described as follows:

$$\begin{aligned}\frac{\partial^2 v(x,y,t)}{\partial t^2} + \lambda \frac{\partial v(x,y,t)}{\partial t} &= \beta \left( \frac{\partial^2 v(x,y,t)}{\partial x^2} + \frac{\partial^2 v(x,y,t)}{\partial y^2} \right) - f(x,y)\sin(v(x,y,t)) \\ &+ \ g(x,y,t), \qquad\qquad\qquad (x,y) \in \Omega_1, t \geq 0\end{aligned} \tag{5}$$

According to the next initial conditions:

$$F(x,y)\begin{cases} v(x,y,0) &= \phi_1(x,y), \qquad (x,y)\in\Omega_1 \\ \dfrac{\partial v}{\partial t}(x,y,0) &= \phi_2(x,y), \qquad (x,y)\in\Omega_1 \end{cases} \tag{6}$$

And the second non-linear system is described as:

$$\begin{aligned} &\frac{\partial^2 v(x,y,t)}{\partial t^2}+\alpha\frac{\partial v(x,y,t)}{\partial t}=\gamma\left(\frac{\partial^2 v(x,y,t)}{\partial x^2}+\frac{\partial^2 v(x,y,t)}{\partial y^2}\right) \\ &-\Big(\mu_1 v(x,y,t)+\mu_2 v^n(x,y,t)\Big) \qquad + \ h(x,y,t), \qquad (x,y)\in\Omega_2, t>0 \end{aligned} \tag{7}$$

Depending on the initial conditions:

$$G(x,y)\begin{cases} v(x,y,0) &= \psi_1(x,y), \qquad (x,y)\in\Omega_2 \\ \dfrac{\partial v}{\partial t}(x,y,0) &= \psi_2(x,y), \qquad (x,y)\in\Omega_2 \end{cases} \tag{8}$$

Where (5) is known as a nonlinear Sine-Gordon system in two dimensions, with $\Omega_1 \in \mathbb{R}^2$, $f(x,y)$ may be thought of as just a Josephson density, with $\phi_1(x,y)$, and $\phi_2(x,y)$ representing wave patterns as well as curves and acceleration, correspondingly. In (7) is named the two-dimensional Klein-Gordon equation which has a fourfold non-linearity when $n=2$ and a cubical non-linearity when $n=3$, $\Omega_2\in\mathbb{R}^2$, and $h(x,y,t)$ seems to be a known function, $\psi_1(x,y)$ and $\psi_2(x,y)$ are starting functions. Parameters $\lambda,\beta,\alpha,\mu_1,\mu_2$ and $\gamma$ are constants.

We applied the FNN technique to find a functional $v_{FNN}$ which approaches the real solution $v$. This method minimizes the loss functions associated with the two systems, defined as follows:

$$\begin{aligned} \hat{\mathbf{J}}(\mathbf{q}) = \Bigg\| &\frac{\partial^2 v_{FNN}(x,y,t;\mathbf{q})}{\partial t^2}+\lambda\frac{\partial v_{FNN}(x,y,t;\mathbf{q})}{\partial t} \\ &-\beta\left(\frac{\partial^2 v_{FNN}(x,y,t;\mathbf{q})}{\partial x^2}+\frac{\partial^2 v_{FNN}(x,y,t;\mathbf{q})}{\partial y^2}\right) \\ &+f(x,y)\sin(v_{FNN}(x,y,t;\mathbf{q}))-g(x,y,t)\Bigg\|^2_{(x,y)\in\Omega_1,t\geq 0,\xi_1} \\ &+\Big\|v_{FNN}(x,y,0;\mathbf{q})-F(x,y)\Big\|^2_{(x,y)\in\Omega_1,\xi_2} \end{aligned} \tag{9}$$

$$\begin{aligned} \hat{\mathbf{L}}(\mathbf{q}) = \Bigg\| &\frac{\partial^2 v_{FNN}(x,y,t;\mathbf{q})}{\partial t^2}+\alpha\frac{\partial v_{FNN}(x,y,t;\mathbf{q})}{\partial t} \\ &-\gamma\left(\frac{\partial^2 v_{FNN}(x,y,t;\mathbf{q})}{\partial x^2}+\frac{\partial^2 v_{FNN}(x,y,t;\mathbf{q})}{\partial y^2}\right) \\ &+\Big(\mu_1 v_{FNN}(x,y,t;\mathbf{q})-\mu_2 v^n_{FNN}(x,y,t;\mathbf{q})\Big) \\ &-h(x,y,t)\Bigg\|^2_{(x,y)\in\Omega_2,t>0,\xi_1} + \Big\|v_{FNN}(x,y,0;\mathbf{q})-G(x,y)\Big\|^2_{(x,y)\in\Omega_2,\xi_2} \end{aligned} \tag{10}$$

In (9) represents the loss function of (5) and (6), whereas in (10) represents the loss function of (7) and (8). $\xi_1,\xi_2$ design the measurements. $\mathbf{q}$ represents a neural network's vector of parameters weights and biases. The approximate solution $v_{FNN}$, used in the equations and loss functions, is generated by the output of the FNN described in the previous section (see in (3)) and is expressed as (11):

$$v_{FNN}=k^M(x,y,t;\mathbf{q}). \tag{11}$$

Our approach uses FNNs to approximate the solutions of Sine-Gordon and Klein-Gordon PDEs. First, we transform these problems into loss functions incorporating initial and boundary conditions, which allows optimization techniques to be used to train the neural networks. The backpropagation method, detailed in [32], is used to determine the derivatives of the function $v_{FNN}$ concerning the network inputs. TensorFlow [33] offers an automatic differentiation package in inverse mode, capable of calculating each differential of the loss function and producing the necessary gradients.

