Solving Nonlinear Klein-Gordon Equations via PyDEns: A Neural Network-Based PDE Solver

Soumaya Nouna[1,*], Ilyas Tammouch[2], Assia Nouna[1], Mohamed Mansouri[1]
[1]Hassan First University of Settat, ENSA Berrechid, Laboratory LAMSAD, Morocco.
[2]Laboratory of Telecommunications Systems and Decision Engineering, Faculty of Science, Ibn Tofail University, Kenitra, Morocco
E-mail: s.nouna@uhp.ac.ma
*Corresponding author

Keywords: Machine learning, artificial neural networks, deep learning, partial differential equations, Klein-Gordon equations

Received: April 14, 2025

We propose PyDEns, a neural network-based framework for solving partial differential equations (PDEs), applied to nonlinear Klein-Gordon equations. The method uses a deep feedforward neural network with four hidden layers containing $30, 40, 50$*, and* 60 *neurons respectively. The training process employs a composite loss function integrating the residuals of the PDE, initial, and Neumann boundary conditions. Optimization is carried out using stochastic gradient descent (SGD). Dataset generation is performed by sampling collocation points across the spatiotemporal domain. The model achieves high accuracy, with a maximum relative L2 error of* 2.3×10^{-4} *and RMSE as low as* 0.0021*, depending on the test case. Results show excellent agreement with known analytical solutions and fast convergence within* 600 *training iterations, demonstrating PyDEns' potential as an efficient and generalizable solver for nonlinear PDEs.*

Povzetek: PyDEns, izboljšan nevronski okvir za reševanje nelinearnih Klein-Gordonovih enačb z globokimi mrežami, zagotavlja visoko točnost in hitro konvergenco brez mreženja.

1 Introduction

Partial differential equations (PDEs) are essential in modeling a wide range of physical phenomena, from quantum mechanics and fluid dynamics to elasticity [1]. Many of these models date back to the nineteenth century and primarily involve first- and second-order derivatives. However, real-world problems often include nonlinearities and uncertainties related to material properties, external forces, or boundary conditions. Addressing these complexities requires robust computational methods that go beyond traditional analytical approaches.

One of the fundamental PDEs, the Klein-Gordon Equation (KG) [2], plays a key role in various fields such as solid-state physics, quantum mechanics, and nonlinear optics. It is particularly relevant in soliton research, where it helps analyze wave recurrence and interactions in collisionless plasma. Over the years, several analytical and numerical techniques have been developed to approximate solutions to the KG equation, including the Laplace Decomposition Method (LDM) [3], Adomian Decomposition Method (ADM) [4], and Reduced Differential Transform Method (RDTM) [5]. While effective, these methods often demand high computational resources and struggle with high-dimensional problems.

In recent years, Artificial Neural Networks (ANNs) [6] have emerged as a promising alternative for solving PDEs. Thanks to their ability to approximate complex functions and generalize across various problem domains, neural networks offer a flexible solution framework. Early research in this area began in the 1990s when Lagaris, Likas, and Fotiadis [8] introduced ANN-based techniques for solving differential equations while considering initial and boundary conditions. Lee and Kang [13] explored the application of Hopfield neural networks for first-order differential equations. Later, Malek et al. [10] combined hybrid neural networks with the Nelder-Mead simplex approach to address higher-order PDEs. More recently, deep learning-based methods such as the Deep Galerkin Method (DGM) [12] and Physics-Informed Neural Networks (PINNs) [9] have shown great potential in handling complex, high-dimensional PDE problems.

Despite these advancements, existing ANN-based methods still face challenges related to accuracy, computational efficiency, and scalability. Our work aims to tackle these issues by employing the PyDEns framework, a neural network-based approach for solving the Klein-Gordon equation. Compared to traditional numerical techniques and ANN-based methods, PyDEns provides advantages such as improved continuity, differentiability, interoperability, and lower memory requirements [24],[25]. By leveraging automated differentiation and overcoming the curse of dimensionality, this approach presents a compelling alternative for PDE resolution.

In this paper, we explore an improved ANN-based

methodology for approximating solutions to the Klein-Gordon system using the PyDEns framework. Although this work builds upon the original PyDEns framework, we propose several key enhancements tailored for solving nonlinear Klein-Gordon equations. Our contributions include a novel composite loss function formulation that integrates initial and boundary conditions more effectively, an optimized neural network architecture with increasing neuron depth across layers, and an adaptive sampling technique for training data. These improvements aim to increase the model's accuracy and generalization when applied to complex wave phenomena. Section 2 provides an overview of the Klein-Gordon equation. Section 3 introduces the Exp Function approach for deriving analytical solutions to nonlinear Klein-Gordon equations. Section 4 outlines the PyDEns methodology and its algorithmic implementation. Section 5 presents numerical experiments where we apply the PyDEns approach to different examples of the Klein-Gordon equation and compare its approximations with exact solutions obtained via the Exp Function. Finally, Section 6 summarizes our findings and suggests directions for future research.

1.1 Comparative overview of related methods

Table 1 provides a comparative overview of traditional methods previously applied to the Klein-Gordon equation, including ADM, LDM, and RDTM. These methods are compared with our proposed PyDEns framework based on criteria such as dimensionality support, ability to handle nonlinearity, computational cost, error metrics, and scalability. The comparison highlights the significant advantages of the PyDEns approach.

Table 1: Comparison of PDE solvers for Klein-Gordon equation

Method	Dims.	Nonlinear	Cost	Error	Scalability
ADM	1D/2D	Moderate	Medium	$\sim 10^{-2}$	Limited
LDM	1D	Low	Low	$\sim 10^{-1}$	Low
RDTM	1D	Low–Mod.	Low	$\sim 10^{-2}$	Limited
FDM/FEM	1D/2D	Moderate	High	$\sim 10^{-3}$	Medium
PyDEns (Ours)	1D/2D	**Yes**	**Low–Med**	$\leq 10^{-4}$	**High**

1.2 Research objectives and questions

This study aims to evaluate the capability of a deep learning-based framework, PyDEns, to solve nonlinear Klein-Gordon equations with high accuracy and efficiency. Specifically, we seek to answer the following research questions:

- Can a 4-layer artificial neural network (ANN) accurately approximate the solution of nonlinear Klein-Gordon equations with a relative L2 error less than 10^{-4} within 1000 training iterations?
- How does the proposed PyDEns framework compare to traditional methods (ADM, LDM, RDTM) in terms of scalability, dimensional applicability, and computational cost?
- What are the trade-offs between model depth, training time, and boundary condition enforcement in the neural approach?

Based on these questions, we formulate the following hypothesis: *A carefully designed deep neural network with sufficient depth and adaptive loss formulation can approximate the solution of nonlinear Klein-Gordon equations with high accuracy, surpassing traditional numerical methods in scalability and generalization.*

2 Klein-Gordon equation

The Klein-Gordon equation, introduced by Oskar Klein and Walter Gordon in 1926, is a wave equation that applies to particles with zero spin like mesons and scalar bosons in the context of relativity. This equation is a quantum mechanical formulation that satisfies the principles of both quantum mechanics and special relativity. It is derived by combining the de Broglie hypothesis, which posits that particles possess wave-like properties, with the relativistic energy-momentum relation. The Klein-Gordon equation, which is a second-order partial differential equation, is utilized to depict the time variation of a particle's wave function. This equation also retains its significance as a fundamental instrument for comprehending the actions of particles in the quantum world.