Next, we apply SGD to minimize the loss functions $\hat{\mathbf{J}}$ and $\hat{\mathbf{L}}$, thereby optimizing the neural network parameters. The Algorithms 1 and 2 describe in detail the specific steps of this procedure for the Sine-Gordon and Klein-Gordon equations respectively, taking into account the particularities of each equation.

**Algorithm 1** The deep neural network algorithm for nonlinear Sine-Gordon equation

```
1. Determine the FNN model (layers, neurons, and the activation function)
2. Initializing FNN parametric $\mathbf{q}^0$
3. Process FNN $(\xi_1, \xi_2)$
4.     Repeating
5.         Generating $c_1$ and $c_2$ points from $\Omega_1 \times [0, \mathbf{T}]$ and $\Omega_1 \times \{0\}$. Get distributed $\xi_1, \xi_2$.
6.         Utilizing sampling lots, approximate the loss (9):
```

$$\begin{aligned}\hat{\mathbf{J}}(\mathbf{q}) = \sum_{(x_j,y_j,t_j)\in c_1} \Bigg( & \frac{\partial^2 v_{FNN}(x_j,y_j,t_j;\mathbf{q})}{\partial t^2} + \lambda \frac{\partial v_{FNN}(x_j,y_j,t_j;\mathbf{q})}{\partial t} \\ & - \beta \left( \frac{\partial^2 v_{FNN}(x_j,y_j,t_j;\mathbf{q})}{\partial x^2} + \frac{\partial^2 v_{FNN}(x_j,y_j,t_j;\mathbf{q})}{\partial y^2} \right) \\ & + f(x_j,y_j)\sin(v_{FNN}(x_j,y_j,t_j;\mathbf{q})) - g(x_j,y_j,t_j) \Bigg)^2 \\ & + \sum_{(x_j,y_j,0)\in c_2} \Bigg( v_{FNN}(x_j,y_j,0;\mathbf{q}) - F(x_j,y_j) \Bigg)^2 \end{aligned}$$

```
7.         Consider taking a step of descending gradient:
```

$$\mathbf{q}_{m+1} = \mathbf{q}_m - \delta_m \nabla_{\mathbf{q}} \hat{\mathbf{J}}(\mathbf{q}_m)$$

```
8.     Until its convergence.
9. Final Process
```

**Theorem 3..1** *Consider $f_N : [0,1]^r \to [0,1]$ as the function computed using a single-layer Relu neural network with r inputs, one output, and s Hidden Nodes. Thus, there also existed an alternative Relu neural network, having $m$ Hidden Layers with a width of r $+ 2$, that computes the identical function $f_N$.*

The theorem's proof utilizes an approach that uses r nodes of every layer for storing the first input, one for computing each node of an originally constructed network, and the final node for storing the result of summing functions.

*Proof.* As $f_N$ can be produced using a single-layer Relu network having $r$ hidden nodes, it takes the following expression:

$$f_N(X) = \varphi(\sum_{j=1}^{m} w_j \varphi(l_j(X)) + b)$$

Wherein every $l_j$ has a format of $l_j = Y_j \cdot X + b_j$ where $Y_j \in \mathbb{R}^r$, $b_j \in \mathbb{R}$ and $\varphi$ is the Relu activation function. As $[0,1]^r$ is a compact function, and Relu network-generated function is continuous. The term $\sum_{i=1}^{j} w_i \varphi(l_i(X))$ is at a minimum for every $j \leqslant m$, so there is a number $\mathcal{S} > 0$ like that:

$$\mathcal{S}+\sum_{i=1}^{j} w_i\varphi(l_i(X)) > 0$$

For all $X \in [0,1]^r$ as well as all $j \leqslant m$.

We now have a novel network where every hidden layer utilizes $r$ nodes for copying the originally entered. Furthermore, every $j$-th hidden layer also has an additional node which calculates function $\varphi(l_j(X))$ based on the input nodes that were replicated in the preceding layer, plus a final node which calculates:

$$\varphi(\mathcal{S}+\sum_{i=1}^{j-1} w_i\varphi(l_i(X))) = \mathcal{S}+\sum_{i=1}^{j-1} w_i\varphi(l_i(X))$$

By making the linear composition of both supplementary nodes from the preceding layer. Briefly, the $j$-th layer until $j \leqslant m$ shall calculate the following function:

$$\mathcal{H}^j(X) = (X_1, ..., X_r, \varphi(l_j(X)), \varphi(\mathcal{S}+\sum_{i=1}^{j-1} w_i\varphi(l_i(X)))).$$

Finally, the last layer calculates:

$$\begin{aligned}\mathcal{H}^{m+1}(X) &= \varphi\big[w_m\varphi(l_m(X)) + \varphi(\mathcal{S}+\textstyle\sum_{i=1}^{m-1} w_i\varphi(l_i(X)) - \mathcal{S} + b)\big]\\ &= \varphi(b+\textstyle\sum_{j=1}^{m} w_j\varphi(l_j(X)))\\ &= f_N(X)\end{aligned}$$

That finishes this proof with the use of $m$ hidden layers, every one of width $r+2$.

**Algorithm 2** The deep neural network algorithm for nonlinear Klein-Gordon equation

```
1. Determine the FNN model (layers, neurons, and the activation function)
2. Initializing FNN parametric $\mathbf{q}^0$
3. Process FNN ($\xi_1, \xi_2$)
4.     Repeating
5.         Generating $d_1$ and $d_2$ points from $\Omega_2 \times [0, \mathbf{T}]$ and $\Omega_2 \times \{0\}$. Get distributed $\xi_1, \xi_2$.
6.         Utilizing sampling lots, approximate the loss (10):
```

$$\begin{aligned}\hat{\mathbf{L}}(\mathbf{q}) = &\sum_{(x_j,y_j,t_j)\in d_1}\Bigg(\frac{\partial^2 v_{FNN}(x_j,y_j,t_j;\mathbf{q})}{\partial t^2} + \alpha\frac{\partial v_{FNN}(x_j,y_j,t_j;\mathbf{q})}{\partial t}\\ &-\gamma\Bigg(\frac{\partial^2 v_{FNN}(x_j,y_j,t_j;\mathbf{q})}{\partial x^2} + \frac{\partial^2 v_{FNN}(x_j,y_j,t_j;\mathbf{q})}{\partial y^2}\Bigg)\\ &+\Bigg(\mu_1 v_{FNN}(x_j,y_j,t_j;\mathbf{q}) - \mu_2 v^n_{FNN}(x_j,y_j,t_j;\mathbf{q})\Bigg) - h(x_j,y_j,t_j)\Bigg)^2\\ &+\sum_{(x_j,y_j,0)\in d_2}\Bigg(v_{FNN}(x_j,y_j,0;\mathbf{q}) - G(x_j,y_j)\Bigg)^2\end{aligned}$$

```
7.         Consider taking a step of descending gradient:
```

$$\mathbf{q}_{m+1} = \mathbf{q}_m - \delta_m \nabla_{\mathbf{q}}\hat{\mathbf{J}}(\mathbf{q}_m)$$

```
8.     Until its convergence.
9. Final process
```

## 4. NUMERICAL RESULTS

In this section, we apply the method described above to test examples associated with nonlinear Sine-Gordon and Klein-Gordon systems in two-dimensional space. The approximate solutions obtained by FNNs, implemented with the TensorFlow open-source framework, are compared with the exact solutions. Each experiment was performed with a learning rate of $\alpha = 0.005$ and using the hyperbolic tangent activation function (Tanh). The results show the effectiveness of our approach in finding accurate solutions to the PDEs under consideration.

### 4.1. Problem 1

Let us examine the inhomogeneous bidimensional Sine-Gordon problem as follows:

$$\begin{aligned}\frac{\partial^2 v(x,y,t)}{\partial t^2} &= \frac{\partial^2 v(x,y,t)}{\partial x^2} + \frac{\partial^2 v(x,y,t)}{\partial y^2} - \sin(v(x,y,t)) \\ &\quad + \sin(\sin(x+y+t)) + \sin(x+y+t), \qquad (x,y) \in \Omega_1, t \in [0,1]\end{aligned} \tag{12}$$

Under the following initial conditions:

$$\left\{\begin{array}{rcll} v(x,y,0) & = & \sin(x+y), & (x,y) \in \Omega_1 \\ \dfrac{\partial v}{\partial t}(x,y,0) & = & \cos(x+y), & (x,y) \in \Omega_1 \end{array}\right. \tag{13}$$

With $\Omega_1 = [0,4] \times [0,4]$, for which the analytical solution to this problem can be defined as follows:

$$v(x,y,t) = \sin(x+y+t) \tag{14}$$

Firstly, the FNN, named $\hat{v}(x,y,t) = v_{FNN}(x,y,t)$, was constructed for approximating a solution of (12) and (13). In terms of the network structure, a neural network is composed of five hidden layers, the first one containing 40 neurons, the second one containing 50 neurons, the third one containing 60 neurons, the fourth one containing 70 neurons, and the fifth one containing 80 Neurons. Then the following step consists of training the neural network parameters. We continuously optimize these parameters through minimization of the loss (9) generated based on the information from initial conditions. In addition, we choose the space-time dimension data for training a neural network. Spatio-temporal domain training data are randomly selected, and the quantity is 100 training data for each space and time domain.

Figure 2(a) presents the FNN model prediction derived by entrainment, and it is observed the resulting prediction is almost the same as the exact solution. Several moments are chosen for comparison between the approximated and the correct solution to evaluate the prediction accuracy. Figure 2(b) compares the approximate solution with the exact solution for different times $t = 0.1$ and $t = 0.9$. As a result, Figure 2(b) demonstrates that the FNN approaches and the exact solutions are significantly consistent, suggesting that the neural network models constructed have a high capacity for solving PDEs. Furthermore, this example's relative error has been computed as equal to $10^{-4}$, which confirms the performance of this approach. These results show that our method offers comparable or even better accuracy than the approaches mentioned above, demonstrating the significant value of our work.