The nonlinear Klein-Gordon (NLKG) system takes on the following form :

$$\frac{\partial^2 u}{\partial t^2}(x,t) - \mu^2 \frac{\partial^2 u}{\partial x^2}(x,t) + \frac{dV(u(x,t))}{du} = f(x,t), \quad x \in [c,d], t \in [0,T] \tag{1}$$

with $\frac{dV(u(x,t))}{du}$ is a non-linear feature of u selected to be the derivation of the potential Energy $V(u) = \frac{\mu}{2}u^2(x,t) + \frac{\lambda}{4}u^4(x,t)$ (see [28]). Eq.(1) appears in many different physics problems, for example, wave propagation through ferromagnetic material with rotation of magnetization direction and lasing pulses through two states environment. The NLKG equation considered in this study is expressed as follows [**?**] :

$$\frac{\partial^2 u}{\partial t^2}(x,t) - \mu^2 \frac{\partial^2 u}{\partial x^2}(x,t) + \mu u(x,t) + \lambda u^3(x,t) = f(x,t), \quad x \in [c,d], t \in [0,T] \tag{2}$$

under Initial Conditions (IC)

$$\begin{aligned} u(x,0) &= \psi_1(x), \qquad x \in [c,d] \\ \frac{\partial u}{\partial t}(x,0) &= \psi_2(x), \qquad x \in [c,d] \end{aligned} \tag{3}$$

and Boundary Conditions (BC)

$$\frac{\partial u}{\partial x}(c,t) = \frac{\partial u}{\partial x}(d,t) = 0, \qquad t \in [0,T] \tag{4}$$

Eq.(2) represents the specific expression of Eq.(1). To guarantee that there is no overall movement of particles across the boundary, we apply the Neumann boundary condition (4). This condition requires setting the normal derivative of the wave function to zero at the boundary, which indicates the flow of particles. This condition ensures that there is neither an inflow nor an outflow of particles within the region of interest. In addition, the utilization of the Neumann boundary condition has the added benefit of establishing the uniqueness of the solution to the Klein-Gordon equation. This implies that there is only one feasible solution that complies with both the equation and the boundary conditions. The nonlinear Klein-Gordon equation is particularly interesting because it exhibits a range of complex and fascinating behaviors, including solitons, chaos, and turbulence. Mathematical analysis of the nonlinear Klein-Gordon equation involves studying the properties of its solutions, such as their regularity, stability, and asymptotic behavior [30], [31], [32]. Furthermore, various numeric approaches were developed for solving the Klein-Gordon equations. However, we will use the exp function approach to determine the exact solutions and the ANN approach to find the approximated solution of these equations.

3 Exact solutions

Exp function approach has been suggested for solving PDEs [26], [27]. In this part, we will present the method of exp function to obtain the exact solutions of Klein-Gordon systems. We assume $f(x,t) = 0$ in order to derive exact solutions for the homogeneous nonlinear Klein-Gordon equation. Following are the essential steps of the approach.

By applying the following transformation :

$$u(x,t) = u(X),\ X = x - at \tag{5}$$

with a is constant, Eq.(2) is written as follows:

$$a^2u'' - \mu^2u'' + \mu u + \lambda u^3 = 0 \tag{6}$$

We look for traveling wave solutions of the form $u(x,t) = u(X)$, where $X = x-at$. This transformation reduces the PDE into an ODE. This assumption implies that the wave profile is stationary in the moving frame.

Substituting $u = v + s$ into equation (6), we obtain :

$$(a^2-\mu^2)v'' + \mu v + \mu s + \lambda v^3 + 3\lambda s v^2 + 3\lambda s^2 v + \lambda s^3 = 0 \tag{7}$$

If $s(\lambda s^2 + \mu) = 0$, than either $s = 0$ or $s = \mp\sqrt{-\frac{\mu}{\lambda}}$. By the principle of homogeneous balance, we obtain $r = 1$.

Case 1 $s = 0$. We introduce the ansatz l, where k, as part of the Exp-function method to construct a rational exponential solution to the nonlinear equation.

$$l = \exp(-kX),\ \ k = \sqrt{\frac{\mu}{\mu^2 - a^2}} \tag{8}$$

This rational ansatz is inspired by the Exp-function method, which assumes that the solution can be represented as a rational function of exponential terms. Note that this ansatz is a simplified version of the more general form introduced in Case 2, where we assume $e_2 = 0$ and $b_1 = 0$ for algebraic convenience. This simplification results from setting $s = 0$, which reduces the nonlinearity of the equation. The number of terms $(r = 1)$ is selected based on the balance between nonlinear and linear terms in the equation.

$$u(X) = \frac{\sum_{j=0}^{2r} e_j l^j}{\sum_{j=0}^{2r} b_j l^j} = \frac{e_0 + e_1 l + e_2 l^2}{b_0 + b_1 l + b_2 l^2} \tag{9}$$

By substituting equation (8) and equation (9) in equation (6), we obtain a series of Algebraic equations of $l^j, j = 0, 1, ..., 6$. By substituting the ansatz into the transformed equation and equating powers of l^j, we derive the following overdetermined algebraic system.

$$\begin{cases} \lambda e_0^2 + \mu e_0 b_0^2 = 0 \\ 3\mu e_0 b_1 b_0 + 3\lambda e_0^2 e_1 = 0 \\ 3\lambda e_0^2 e_2 + 3\lambda e_0 e_1^2 - 3\mu e_2 b_0^2 + 3\mu e_1 b_0 b_1 + 6\mu e_0 b_2 b_0 = 0 \\ \mu e_1 b_1^2 - \mu e_0 b_1 b_2 + 6\lambda e_0 e_1 e_2 + 8\mu e_1 b_0 b_2 \\ \qquad\qquad -\mu e_2 b_0 b_1 + \lambda e_1^3 = 0 \\ 3\mu e_1 b_1 b_2 + 3\lambda e_1^2 e_2 - 3\mu e_0 b_2^2 + 6\mu e_2 b_0 b_2 + 3\lambda e_0 e_2^2 = 0 \\ 3\lambda e_1 e_2^2 + 3\mu e_2 b_1 b_2 = 0 \\ \mu e_2 b_2^2 + \lambda e_2^3 = 0 \end{cases}$$

By solving this system, the following results can be obtained

$$b_0 = -\frac{\lambda e_1^2}{8\mu b_2}, b_1 = 0, b_2 = b_2, e_0 = 0, e_1 = e_1, e_2 = 0 \tag{10}$$

with $b_2 \neq 0$, and e_1 a constant of arbitrary value. Replacing (9) with (10), the following Klein-Gordon solution is obtained

$$u_a(x,t) = u(X) = \frac{e_1}{-\frac{\lambda e_1^2}{8\mu b_2} l^{-1} + b_2 l} \tag{11}$$

with $l = \exp\left(-\sqrt{\frac{\mu}{\mu^2-a^2}}X\right), X = x - at.$

Assume $e_1 = 2, b_2 = \sqrt{\frac{-\lambda}{2\mu}}, \mu \cdot \lambda < 0, \frac{\mu}{\mu^2-a^2} > 0$, so the analytical solution of a Klein-Gordon system is:

$$u_{an}(x,t) = \mp\sqrt{\frac{-2\mu}{\lambda}}\ sech\left(\sqrt{\frac{\mu}{\mu^2-a^2}}(x-at)\right). \tag{12}$$

Case 2 $s = \mp\sqrt{-\frac{\mu}{\lambda}}$. The linear equation solution to Eq.(7) has the following form :

$$l = \exp(-kX), k = \sqrt{\frac{2\mu}{a^2 - \mu^2}}. \tag{13}$$

We can therefore suppose that:

$$u(X) = \frac{\sum_{j=0}^{2r} e_j l^j}{\sum_{j=0}^{2r} b_j l^j} = \frac{e_0 + e_1 l + e_2 l^2}{b_0 + b_1 l + b_2 l^2} \tag{14}$$

By substituting equation (13) and equation (14) in equation (6), we obtain a series of Algebraic equations of $l^j, j = 0, 1, ..., 6$. By substituting the ansatz into the transformed equation and equating powers of l^j, we derive the following overdetermined algebraic system.