### 4.2. Problem 2

Here we have the Sine-Gordon model, which can be used as a bi-dimensional equation:

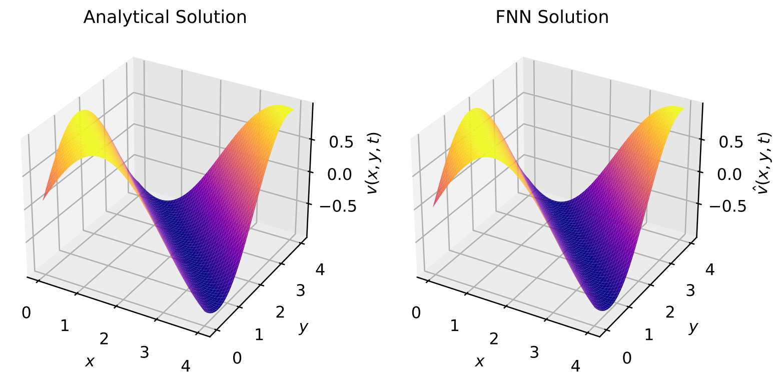
$$\frac{\partial^2 v(x,y,t)}{\partial t^2} = \frac{\partial^2 v(x,y,t)}{\partial x^2} + \frac{\partial^2 v(x,y,t)}{\partial y^2} - \sin(v(x,y,t)), \qquad (x,y) \in \Omega_1, t \in [0,1] \tag{15}$$

Depending on initial conditions:

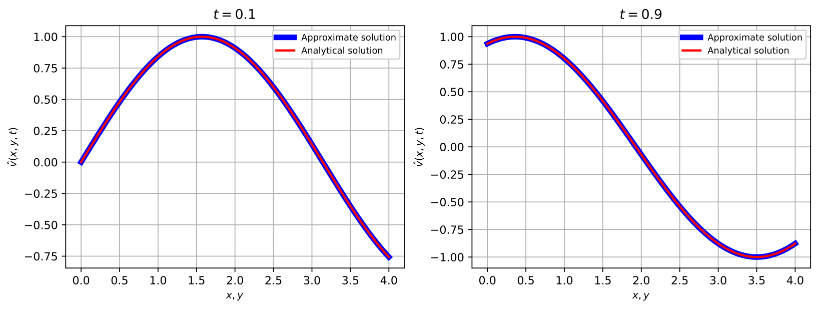
$$\left\{\begin{array}{rcll} v(x,y,0) & = & 4\tan^{-1}(\exp(x+y)), & (x,y) \in \Omega_1 \\ \dfrac{\partial v}{\partial t}(x,y,0) & = & -\dfrac{4\exp(x+y)}{1+\exp(2x+2y)}, & (x,y) \in \Omega_1 \end{array}\right. \tag{16}$$

Thus, $\Omega_1 = [-2,2] \times [-2,2]$, and the correct solution to the above problem is given as:

$$v(x,y,t) = 4\tan^{-1}(\exp(x+y-t)) \tag{17}$$



(a)



(b)

Figure 2. Comparison of analytic solution and FNN solution for the two-dimensional Sine-Gordon equation: (a) analytical and approximated solution and (b) comparing the analytical and approximated solutions

We have built a FNN for approximating the solution of (15) and (16), denoted as $\hat{v}(x, y, t)$. For the analysis of a neural network, we have used the same way as problem 1, in which we have to build an FNN with five hidden layers with the same nodes used in the previous problem. Then, we trained the neural network parameters. We continuously optimize these parameters while minimizing a loss (9), which is generated based on the information from the initial conditions. Furthermore, as in the previous experiment, we select the spatial-temporal dimension data for training an FNN. Also, spatiotemporal domain training data are randomly selected, and the quantity is 100 training data for each spatial as well as the temporal domain.

In Figure 3(a), we can observe a prediction by the FNN model, where we can see that the approximated solution is similar to the analytical solution. Different time points from the prediction outputs are selected for comparison against the analytical solution to verify the accuracy of this approximation. Figure 3(b) shows comparisons between the accurate solution and the approximate solution for the time points $t = 0.2$ and $t = 0.5$. This figure shows that the neural network approximations and the analytical solutions are very consistent, indicating that our model can successfully solve the two-dimensional Sine-Gordon equation. Finally, the relative error of our approach was calculated to be $10^{-4}$, validating its performance. These results show that our method offers comparable or better accuracy than existing approaches, demonstrating the significant value of our work.

## 4.3. Problem 3

The following equation is a bi-dimensional nonlinear Klein-Gordon equation (7) in which $n = 2$, $\alpha = 0, \gamma = 1, \mu_1 = 0$, and $\mu_2 = 1$:

$$\frac{\partial^2 v(x,y,t)}{\partial t^2} = \frac{\partial^2 v(x,y,t)}{\partial x^2} + \frac{\partial^2 v(x,y,t)}{\partial y^2} \\ -v^2(x,y,t) - xy\cos(t) + x^2y^2\cos^2(t), (x,y) \in \Omega_2, t > 0 \tag{18}$$

Under the next initial conditions:

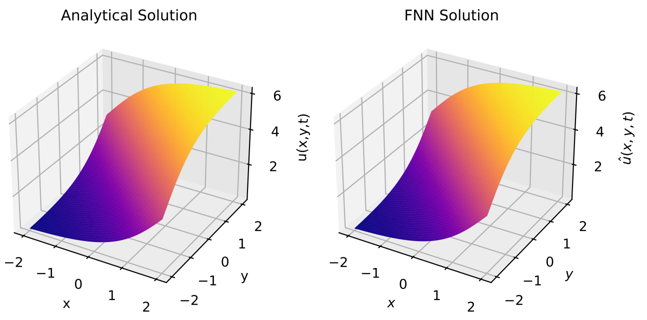
$$\begin{cases} v(x,y,0) = xy, & (x,y) \in \Omega_1 \\ \dfrac{\partial v}{\partial t}(x,y,0) = 0, & (x,y) \in \Omega_1 \end{cases} \tag{19}$$

With $\Omega_1 = [0,4] \times [0,4]$, in which the correct solution for this problem could be given in the following form:

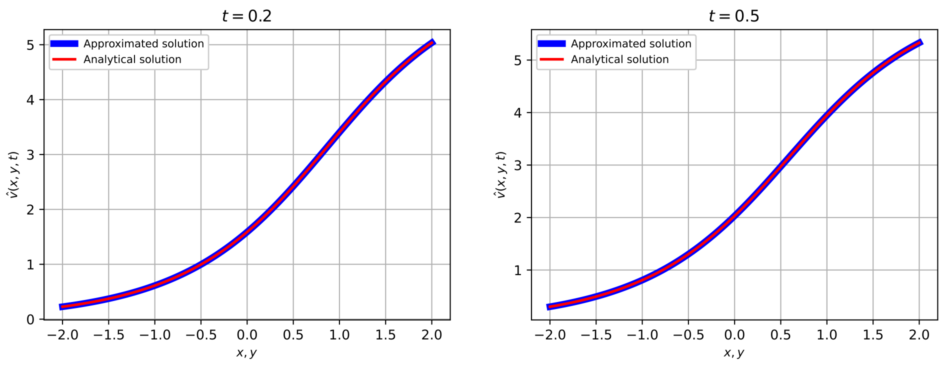
$$v(x,y,t) = xy\cos(t) \tag{20}$$

The FNN has been designed for approximating a solution of the (19), (20), indicated by the value $\hat{v}(x,y,t)$. A neural network is composed of six hidden layers. The first one contains 50 nodes, the second one contains 60 nodes, the third one contains 70 nodes, the fourth one contains 70 nodes, the fifth one contains 80 nodes, and the sixth one containing 90 nodes. We optimize these parameters continuously by minimizing a loss (10), which is generated based on information coming from initial conditions. The space-time field training data are randomly chosen, and the quantity of it is 100 training data for each space and time field.

Figure 4(a) represents the approximations of a Klein-Gordon system produced using FNNs, in which the approximated solution looks precisely like the analytical solution. Figure 4(b) compares the approximation and the analytical solution for different time points $t = 0.5$ and $t = 0.9$ to test the performance of the model. This figure shows that an approximate solution is similar to an analytical solution, confirming that our neural network method can successfully simulate the Klein-Gordon equations. In addition, the relative error was calculated to be $10^{-4}$, confirming the effectiveness of our approach. These results show that our method offers comparable or better accuracy than existing approaches, demonstrating the significant value of our work.

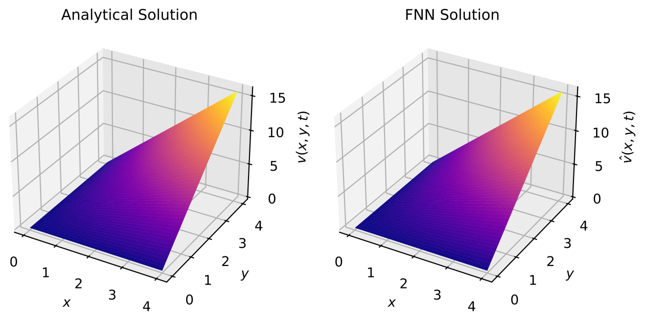


(a)

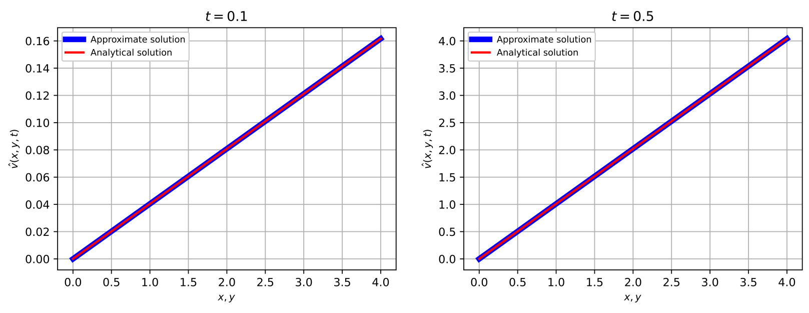


(b)

Figure 3. Visualization of numerical and analytical solutions of the Sine-Gordon equation at various times: (a) analytical and approximated solution and (b) comparing the analytical and approximated solutions



(a)



(b)

Figure 4. Comparison of numerical and analytical solutions for the two-dimensional Klein-Gordon equation: (a) analytical and approximated solution and (b) comparing the analytical and approximated solutions

## 4.4. Problem 4

Let's considered the bidimensional nonlinear Klein-Gordon equation (7) in which $n = 3, \alpha = 0$, $\gamma = 1, \mu_1 = 0$, and $\mu_2 = 1$:

$$\begin{aligned}\frac{\partial^2 v(x,y,t)}{\partial t^2} &= \frac{\partial^2 v(x,y,t)}{\partial x^2} + \frac{\partial^2 v(x,y,t)}{\partial y^2} - v^3(x,y,t) \\ &\quad + \cos(x)\cos(y)\cos(t) + \cos^3(x)\sin^3(y)\sin^3(t), \qquad (x,y) \in \Omega_2, t > 0\end{aligned} \tag{21}$$