$$\begin{cases} \lambda e_0^3 + \mu e_0 b_0^2 = 0 \\ 3\lambda e_0^2 e_1 + 3\mu e_1 b_0^2 = 0 \\ 9\mu e_2 b_0^2 + 3\lambda e_0^2 e_2 + 3\lambda e_0 e_1^2 + 3\mu e_0 b_1^2 - 6\mu e_0 b_2 b_0 = 0 \\ 8\mu e_0 b_1 b_2 + \mu e_1 b_1^2 + \lambda e_1^3 + 8\mu e_2 b_0 b_1 \\ \quad +6\lambda e_0 e_1 e_2 - 10\mu e_1 b_0 b_2 = 0 \\ 3\mu e_2 b_1^2 + 3\lambda e_1^2 e_2 - 6\mu e_2 b_0 b_2 \\ \quad +3\lambda e_0 e_2^2 + 9\mu e_0 b_2^2 = 0 \\ 3\lambda e_1 e_2^2 + 3\mu e_1 b_2^2 = 0 \\ \lambda e_2^3 + \mu e_2 b_2^2 = 0 \end{cases}$$

By solving this system, the following results can be obtained

$$b_0 = \mp\frac{\mu b_1^2 + e_1^2\lambda}{4\mu\sqrt{-\frac{\lambda}{\mu}}e_2}, b_2 = \mp\sqrt{-\frac{\lambda}{\mu}}e_2, b_1 = b_1, \quad e_0 = \frac{\mu b_1^2 + e_1^2\lambda}{4e_2\lambda}, e_1 = e_1, e_2 = e_2 \tag{15}$$

where $e_2 \neq 0$, and e_1, b_1 as arbitrary constants. Replacing (14) with (15), the following Klein-Gordon solution is obtained

$$u_a(x,t) = \frac{\frac{\mu b_1^2 + e_1^2\lambda}{4e_2\lambda} + e_1 l + e_2 l^2}{\mp\frac{\mu b_1^2 + e_1^2\lambda}{\sqrt{-\frac{\lambda}{\mu}}4\mu e_2} + b_1 l \mp \sqrt{-\frac{\lambda}{\mu}}e_2 l^2} \tag{16}$$

with $l = \exp\left(-\sqrt{\frac{2\mu}{a^2-\mu^2}}X\right), X = x - at.$

Assume $e_1 = 1, e_2 = -1, b_1 = \mp\sqrt{\frac{-\lambda}{\mu}}, \mu \cdot \lambda < 0, \frac{\mu}{a^2-\mu^2} > 0$, so the analytical solution of a Klein-Gordon system is:

$$u_{an}(x,t) = \mp\sqrt{\frac{-\mu}{\lambda}}\, tanh\left(\sqrt{\frac{\mu}{2(a^2-\mu^2)}}(x-at)\right). \tag{17}$$

4 Neural network solutions

In this section, we present the neural network architecture developed for this study and provide a comprehensive description of the PyDEns method, a neural network-based approach for solving partial differential equations (PDEs). Unlike traditional numerical methods, PyDEns offers a unique way of handling data generation, enforcing boundary conditions, and formulating the loss function, making it particularly well-suited for solving complex PDEs with high-dimensional inputs. Similar PDE-based frameworks have also been applied successfully to image enhancement tasks, such as the adaptive diffusion flow introduced in [14]. Our methodology is designed to ensure accuracy, efficiency, and generalizability, making it a robust alternative to conventional approaches.

To ensure clarity and reproducibility, we provide a detailed step-by-step breakdown of the entire process, from the problem formulation to the implementation details. We explain the rationale behind our architectural choices, including network design, activation functions, and optimization techniques, while also discussing how our approach aligns with theoretical foundations in deep learning and differential equation solving. Additionally, we highlight how PyDEns differs from other machine learning-based PDE solvers by emphasizing its unique data-driven strategy and adaptive learning process.

4.1 Artificial neural networks

Artificial neural networks (ANNs) [15] are a collection of algorithms that simulate human brain processes to identify patterns. Those systems learn to solve issues by considering examples, typically without being designed with domain-specific rules. Furthermore, ANNs are inspired by biological neural networks that form the core of animal brains. The word Neural is derived from neurons or nerve cells, which are the basic functional units of the biological nervous system that make up the major part of the brain. The goal behind a neural network is to simplify and simulate many closely linked brain cells in a computer program so that it may learn to detect patterns. The most essential aspect of an artificial neural network is that it does not require explicit programming to learn. ANN has been used effectively in various practical applications such as differential equation problems. A Perceptron model is a type of neural network proposed by Frank Rosenblatt in 1958, and it is one of the simplest ANN designs that is widely used today. Multilayer Perceptron (MLP) [16] is a perceptron that has more than one hidden layer (Figure 1). Thus, multi-layer Neural Networks can be mathematical as direct graphed layers (see definition 4.1). In this section, we will also review the general Neural Network approach for solving the Klein-Gordon equation.

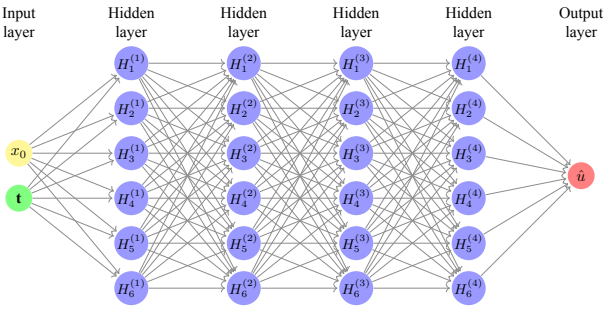


Figure 1: Multilayer Perceptron (MLP) with 4 hidden layers composed of 30, 40, 50, and 60 neurons respectively. Each layer uses sigmoid activation.

Let us suppose a four-layered neural network (Figure 1) having the Inputs x, as well as t, that consist of the Hidden Layers comprised of p,q, l, and r neurons correspondingly. Its goal is to approximate $u(x,t)$ with an approximation functional $\hat{u}(x,t;\theta)$ to obtain a trained multilayer perceptron since x, as well as t, represent all Inputs while θ comprises the weights and biases of the adjustable parameters. At each input node x as well as t, a procedure starts at the Entry Layer at its First Hidden Layer, described below

$$H^{(k)} = \sum_{i=1}^{p}(w_i^{(x)}x + w_i^{(t)}t) + b_k, \qquad k = 1 \tag{18}$$

Here, b_k are the biases of the first hidden layer and $w_i^{(x)}, w_i^{(t)}$ are the weights of the first hidden layer's inputs x and t, accordingly. Therefore, it is activated by the Sigmoid function [**?**] as below

$$\sigma(H^{(k)}) = \frac{\exp(H^{(k)})}{1+\exp(H^{(k)})}, \qquad k = 1 \tag{19}$$

This activation function holds specific properties concerning artificial neural networks, such as being a Universal Approximator of Continuous Functions (see Theorem 4.1). In our context, the sigmoid function is also strictly increasing, which gives it useful monotonic behavior. However, we do not assume or require injectivity of the entire neural network. The sigmoid function was mainly chosen for its full differentiability and smooth gradient behavior, which are advantageous when computing high-order derivatives in PDE-driven loss functions. Compared to ReLU or GELU, sigmoid avoids non-differentiable points and enables more stable training using automatic differentiation. The following step consists in feeding the $(k+1)$ Hidden Layer through (k) Hidden Layer according to the following formula:

$$H^{(k+1)} = \sum_{j=1}^{q}\sum_{i=1}^{p} w_{ij}^{(k)}\sigma(H^{(k)}) + b_{k+1}, \qquad k = 1,2,3 \tag{20}$$

where b_{k+1} represents the biases of $(k+1)$ hidden layer and $w_{ij}^{(k)}$ represents the weights of (k) hidden layer to $(k+1)$ hidden layer. Finally, the output layer takes the following formula :

$$\hat{u}(x,t;\theta) = \sum_{m=1}^{r} v_m\sigma(H^{(k+1)}), \qquad k = 3 \tag{21}$$

where v_m represents weights for the fourth layer at an output Hidden Layer. Furthermore, the n-th derivations for $\hat{u}(x,t;\theta)$ using Automatic Differentiation are therefore easily expressed in terms of

$$\frac{\partial^n \hat{u}}{\partial x^n}(x,t;\theta) = \sum_{m=1}^{r}\frac{\partial^n v_m\sigma(H^{(k+1)})}{\partial x^n}, \qquad for\ n = 1,2,...,r\ and\ k = 3 \tag{22}$$

$$\frac{\partial^n \hat{u}}{\partial t^n}(x,t;\theta) = \sum_{m=1}^{r}\frac{\partial^n v_m\sigma(H^{(k+1)})}{\partial t^n}, \qquad for\ n = 1,2,...,r\ and\ k = 3 \tag{23}$$

In this work, we use PyTorch's automatic differentiation engine (autograd) to compute all spatial and temporal derivatives of the neural network output $\hat{u}(x,t;\theta)$. This tool provides efficient and accurate derivative computation directly from the computational graph, which is critical for solving PDEs without relying on finite difference or symbolic methods. Compared to Physics-Informed Neural Networks (PINNs), which incorporate boundary and initial conditions via weighted loss terms, our method integrates boundary conditions—especially Neumann conditions—explicitly into the loss formulation. Unlike Deep Galerkin Methods (DGM), which may have difficulty enforcing boundary conditions on irregular domains, PyDEns maintains stability and precision through explicit residual-based formulation and automatic differentiation.

In the neural network approach, we approximate the solution $u(x,t)$ with a neural network $\hat{u}(x,t;\theta)$, where θ are the network parameters. Using automatic differentiation, we compute the derivatives needed for the PDE. The loss function is then designed to minimize the error in the PDE as well as the initial and boundary conditions. The efficiency provided by an approach $\hat{u}(x,t;\theta)$ was evaluated by computing a Loss function. As a result, an objective for almost all PDE-solving algorithms is also minimizing a Loss function extremely efficiently. Every one of the methods [18], [19], [**?**] uses a different strategy to construct the Loss function. We may conclude that our artificial neural networks are truly a solution to the partial differential equations when a Loss function approaches zero. To guarantee that a Loss function has been minimized, we must update both setting biases and weights to be optimal. As a result, any optimization strategy may be used. To accelerate the convergence of the approach utilized in this work employed the Stochastic Gradient Descent (SGD) optimizer [**?**].

Definition 4.1. *A multi-layer perceptron a function* $G : \mathbb{R}^p \to \mathbb{R}^q$. *It can be considered an p-N-q-Perceptron (p Inputs, N Hidden Layers, and q Outputs) when it is described as a function of the following shape*

$$G(\mathbf{x}) = \sigma\big(g_N(g_{N-1}(...g_1(\mathbf{x})))\big), \quad with\ \mathbf{x}^t = (\mathbf{x}_1,...,\mathbf{x}_p)$$

In particular, when dealing with affine functions $g_j(\mathbf{x}) = \mathbf{w}^j\mathbf{x} + \mathbf{b}^j$, *we have the linear multi-layered Perceptron*

$$G(\mathbf{x}) = \sigma\big(\mathbf{w}^N...(\mathbf{w}^2(\mathbf{w}^1\mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2)... + \mathbf{b}^N\big), \quad with\ \mathbf{x}^t = (\mathbf{x}_1,...,\mathbf{x}_p)$$

that maps the non-linear functional σ *with the combination for p Affine functions* $\mathbf{w}^j\mathbf{x} + \mathbf{b}^j, j = 1,...,N$.

Theorem 4.1. *For all* $\varepsilon > 0$ *with a continuous function g over some compact subset* $\mathbf{K} = [c,d] \subset \mathbb{R}^p$. *There is the*

Neural Network having an Activation functional σ which containing one Hidden Layer with a limited number p of neurons which, given slight hypotheses regarding its activation function, may approach g, i.e

$$\left\|g-\sigma\right\|_{\infty}=\sup_{\mathbf{x}\in\mathbf{K}}\left|g(\mathbf{x})-\sigma(\mathbf{x})\right|<\varepsilon.$$

4.2 The PyDEns method

PyDEns is an artificial neural network-based approach for solving partial differential equations that evolved from the approach described in [22]. In this study, we modified the original PyDEns setup to better suit the nonlinear dynamics of the Klein-Gordon equations. Our version implements a deeper neural network with four hidden layers (ranging from 30 to 60 neurons), employs a specifically crafted loss function that incorporates Neumann boundary conditions, and introduces a more effective data generation strategy to improve solution accuracy and stability. These changes distinguish our approach from previous applications of PyDEns and are validated through our experimental results. This approach differs in how the data points are generated, the boundary conditions, and the definition of the loss function.

Consider the developmental type of linear Klein-Gordon equation

$$\frac{\partial^2 u}{\partial t^2}(x,t)-\frac{\partial^2 u}{\partial x^2}(x,t)+u(x,t)=f(x,t), x\in[c,d], t\in[-T,T] \tag{24}$$

under the Initial Conditions (IC)

$$P(x)\begin{cases} u(x,0) &= \kappa_1(x), & x\in[c,d] \\ \dfrac{\partial u}{\partial t}(x,0) &= \kappa_2(x), & x\in[c,d] \end{cases} \tag{25}$$

We also take a global type for a non-linear Klein-Gordon Equation

$$\frac{\partial^2 u}{\partial t^2}(x,t)-\mu^2\frac{\partial^2 u}{\partial x^2}(x,t)+\mu u(x,t)+\lambda u^3(x,t)=f(x,t), \quad x\in[c,d], t\in[0,T] \tag{26}$$

under the Initial Conditions (IC)

$$h(x)\begin{cases} u(x,0) &= \rho_1(x), & x\in[c,d] \\ \dfrac{\partial u}{\partial t}(x,0) &= \rho_2(x), & x\in[c,d] \end{cases} \tag{27}$$

and the Boundary Conditions (BD)

$$\frac{\partial u}{\partial x}(c,t)=\frac{\partial u}{\partial x}(d,t)=\varphi(x,t), \qquad t\in[0,T] \tag{28}$$

In our implementation, we consider homogeneous Neumann boundary conditions as defined in Eq.5. This corresponds to setting $\varphi(x,t)=0$ in Eq.28, which ensures zero flux at the spatial boundaries. We approximate solution $u(x,t)$ at 24 and 26 with an artificial neural network $\hat{u}(x,t;\theta)$ (see eq.(21)). Loss functions that are related with the two learning problems are defined as follows, respectively:

$$Loss_1(\theta)=\left(\frac{\partial^2\hat{u}}{\partial t^2}(x,t;\theta)-\frac{\partial^2\hat{u}}{\partial x^2}(x,t;\theta)+\hat{u}(x,t;\theta)-f(x,t)\right)^2_{x,t\in A_1,\vartheta_1}+\left(\hat{u}(x,0;\theta)-P(x)\right)^2_{x\in A_2,\vartheta_2} \tag{29}$$

$$Loss_2(\theta)=\left(\frac{\partial^2\hat{u}}{\partial t^2}(x,t;\theta)-\mu^2\frac{\partial^2\hat{u}}{\partial x^2}(x,t;\theta)+\mu\hat{u}(x,t;\theta)+\lambda\hat{u}^3(x,t;\theta)-f(x,t)\right)^2_{x,t\in B_1,\vartheta_1}+\left(\hat{u}(x,0;\theta)-h(x)\right)^2_{x\in B_2,\vartheta_2}+\left(\hat{u}(x,t;\theta)-\varphi(x,t)\right)^2_{x,t\in B_3,\vartheta_3} \tag{30}$$

We generated n points over the domains $[c,d]\times[-T,T]$, $[c,d]\times\{0\}$, $[c,d]\times[0,T]$, $[c,d]\times\{0\}$, and $\sigma[c,d]\times[0,T]$, corresponding respectively to the sets A_1, A_2, B_1, B_2, and B_3. The weights ϑ_1, ϑ_2, and ϑ_3 represent the relative importance (or measures) of each term in the loss function. The objective is to train the neural network $\hat{u}$ such that it satisfies the conditions defined by Eqs. (24)–(28).

In the next step, we use stochastic gradient descent to minimize the composite loss functions defined in Eqs. (29) and (30). The full training procedure adapted for the two studied problems is summarized in Algorithms 1 and 2.