Under the following initial conditions:

$$\begin{cases} v(x,y,0) &= & 0, & (x,y) \in \Omega_1 \\ \dfrac{\partial v}{\partial t}(x,y,0) &= & \cos(x)\cos(y), & (x,y) \in \Omega_1 \end{cases} \tag{22}$$

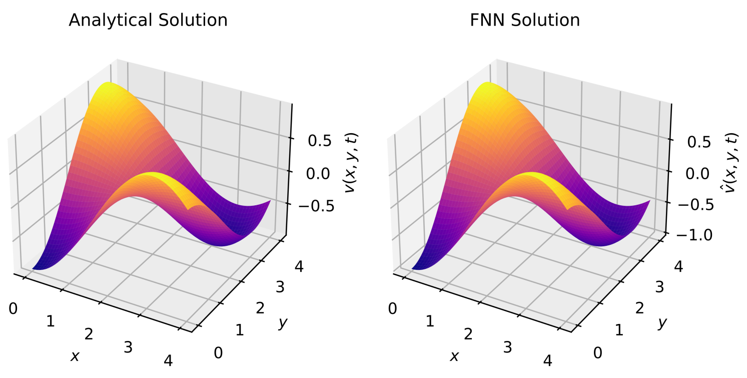
With $\Omega_1 = [0,4] \times [0,4]$, in which the analytical solution for this problem could be given in the following form:

$$v(x,y,t) = \cos(x)\cos(y)\sin(t) \tag{23}$$

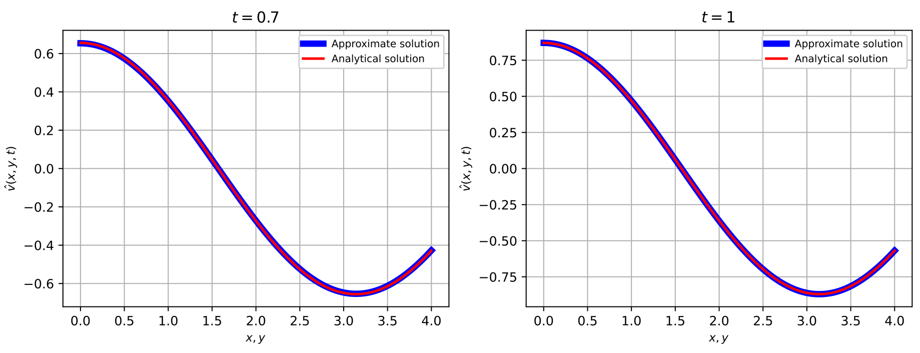
We have constructed the FNN for approximating the solution of (21) and (22), nominated by $\hat{v}(x,y,t)$. To analyze the neural network, we have used the same procedure as in problem 3, i.e. we have to build a six-hidden layer neural network of the same nodes used for a preceding problem. We continuously optimize these parameters by minimizing a loss (10), which is generated based on the information from the initial conditions. In addition, training data the space and time domains are randomly selected, and the amount of 100 of training data for each space-time domain.

In Figure 5(a), an approach of an FNN is illustrated, where we can observe that the approximated solution is identical to the exact resolution. Hence, we pick several instances in the prediction outputs for comparison to the accurate solution for testing the precision and accuracy of this approximation. Figure 5(b) shows the difference between an analytical solution and an approximate solution for different times $t = 0.7$ and $t = 1$. This figure shows that the FNN approximations and the precise solutions are significantly consistent, indicating that our model can solve this equation successfully. For this problem, the relative error was calculated to be

$10^{-4}$, confirming the effectiveness of our approach. These results show that our method offers comparable or better accuracy than existing approaches, even for complex equations, demonstrating the power and usefulness of our approach and justifying further research.



(a)



(b)

Figure 5. Numerical and exact solutions for the Klein-Gordon equation with cubic non-linearity: (a) analytical and approximated solution and (b) comparing the analytical and approximated solutions

### 4.5. Discussion and comparison

Our study investigated the use of FNNs for approximating solutions of PDEs, filling a gap in previous research that has not explicitly addressed their influence on accuracy and efficiency in complex spatiotemporal contexts. By comparing our results with those of existing traditional methods, we have shown that increasing the number of nodes in hidden layers improves accuracy without compromising computational efficiency. Although this study validated the approach for specific PDEs, future research could extend its application to even more complex and high-dimensional equations. In conclusion, our results highlight the significant potential of neural networks for solving complex scientific and engineering problems and justify further research in this area.

## 5. CONCLUSION

In this study, we proposed an approach involving solutions of PDEs by applying a methodology of deep learning designs, especially the feed-forward neural network. We have conducted experiment analyses for four major PDEs, including two-dimensional Sine-Gordon and Klein-Gordon, to determine the effectiveness of the approach by comparing its outputs to accurate solutions and estimating its mistakes. The approach obtained successful experimental results through the power of the neural network function approximation. The FNN have been deployed via the TensorFlow open source Framework. This approach has shown promise to solve this initial set of issues. Nevertheless, more numeric experimentation has to be made to reinforce our results. Even with the successful implementation of those novel deep learning techniques for solving PDEs, several significant issues concerning theory and practice need to be explored and discussed in more detail.