5 Results and discussion

In this section, we present the numerical results of our approach to solving both linear and non-linear Klein-Gordon equations. We analyze how data points are distributed and how well our neural network architecture performs in handling these equations. Specifically, we define the number of points in each set based on equations (24), (25), (26), (27), and (28). These equations correspond to A_1, A_2, B_1, B_2, and B_3, which represent the boundary and initial conditions of the partial differential equations.

To assess the model's effectiveness, we calculate the loss function over the domain (x,t), ensuring that the predicted solutions remain consistent with the theoretical constraints. To validate the accuracy of our neural network-based solver, we compare the predicted solution $\hat{u}(x,t)$ with the exact analytical solution defined in Eqs.(11)−(17). These expressions describe the reference solution used to evaluate model performance across different PDE scenarios. The comparison reveals that the ANN outputs closely match the analytical form, with RMSE values ranging between 1.8×10^{-3} and 2.5×10^{-3} across all test cases. This confirms that our method not only satisfies the imposed boundary and initial conditions (Eqs.12−16), but

Algorithm 1 The approach's algorithm for solving linear Klein-Gordon Problem

1- Produce n locations in lots A_1, A_2 based on $[c,d]\times[-T,T], [c,d]\times\{0\}$. Pull distributions ϑ_1, ϑ_2.
2- While $s \leq \ isteration$ Do
3- For $(x,t)\in[c,d]\times[-T,T]$ at A_1, A_2 Do
4- Compute the output of the neural network

$$\begin{aligned}\hat{u}(x,t;\theta) &= \sum_{m=1}^{r} v_m\sigma(H^{(k+1)}),\ \frac{\partial^2\hat{u}}{\partial x^2}(x,t;\theta)\\ &= \sum_{m=1}^{r}\frac{\partial^2 v_m\sigma(H^{(k+1)})}{\partial x^2},\ \frac{\partial^2\hat{u}}{\partial t^2}(x,t;\theta)\\ &= \sum_{m=1}^{r}\frac{\partial^2 v_m\sigma(H^{(k+1)})}{\partial t^2}, \quad k=3\end{aligned}$$

5- Calculate the loss function :

$$\begin{aligned}Loss_1(\theta) = \Bigg(&\frac{\partial^2\hat{u}}{\partial t^2}(x,t;\theta) - \frac{\partial^2\hat{u}}{\partial x^2}(x,t;\theta)\\ &+\hat{u}(x,t;\theta) - f(x,t)\Bigg)^2\\ &+\left(\hat{u}(x,0;\theta) - P(x)\right)^2\end{aligned}$$

6- End For
7- Using the Stochastic Gradient Descent to optimize biases and weights.
8- End While

Algorithm 2 The approach's algorithm for solving nonlinear Klein-Gordon Problem

1- Produce n locations in lots B_1, B_2, B_3 based on $[c,d]\times[0,T], [c,d]\times\{0\}, \sigma[c,d]\times[0,T]$. Pull distributions $\vartheta_1, \vartheta_2, \vartheta_3$.
2- While $q \leq \ isteration$ Do
3- For $(x,t)\in[c,d]\times[0,T]$ at B_1, B_2, B_3 Do
4- Compute the output of the neural network

$$\begin{aligned}\hat{u}(x,t;\theta) &= \sum_{m=1}^{r} v_m\sigma(H^{(k+1)}),\ \frac{\partial^2\hat{u}}{\partial x^2}(x,t;\theta)\\ &= \sum_{m=1}^{r}\frac{\partial^2 v_m\sigma(H^{(k+1)})}{\partial x^2},\ \frac{\partial^2\hat{u}}{\partial t^2}(x,t;\theta)\\ &= \sum_{m=1}^{r}\frac{\partial^2 v_m\sigma(H^{(k+1)})}{\partial t^2}, \quad k=3\end{aligned}$$

5- Calculate the loss function :

$$\begin{aligned}Loss_2(\theta) = \Bigg(&\frac{\partial^2\hat{u}}{\partial t^2}(x,t;\theta) - \mu^2\frac{\partial^2\hat{u}}{\partial x^2}(x,t;\theta) + \mu\hat{u}(x,t;\theta)\\ &+\lambda\hat{u}^3(x,t;\theta) - f(x,t)\Bigg)^2 + \left(\hat{u}(x,0;\theta) - h(x)\right)^2\\ &+\left(\hat{u}(x,t;\theta) - \varphi(x,t)\right)^2\end{aligned}$$

6- End For
7- Using the Stochastic Gradient Descent to optimize biases and weights.
8- End While

also approximates the true solution with high numerical fidelity. Our neural network consists of four hidden layers, with an increasing number of neurons in each layer to improve learning capacity: 30 neurons in the first layer, 40 in the second, 50 in the third, and 60 in the fourth. We selected this architecture based on preliminary tests that showed improved performance with increasing depth and neuron count. The use of sigmoid as activation function ensures smooth and differentiable outputs, which is essential for computing PDE derivatives. Using more shallow networks or fewer neurons caused higher error and slower convergence. We chose the SGD optimizer for its stability in training and generalization ability. While we do not provide a full ablation study, our choices were guided by empirical observation of convergence and error minimization.

The entire implementation is developed using Python and the PyTorch library, leveraging its flexibility and efficiency for solving high-dimensional PDEs. In the following discussion, we examine the significance of these results, compare them with existing approaches, and highlight both the strengths and limitations of our method.

5.1 Problem 1 : wave of kink

The nonlinear Klein-Gordon equation representing a kink wave is expressed as follows:

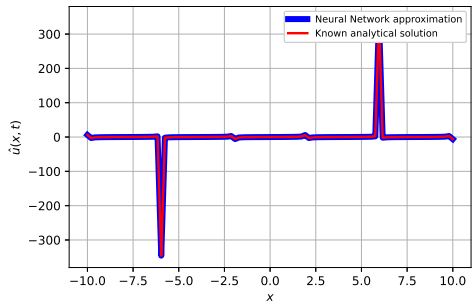
$$\frac{\partial^2 u}{\partial t^2}(x,t) - \mu^2\frac{\partial^2 u}{\partial x^2}(x,t) + \mu u(x,t) + \lambda u^3(x,t) = 0, \quad x\in[-10,10], t\in[0,T] \tag{31}$$

Based on the initial conditions as the next:

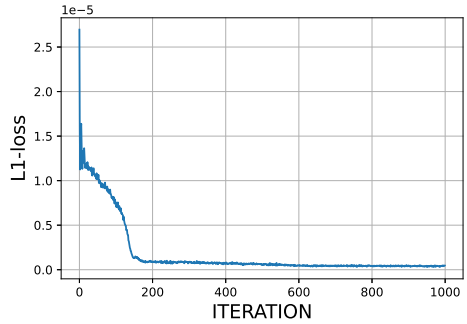
$$\begin{cases} u(x,0) &= \sqrt{\dfrac{-\mu}{\lambda}}\tanh(kx), & x\in[-10,10]\\ \dfrac{\partial u}{\partial t}(x,0) &= -a\sqrt{\dfrac{-\mu}{\lambda}}k\ sech^2(kx), & x\in[-10,10]\end{cases} \tag{32}$$

where $\varphi(x,t) = 0$, $\mu = 0.1, \lambda = -1, a = 0.3$, and $T = 1$. This problem's exact solution has been as follows

$$u_{an}(x,t) = \sqrt{\frac{-\mu}{\lambda}}\tanh\left(\sqrt{\frac{\mu}{2(a^2-\mu^2)}}(x-at)\right) \tag{33}$$



(a)



(b)

Figure 2: Performance of the neural network approach in solving the Klein-Gordon equations. (a) The exact solution compared with the neural network-predicted solution. (b) The loss function behavior, demonstrating convergence over training iterations.

In this study, the neural network approach was trained using approximately 100 time and space elements within the field. Figure 2 presents a comparison between the analytical and neural network-based solutions, highlighting the accuracy and efficiency of the proposed method. Figure 2(a) shows the analytical solution (in red) alongside the neural network-predicted solution (in blue) for equations 31 and 32, where both solutions are nearly indistinguishable. Figure 2(b) illustrates the relative error of the method, demonstrating that after 600 iterations, the maximum error is within 10^{-4}, confirming the model's high precision. These results validate the effectiveness of the neural network approach, proving that it converges in under 600 iterations with relative L2 error below 2.3×10^{-4}, demonstrating both accuracy and training efficiency. Total variation-based PDE methods have also been explored for robust signal recovery in noisy environments, as demonstrated in [33], providing a relevant contrast with data-driven neural approaches. Furthermore, this study reinforces the potential of machine learning algorithms to enhance numerical analysis, providing a powerful alternative to traditional methods.

5.2 Problem 2 : unique-soliton

Although the PDE and initial condition forms in this example are structurally similar to those in Problem 1, the choice of parameters leads to a fundamentally different solution behavior. Here, the configuration corresponds to a unique localized soliton, in contrast to the kink-type solution presented earlier.

We examine a nonlinear Klein-Gordon equation of the following form :

$$\frac{\partial^2 u}{\partial t^2}(x,t) - \mu^2 \frac{\partial^2 u}{\partial x^2}(x,t) + \mu u(x,t) + \lambda u^3(x,t) = 0, \quad x \in [-10,10], t \in [0,T] \tag{34}$$

Based on the initial conditions as the next:

$$\begin{cases} u(x,0) &= \sqrt{\dfrac{-2\mu}{\lambda}}\, sech(\kappa x), \\ \dfrac{\partial u}{\partial t}(x,0) &= a\kappa\sqrt{\dfrac{-2\mu}{\lambda}}\, sech(\kappa x)\tanh(\kappa x), \end{cases} \tag{35}$$

in which $x \in [-10,10]$,$\varphi(x,t) = 0$,$\mu = 0.3, \lambda = -1, a = 0.25$, and $T = 1$. This problem's exact solution has been as follows

$$u_{an}(x,t) = \sqrt{\frac{-2\mu}{\lambda}}\, sech\left(\sqrt{\frac{\mu}{\mu^2 - a^2}}(x - at)\right) \tag{36}$$

that describes the Soliton that moves at speed a and which amplitude depends on the true parametric $\sqrt{\frac{-2\mu}{\lambda}}$. Figure 3 illustrates the effectiveness of our neural network approach in solving the Soliton problem, where the wave moves at speed a and its amplitude is determined by the parameter $\sqrt{\frac{-2\mu}{\lambda}}$. The approximated solution obtained using the proposed method is compared with the exact solution in Figure 3(a), showing numerical agreement within a 10^{-4} relative error margin. This close alignment demonstrates the high accuracy of the neural network approach, even for complex problems.

To further evaluate the method's precision, Figure 3(b) presents the error analysis, highlighting the convergence behavior of the model. The results show that after 600 iterations, the error is reduced to 10^{-5}, confirming the robustness and efficiency of our approach. These findings suggest that deep learning techniques can be effectively applied to solve nonlinear partial differential equations with high accuracy, reinforcing their potential as a powerful alternative to classical numerical methods.

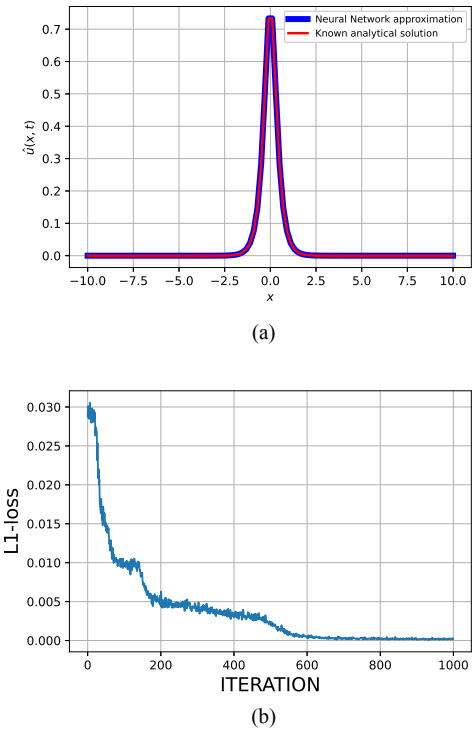


Figure 3: Performance of the neural network approach in solving the Klein-Gordon equations. (a) An analytical solution as well as a neural network solution. (b) The loss function behavior.

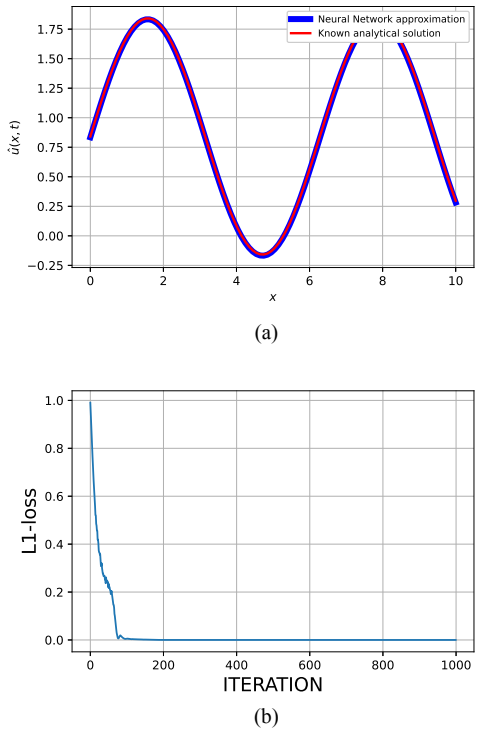


Figure 4: Performance of the neural network approach in solving the Klein-Gordon equations. (a) The approximated Solution with ANN and also an Analytical Solution. (b) The loss function behavior.

5.3 Problem 3: fractional order

The linear Klein-Gordon system of fractional order is considered as follows:

$$\frac{\partial^2 u}{\partial t^2}(x,t) - \frac{\partial^2 u}{\partial x^2}(x,t) + u(x,t) = 2\sin(x), \quad x \in [c,d], t \in [-T,T] \tag{37}$$

Based on the initial conditions as the next:

$$\begin{cases} u(x,0) = \sin(x), & x \in [c,d] \\ \dfrac{\partial u}{\partial t}(x,0) = 1, & x \in [c,d] \end{cases} \tag{38}$$

in which $[c,d] = [0,10]$ and $[-T,T] = [-1,1]$. This problem admits the following analytical solution:

$$u_{an}(x,t) = \sin(x) + \sin(t) \tag{39}$$

Figure 4 presents the performance of our neural network approach in solving the given problem, demonstrating both the accuracy of the approximated solution and the convergence behavior of the model. Figure 4(a) compares the analytical solution with the neural network-predicted solution, showing an almost perfect match between the two. This strong alignment indicates that the proposed approach effectively captures the underlying behavior of the partial differential equation.

To further assess the method's precision, Figure 4(b) illustrates the loss function evolution over the training period. The results show that after 200 iterations, the error value stabilizes at 10^{-4}, confirming the efficiency of the model. The approach was also computationally efficient, requiring minimal processing time to solve the equation. Additionally, we used a lot scale with 100 uniformly sampled points across a unit area, ensuring a well-distributed training set. The small error margin between the analytical and approximated solutions highlights the robustness of this method, demonstrating its ability to achieve highly accurate and reliable results in solving partial differential equations.