## REFERENCES

[1] J. Singh, D. Kumar, and S. Kumar, "An efficient computational method for local fractional transport equation occurring in fractal porous media," *Computational and Applied Mathematics*, vol. 39, no. 3, 2020, doi: 10.1007/s40314-020-01162-2.

[2] V. P. Dubey, R. Kumar, D. Kumar, I. Khan, and J. Singh, "An efficient computational scheme for nonlinear time fractional systems of partial differential equations arising in physical sciences," *Advances in Difference Equations*, vol. 2020, no. 1, 2020, doi: 10.1186/s13662-020-2505-6.

[3] S. N. M. Ruijsenaars, "Sine-gordon equation," in *Encyclopedia of Mathematical Physics*, Oxford, United Kingdom: Academic Press, 2004, pp. 576–583, doi: 10.1016/B0-12-512666-2/00187-5.

[4] R. Hirota, "Nonlinear partial difference equations iii; discrete sine-gordon equation," *Journal of the Physical Society of Japan*, vol. 43, no. 6, pp. 2079–2086, 1977, doi: 10.1143/JPSJ.43.2079.

[5] A. M. Grundland and E. Infeld, "A family of nonlinear klein–gordon equations and their solutions," *Journal of Mathematical Physics*, vol. 33, no. 7, pp. 2498–2503, 1992, doi: 10.1063/1.529620.

[6] V. Benci and D. Fortunato, "Solitary waves of the nonlinear klein-gordon equation coupled with the maxwell equations," *Reviews in Mathematical Physics*, vol. 14, no. 4, pp. 409–420, 2002, doi: 10.1142/S0129055X02001168.

[7] A. Bekir and E. Aksoy, "Exact solutions of nonlinear evolution equations with variable coefficients using exp-function method," *Applied Mathematics and Computation*, vol. 217, no. 1, pp. 430–436, 2010, doi: 10.1016/j.amc.2010.05.046.

[8] E. Y. Deeba and S. A. Khuri, "A decomposition method for solving the nonlinear klein-gordon equation," *Journal of Computational Physics*, vol. 124, no. 2, pp. 442–448, 1996, doi: 10.1006/jcph.1996.0071.

[9] D. Kaya, "A numerical solution of the sine-gordon equation using the modified decomposition method," *Applied Mathematics and Computation*, vol. 143, no. 2–3, pp. 309–317, 2003, doi: 10.1016/S0096-3003(02)00363-6.

[10] Y. Keskin, I. Çağlar, and A. B. Koç, "Numerical solution of sine-gordon equation by reduced differential transform method," *Proceedings of the World Congress on Engineering 2011, WCE 2011*, vol. 1, pp. 109–113, 2011.

[11] Y. Keskin and G. Oturanç, "Application of reduced differential transformation method for solving gas dynamics equation," *International journal of contemporary mathematical sciences*, vol. 5, no. 22, pp. 1091–1096, 2010.

[12] B. Batiha, M. S. M. Noorani, and I. Hashim, "Numerical solution of sine-gordon equation by variational iteration method," *Physics Letters, Section A: General, Atomic and Solid State Physics*, vol. 370, no. 5–6, pp. 437–440, 2007, doi: 10.1016/j.physleta.2007.05.087.

[13] E. Yusufoğlu, "The variational iteration method for studying the klein-gordon equation," *Applied Mathematics Letters*, vol. 21, no. 7, pp. 669–674, 2008, doi: 10.1016/j.aml.2007.07.023.

[14] M. S. H. Chowdhury and I. Hashim, "Application of homotopy-perturbation method to klein–gordon and sine-gordon equations," *Chaos, Solitons & Fractals*, vol. 39, no. 4, pp. 1928–1935, Feb. 2009, doi: 10.1016/j.chaos.2007.06.091.

[15] A. K. Alomari, M. S. M. Nooran, and R. M. Nazar, "Approximate analytical solutions of the klein-gordon equation by means of the homotopy analysis method," *Journal of Quality Measurement and Analysis*, vol. 4, no. 1, pp. 45–57, 2008.

[16] U. Yücel, "Homotopy analysis method for the sine-gordon equation with initial conditions," *Applied Mathematics and Computation*, vol. 203, no. 1, pp. 387–395, 2008, doi: 10.1016/j.amc.2008.04.042.

[17] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989, doi: 10.1016/0893-6080(89)90020-8.

[18] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998, doi: 10.1109/72.712178.

[19] D. R. Parisi, M. C. Mariani, and M. A. Laborde, "Solving differential equations with unsupervised neural networks," *Chemical Engineering and Processing: Process Intensification*, vol. 42, no. 8–9, pp. 715–721, 2003, doi: 10.1016/S0255-2701(02)00207-6.

[20] J. Sirignano and K. Spiliopoulos, "DGM: a deep learning algorithm for solving partial differential equations," *Journal of Computational Physics*, vol. 375, pp. 1339–1364, 2018, doi: 10.1016/j.jcp.2018.08.029.

[21] P. Chaudhari, A. Oberman, S. Osher, S. Soatto, and G. Carlier, "Deep relaxation: partial differential equations for optimizing deep neural networks," *Research in Mathematical Sciences*, vol. 5, no. 3, pp. 1–30, 2018, doi: 10.1007/s40687-018-0148-y.

[22] M. Raissi, "Forward-backward stochastic neural networks: deep learning of high-dimensional partial differential equations," *arXiv-Statistics*, pp. 1–17, 2018, doi: 10.1142/9789811280306_0018.