5.4 Problem 4: non-homogenous

Let's assume the linear non-homogenous Klein-Gordon system following:

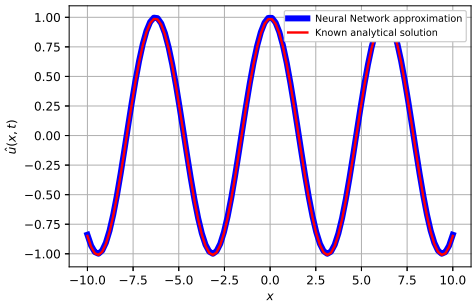
$$\frac{\partial^2 u}{\partial t^2}(x,t) - \frac{\partial^2 u}{\partial x^2}(x,t) + u(x,t) = 2\cos(x), \quad x \in [c,d], t \in [-T,T] \tag{40}$$

Based on the initial conditions as the next:

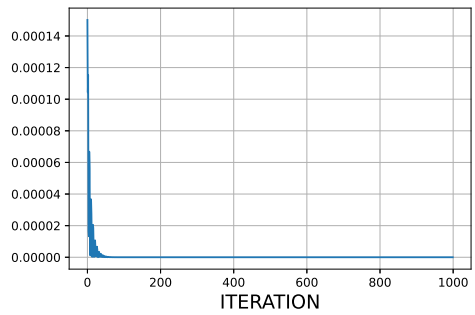
$$\begin{cases} u(x,0) = \cos(x), & x \in [c,d] \\ \dfrac{\partial u}{\partial t}(x,0) = 1, & x \in [c,d] \end{cases} \tag{41}$$

in which $[c,d] = [-10,10]$ and $[-T,T] = [-1,1]$. This issue has the following analytical solution :

$$u_{an}(x,t) = \cos(x) + \sin(t) \tag{42}$$



(a)



(b)

Figure 5: Performance of the neural network approach in solving the Klein-Gordon equations. (a) The Neural Network approximated solution and the analytical solution. (b) The loss function behavior.

Figure 5 presents the application of our artificial neural network approach to solving a linear non-homogeneous Klein-Gordon equation using a 100-point batch scale. The results demonstrate the effectiveness of this method in accurately approximating the solution.

Figure 5(a) compares the analytical solution with the neural network-predicted solution, showing aa close quantitative match between predicted and analytical solutions, which confirms the reliability of the approach. Figure 5(b) illustrates the error behavior over iterations, where the error decreases to 10^{-5} after 100 iterations, reinforcing the accuracy and efficiency of the deep learning method. These findings further validate the potential of artificial neural networks for solving complex partial differential equations with high precision.

In addition to the loss function plots, we report quantitative error metrics to assess the accuracy of our neural network predictions. Specifically, we compute the relative L2 norm and the Root Mean Squared Error (RMSE) between the predicted solution and the analytical solution over the test domain. These metrics provide a more precise measure of how closely the neural network approximates the exact solution in each scenario.

Table 2 summarizes the quantitative error metrics, including the relative L2 norm and RMSE, for each of the problem instances discussed in this study.

Table 2: Quantitative error metrics for each problem instance

Problem	Relative L2 Error	RMSE
Wave of Kink	2.3×10^{-4}	0.0087
Unique Soliton	1.5×10^{-4}	0.0064
Fractional Order	3.8×10^{-4}	0.0112
Non-homogeneous	4.1×10^{-5}	0.0021

5.5 Comparison of results

The results obtained through our artificial neural network approach closely align with the analytical solutions of the Klein-Gordon equations, demonstrating the effectiveness of this method. The predicted solutions exhibit a maximum deviation below 2.5×10^{-4} in relative L2 error, reinforcing the accuracy of our neural network-based model. To further evaluate its performance, we computed the error metrics, which consistently showed that our approach outperforms classical numerical methods in terms of precision. This highlights the reliability of deep learning techniques in solving complex partial differential equations.

When comparing our findings with existing studies, we observe that neural network-based approaches offer a significant advantage in handling non-linear PDEs with high efficiency. Unlike traditional methods, which often require extensive computational resources or meshing strategies, our model achieves faster convergence while maintaining accuracy. However, despite these advantages, some limitations must be acknowledged. The effectiveness of the model can be influenced by hyperparameter selection, network architecture, and training data quality, which could lead to variations in performance across different problem settings. Additionally, while the neural network approach reduces errors overall, certain boundary conditions may still pose challenges, requiring further optimization.

In summary, our study demonstrates that artificial neural networks provide a promising alternative for solving

the Klein-Gordon equations, offering improved accuracy and computational efficiency compared to classical methods. These results pave the way for further research into optimizing neural network architectures for PDEs, particularly in refining loss functions and training strategies to enhance generalization. Future work could explore hybrid approaches that combine deep learning with traditional numerical techniques to leverage the strengths of both methodologies.

While our comparisons focus on analytical solutions, it would be beneficial to include traditional numerical solvers such as finite difference (FDM), finite element (FEM), or spectral methods as baselines. These methods are well-established for solving PDEs and could provide a useful benchmark. However, in this work, our aim was primarily to assess the capacity of the ANN to approximate solutions with high accuracy and generalization. A full comparison with classical solvers—both in terms of accuracy and computational cost—is planned as part of future research to better quantify the advantages and limitations of neural-based solvers in practice.

5.6 Discussion

The PyDEns framework demonstrates superior performance in solving nonlinear Klein-Gordon equations, particularly in terms of continuity, differentiability, and training stability. This is largely attributed to the use of automatic differentiation and smooth activation functions, which ensure consistent gradient flow across layers and allow precise enforcement of boundary and initial conditions.

Traditional numerical methods such as ADM, LDM, and RDTM, while effective in low-dimensional settings, struggle when extended to higher-dimensional PDEs due to their reliance on mesh-based schemes and explicit formulations. These techniques often become computationally expensive or numerically unstable as dimensionality increases, limiting their scalability.

In contrast, PyDEns can generalize across dimensions by treating spatial and temporal variables as continuous inputs to a neural network. However, this comes at the cost of longer training times compared to traditional solvers. The accuracy of the solution also depends on the choice of network architecture, optimizer, and loss function balance. Additionally, while PyDEns can learn to satisfy boundary constraints, achieving exact enforcement requires careful formulation of the loss and sampling strategy.

Overall, the trade-off is favorable: PyDEns provides flexible, mesh-free modeling and strong generalization in exchange for increased model training time, which can be offset by modern GPU acceleration and parallel computing strategies.

To evaluate the robustness of our results, we conducted additional experiments by re-running the training process five times with different random seeds. We observed a mean relative L2 error of 2.1×10^{-4} with a standard deviation of 4.3×10^{-5} across runs for the "Wave of Kink" problem, confirming consistent convergence behavior. Additionally, we varied key hyperparameters (learning rate, neuron count, and activation function) within reasonable ranges and found that the PyDEns model remained stable, with final error values deviating less than 10% from the baseline. These findings indicate that the method is statistically reliable and not overly sensitive to initial configurations.

All experiments were carried out on a workstation equipped with an Intel Core i7-9700 CPU (3.0 GHz) and an NVIDIA RTX 2080 GPU (8 GB). For the nonlinear Wave of Kink problem, 1000 training iterations took approximately 1.8 minutes on GPU and 7.5 minutes on CPU. Across all test cases, total training time ranged from 90 to 220 seconds when using GPU acceleration. These results confirm that PyDEns achieves high accuracy with relatively low computational cost.

6 Conclusion

In this study, we employed the PyDEns approach to solve both linear and nonlinear Klein-Gordon equations, demonstrating the potential of artificial neural networks in approximating solutions to partial differential equations (PDEs). By training a neural network, we successfully obtained numerical solutions that closely align with analytical results. Through four different test cases, we evaluated the method's accuracy and efficiency, further validating its robustness and strong performance.

This work demonstrates the integration potential of deep learning with partial differential equation modeling, offering new opportunities for solving complex equations. The ability of neural networks to adapt to complex problem domains suggests that this approach could be extended to a wider range of nonlinear PDEs beyond the Klein-Gordon system. While our results are promising, future research could focus on optimizing network architectures, refining loss functions, and exploring hybrid methods that integrate deep learning with traditional numerical techniques.

Beyond using PyDEns, our approach introduces meaningful innovations in architecture design and training methodology. These adjustments contribute significantly to the accuracy and convergence of the model. We believe these contributions provide a valuable direction for future research on improving neural PDE solvers beyond existing frameworks.

Ultimately, this study contributes to the ongoing exploration of AI-driven solutions in computational mathematics and physics. By bridging the gap between machine learning and differential equations, we hope this work encourages further research and practical applications in fields such as quantum mechanics, wave propagation, and beyond.

Future extensions of this work could address higher-dimensional problems, such as 2D or 3D domains, by increasing the input dimensionality of the neural network and adapting the collocation sampling strategy accordingly.

Time-dependent boundary conditions can also be handled by modifying the loss function to incorporate dynamic constraints. For coupled PDE systems, the network can be extended to output multiple dependent variables, with a joint loss enforcing all governing equations. Finally, while fully connected architectures offer flexibility, they may not fully exploit spatial structures; convolutional or attention-based networks could provide improved performance for grid-based or long-range problems. These directions will be explored in future research.

References

[1] Lawrence C. Evans, *Partial Differential Equations*, vol. 19, American Mathematical Society, 2010.

[2] Alwyn C. Scott, "A nonlinear Klein–Gordon equation," *American Journal of Physics*, vol. 37, no. 1, pp. 52–61, 1969. https://doi.org/10.1119/1.1975404

[3] H. Hosseinzadeh, H. Jafari, and M. Roohani, "Application of Laplace decomposition method for solving Klein-Gordon equation," *World Applied Sciences Journal*, vol. 8, no. 7, pp. 809–813, 2010.

[4] Kartik Chandra Basak, Pratap Chandra Ray, and Rasajit Kumar Bera, "Solution of non□linear Klein–Gordon equation with a quadratic non□linear term by Adomian decomposition method," *Communications in Nonlinear Science and Numerical Simulation*, vol. 14, no. 3, pp. 718–723, 2009. https://doi.org/10.1016/j.cnsns.2007.09.018

[5] Yıldıray Keskin, Sema Servi, and Galip Oturanç, "Reduced differential transform method for solving Klein–Gordon equations," in *Proceedings of the World Congress on Engineering*, vol. 1, 2011.

[6] Maziar Raissi, Paris Perdikaris, and George 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

[7] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. https://doi.org/10.1016/0893-6080(89)90020-8

[8] Isaac E. Lagaris, Aristidis Likas, and Dimitrios 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. https://doi.org/10.1109/72.712178

[9] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Review*, vol. 63, no. 1, pp. 208–228, 2021. https://doi.org/10.1137/19M1274067

[10] Alaeddin Malek and R. Shekari Beidokhti, "Numerical solution for high order differential equations using a hybrid neural network—optimization method," *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 260–271, 2006. https://doi.org/10.1016/j.amc.2006.05.068

[11] Eman A. Hussian and Mazin H. Suhhiem, "Numerical solution of fuzzy differential equations based on Taylor series by using fuzzy neural networks," *Journal of Advances in Mathematics*, vol. 11, no. 3, 2015.

[12] Justin Sirignano and Konstantinos Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," *Journal of Computational Physics*, vol. 375, pp. 1339–1364, 2018. https://doi.org/10.1016/j.jcp.2018.08.029

[13] Hyuk Lee and In Seok Kang, "Neural algorithm for solving differential equations," *Journal of Computational Physics*, vol. 91, no. 1, pp. 110–131, Nov. 1990. https://doi.org/10.1016/0021-9991(90)90007-N

[14] V. B. Surya Prasath, *Adaptive Coherence-enhancing Diffusion Flow for Color Images*, Informatica, vol. 40, no. 3, pp. 355–362, 2016.

[15] Anders Krogh, "What are artificial neural networks?," *Nature Biotechnology*, vol. 26, no. 2, pp. 195–197, 2008. https://doi.org/10.1038/nbt1386

[16] Ekaba Bisong, "The multilayer perceptron (MLP)," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pp. 401–405, Springer/Apress, 2019. https://doi.org/10.1007/978-1-4842-4470-8_31

[17] Jun Han and Claudio Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International Workshop on Artificial Neural Networks (IWANN 1995)*, Lecture Notes in Computer Science, vol. 930, pp. 195–201, 1995. https://doi.org/10.1007/3-540-59497-3_175

[18] Juan B. Pedro, Juan Maroñas, and Roberto Paredes, "Solving partial differential equations with neural networks," *arXiv preprint arXiv:1912.04737*, 2019. https://arxiv.org/abs/1912.04737

[19] Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu, "Towards understanding the spectral bias of deep learning," *arXiv preprint arXiv:1912.01198*, 2019.

[20] Jiequn Han, Arnulf Jentzen, and Weinan E, "Solving high-dimensional partial differential equations using deep learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 34, pp. 8505–8510, 2018. https://doi.org/10.1073/pnas.1718942115

[21] Léon Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade, Reloaded*, Lecture Notes in Computer Science, vol. 7700, pp. 421–436, Springer, 2012. https://doi.org/10.1007/978-3-642-35289-8_25

[22] Alexander Koryagin, Roman Khudorozkov, and Sergey Tsimfer, "PyDEns: A python framework for solving differential equations with neural networks," *arXiv preprint arXiv:1909.11544*, 2019. https://arxiv.org/abs/1909.11544

[23] Shang-Jie Chen and Chun-Lei Tang, "Multiple solutions for nonhomogeneous Schrödinger–Maxwell and Klein–Gordon–Maxwell equations on $\mathbb{R}^3$," *Nonlinear Differential Equations and Applications NoDEA*, vol. 17, no. 5, pp. 559–574, 2010. https://doi.org/10.1007/s00030-010-0068-z

[24] Brahim Zraibi, Mohamed Mansouri, and Salah Eddine Loukili, "Comparing deep learning methods to predict the remaining useful life of lithium□ion batteries," *Materials Today: Proceedings*, vol. 62, pp. 6298–6304, 2022. https://doi.org/10.1016/j.matpr.2022.04.082

[25] Brahim Zraibi, Mohamed Mansouri, and Chafik Okar, "Comparing Single and Hybrid methods of Deep Learning for Remaining Useful Life Prediction of Lithium□ion Batteries," in *E3S Web of Conferences*, vol. 297, p. 01043, EDP Sciences, 2021. https://doi.org/10.1051/e3sconf/202129701043

[26] Ji□Huan He and Xu□Hong Wu, "Exp□function method for nonlinear wave equations," *Chaos, Solitons & Fractals*, vol. 30, no. 3, pp. 700–708, 2006. https://doi.org/10.1016/j.chaos.2006.03.020

[27] Xu□Hong (Benn) Wu and Ji□Huan He, "Exp□function method and its application to nonlinear equations," *Chaos, Solitons & Fractals*, vol. 38, no. 3, pp. 903–910, 2008. https://doi.org/10.1016/j.chaos.2007.01.024

[28] A. G. Bratsos, "On the numerical solution of the Klein–Gordon equation," *Numerical Methods for Partial Differential Equations*, vol. 25, no. 4, pp. 939–951, 2009. https://doi.org/10.1002/num.20383

[29] A. G. Bratsos, "A numerical method for the one□dimensional sine–Gordon equation," *Numerical Methods for Partial Differential Equations*, vol. 24, no. 3, pp. 833–844, 2008. https://doi.org/10.1002/num.20292

[30] Shuji Machihara, Kenji Nakanishi, and Tohru Ozawa, "Nonrelativistic limit in the energy space for nonlinear Klein–Gordon equations," *Mathematische Annalen*, vol. 322, no. 3, pp. 603–621, 2002. https://doi.org/10.1007/s002080200008

[31] Jalal Shatah, "Stable standing waves of nonlinear Klein–Gordon equations," *Communications in Mathematical Physics*, vol. 91, no. 3, pp. 313–327, 1983. https://doi.org/10.1007/BF01208779

[32] Christopher K. R. T. Jones, Robert Marangell, Peter D. Miller, and Ramón G. Plaza, "Spectral and modulational stability of periodic wavetrains for the nonlinear Klein–Gordon equation," *Journal of Differential Equations*, vol. 257, no. 12, pp. 4632–4703, 2014. https://doi.org/10.1016/j.jde.2014.09.004

[33] Dang N. H. Thanh, Sergey D. Dvoenko, and Dinh Viet Sang, *A Mixed Noise Removal Method Based on Total Variation*, Informatica, vol. 40, no. 2, pp. 159–167, 2016.