[23] A. Koryagin, R. Khudorozkov, and S. Tsimfer, "PyDEns: a python framework for solving differential equations with neural networks," *arXiv-Computer Science*, pp. 1–8, 2019.

[24] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and Intelligent Laboratory Systems*, vol. 39, no. 1, pp. 43–62, 1997, doi: 10.1016/S0169-7439(97)00061-0.

[25] M. M. Lau and K. H. Lim, "Review of adaptive activation function in deep neural network," in *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, IEEE, 2018, pp. 686–690, doi: 10.1109/IECBES.2018.8626714.

[26] N. Nawi, M. Ransing, and R. Ransing, "An improved learning algorithm based on the broyden-fletcher-goldfarbshanno (BFGS) method for back propagation neural networks," in *Sixth International Conference on Intelligent Systems Design and Applications*, IEEE, 2006, pp. 152–157, doi: 10.1109/ISDA.2006.95.

[27] M. H. Fun and M. T. Hagan, "Levenberg-marquardt training for modular networks," *IEEE International Conference on Neural Networks - Conference Proceedings*, vol. 1, pp. 468–473, 1996, doi: 10.1109/icnn.1996.548938.

[28] D. R. S. Saputro and P. Widyaningsih, "Limited memory broyden-fletcher-goldfarb-shanno (l-BFGS) method for the parameter estimation on geographically weighted ordinal logistic regression model (gwolr)," *AIP Conference Proceedings*, vol. 1868, 2017, doi: 10.1063/1.4995124.

[29] R. N. Singarimbun, "Adaptive moment estimation to minimize square error in backpropagation algorithm," *Data Science: Journal of Computing and Applied Informatics*, vol. 4, no. 1, pp. 27–46, 2020, doi: 10.32734/jocai.v4.i1-1160.

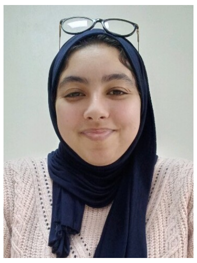
[30] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Heidelberg: Physica-Verlag HD, 2010, pp. 177–186, doi: 10.1007/978-3-7908-2604-3_16.

[31] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861–867, 1993, doi: 10.1016/S0893-6080(05)80131-5.

[32] R. J. Erb, "Introduction to backpropagation neural network computation," *Pharmaceutical Research: An Official Journal of the American Association of Pharmaceutical Scientists*, vol. 10, no. 2, pp. 165–170, 1993, doi: 10.1023/A:1018966222807.
[33] N. Ketkar, "Introduction to tensorflow," in *Deep Learning with Python*, Berkeley, CA: Apress, 2017, pp. 159–194, doi: 10.1007/978-1-4842-2766-4_11.

## BIOGRAPHIES OF AUTHORS

**Soumaya Nouna** is a researcher at the Systems Analysis and Modelling and Decision Support Research Laboratory at Hassan First University in Settat. She is an expert in mathematics, machine learning and deep learning. A doctoral researcher in mathematics and computer science, she brings a wealth of experience to her field. Her skills include the analysis of differential equations, and machine learning algorithms. She is also the author of numerous research articles and is constantly seeking to advance in her areas of expertise. She can be contacted at email: s.nouna@uhp.ac.ma.



**Assia Nouna** is a researcher at the Systems Analysis and Modelling and Decision Support Research Laboratory at Hassan First University in Settat. A doctoral researcher in mathematics and computer science. She is currently working on deep learning and satellite imagery for agricultural applications. Her research aims to enhance agricultural practices through precise soil analysis, improving crop management and yield predictions. Additionally, she has contributed to various projects and publications in the field, demonstrating her expertise in applying advanced computational techniques to solve real-world problems. She can be contacted at email: a.nouna@uhp.ac.ma.

**Mohamed Mansouri** received the Ph.D. degree in mechanical engineering and engineering sciences from the Faculty of Science and Technology, Hassan First University, Settat, Morocco, and from L'INSA, Rouen, France, in 2013. He is currently a Professor and researcher at the National School of Applied Sciences in Berrechid, Department of Electrical Engineering and Renewable Energies. His research interests include mechano-reliability study, industrial engineering, optimization of shape and reliability optimization of coupled fluid-structure systems, and energy storage systems. He can be contacted at email: m.mansouri@uhp.ac.ma.

**Dr. Ilyas Tammouch** is a distinguished researcher and academic affiliated with Ibn Tofail University in Kenitra, Morocco. His expertise spans multiple advanced fields within artificial intelligence, including machine learning, deep learning, data analysis, and evaluation systems. He has contributed extensively to the scientific community, focusing on innovative approaches to harness data-driven insights for practical applications in education, healthcare, and various industrial sectors. His research not only emphasizes technical advancements but also explores the ethical and societal impacts of technology. He is actively involved in academic collaborations and knowledge exchange initiatives, both within Morocco and internationally. He can be contacted at email: ilyas.tammouch@uit.ac.ma.

**Boujamaa Achchab** is a professor and director at ENSA Berrechid, Hassan 1st University, specializing in applied mathematics and computer science. He completed his Ph.D. at Université Claude Bernard Lyon 1 in 1995. His research focuses on numerical analysis, mathematical modeling, and computational finance. Notable works include simulations of the black-scholes equation and studies on stochastic processes. He is proficient in various mathematical and simulation software, with strong analytical skills and experience in collaborative research projects. He can be contacted at email: achchab@yahoo.fr.