Physics-Informed Neural Networks for Quantum Wavefunctions

Istiak Mahmud*, Ayush Asthana†, Mark Hoffmann†, Ahmed Abdelhadi*
*School of Electrical Engineering & Computer Science, University of North Dakota, ND, USA
†Department of Chemistry, University of North Dakota, ND, USA

***Abstract*—The wave functions and permitted energy levels of quantum systems, which characterize the probability distributions and particle behaviors, are able to be determined via the time-independent Schrödinger equation. Our research study introduces an experimental method of solving the time-independent Schrödinger equation by utilizing Physics-Informed Neural Networks (PINNs) with a customized loss function designed for output waveshape prediction. The technique makes use of PINNs' capacity to directly integrate physical rules into the learning process, assuring that the solutions follow the guiding principles of quantum mechanics. We forecast the associated wavefunctions for different energy levels (n states) and effectively solve the Schrödinger equation for distinct quantum states by using PINNs and a customized loss function. Our findings show that PINNs are a reliable and accurate way to represent the fine features of quantum wavefunctions, and they present a viable substitute for more conventional numerical techniques. This methodology not only improves computer performance but also offers more profound understanding of the behavior of systems in quantum mechanics, which may have implications for material science, nanotechnology, and quantum computing.**

***Index Terms*—PINN, Wave Function, Loss Function, Quantum**

I. Introduction

The time-independent Schrödinger equation is a vital source of information about the behavior of quantum systems in stationary states [1]. Its manipulation can lead to significant scientific and technological advances in fields such as quantum computing, material science, and nanotechnology [2], as it is fundamental for decoding the quantum properties of particles and systems [3]. However, traditional numerical methods, while effective, can be computationally expensive and struggle with complex systems [4]. Recently, machine learning methods, particularly neural networks, have shown great potential in solving complex problems in physics [5], [6]. This led to the development of physics-informed neural networks (PINNs) [7], which incorporate physical laws during training to ensure solutions are consistent with the underlying physics [8]. These integrated solutions are not only more accurate but also computationally faster. In this paper, we propose a novel approach to solving the time-independent Schrödinger equation using the PINNs structure. Our approach employs a custom loss function tailored for output waveshape predictions, which predicts the wavefunctions of different energy levels (n states). By leveraging the advantages of PINNs, we aim to provide accurate solutions that account for the fine structures of quantum wavefunctions [9]. Our research begins by describing the fundamentals of PINNs and the theoretical underpinnings of the time-independent Schrödinger equation [10]. We then explain our custom loss function and how it improves wavefunction predictions. The remainder of the paper showcases a series of experiments demonstrating the efficacy of our approach across various quantum states. These findings highlight the potential of PINNs in tackling challenging quantum mechanical problems, offering a powerful new tool to enhance existing numerical techniques. This work demonstrates how PINNs can revolutionize the solving of fundamental equations in physics, paving the way for new research and development directions in applying machine learning to quantum mechanics. Our approach aims to resolve computational challenges and ensure researchers can find the most accurate and efficient solutions to the Schrödinger equation, ultimately advancing the study and development of quantum technologies.

A. Related Work

Xiaotian Jiang et al. [11] simulate the nonlinear dynamics in fiber optics by solving the nonlinear Schrödinger equation (NLSE) using PINNs. Their work highlights the superiority of PINNs over classical algorithms in well-understood optical problems, both in accuracy and speed. They emphasize the importance of including physical parameters as inputs to generalize the model across various settings. However, issues with enforcing different initial and boundary conditions and scaling PINNs to more complex problems remain open. Chaojun Zhang and Yuexing Bai [12] present an improved physics-informed neural network (IPINN) technique to solve the defocusing nonlinear Schrödinger (NLS) equation with time-dependent potential. Their IPINN method can find the exact rogue wave solution and outperforms conventional PINN schemes in accuracy and convergence. They analyze the impact of network layers, neurons, and sample points on performance, demonstrating the IPINN algorithm's capability to solve complex nonlinear differential equations in various scientific areas. However, the practical application of the IPINN algorithm to high-order differential algebraic equations (DAEs) and tight side conditions, as well as integrating more system properties, remains unexplored. Karan Shah et al. [13] explore the application of PINNs to solve the non-relativistic, time-dependent Schrödinger equation (TDSE). Their focus is on the performance and generalizability of PINN solvers across different system parameters, domains, and energy states, using a quantum harmonic oscillator example. The results indicate that PINNs can accurately reproduce quantum system dynamics and offer advantages over conventional numerical solvers, such as mesh-free simulations and grid resolution independence. However, the paper does not fully address the challenges of learning PINNs for high-frequency solutions or larger time domains, and the implementation details for more complex quantum systems, such as interacting qubits, are ambiguous. Liam Harcombe et al. [14] introduce a new methodology based on PINNs to uncover localized states in disordered media, focusing on 1D cases of spectra estimates of Hamiltonians with randomly generated potentials. They propose a novel loss function incorporating physical information to scan the domain for eigenstates with similar eigenenergies. Through several examples, the paper demonstrates the method's effectiveness and compares it with isogeometric analysis (IGA). However, it is superficial in discussing how to extend the approach to high-dimensional systems and proving theoretical convergence rates. The understanding of how more complicated or nonlinear differential equations are learned by the PINN model remains somewhat unclear.

The rest of the paper is structured as follows: Problem statement is discussed in Section II. The methodology is explained in Section III. Section IV exhibits the result analysis. Section V includes our discussion. Finally, Section VI exhibits the paper's conclusion.

II. Problem Statement

The time-independent Schrödinger equation is a fundamental component of quantum mechanics, providing critical insights into the behavior of quantum systems in stationary states. Traditional numerical methods used to solve this equation, while effective, are computationally intensive and often struggle with the complexity of advanced quantum systems. Recent advancements in machine learning, particularly Physics-Informed Neural Networks (PINNs), have demonstrated potential in overcoming these challenges by incorporating physical laws directly into the training process. However, there remains a need for methods that can efficiently and accurately predict the wavefunctions of quantum systems across various states.

A. The time-independent Schrödinger equation (Harmonic Oscillator)

The time-independent Schrödinger equation (TISE) for a harmonic oscillator is a cornerstone of quantum mechanics [15], describing systems with a quadratic potential. This potential, given by equation (1).

$$V(x) = \frac{1}{2} m\omega^2 x^2 \tag{1}$$

Equation 1 represents a particle of mass m oscillating with angular frequency ω around an equilibrium position. The TISE equation itself is expressed as equation (2).

$$-\frac{\hbar^2}{2m}\frac{d^2\psi(x)}{dx^2} + \frac{1}{2} m\omega^2 x^2 \psi(x) = E\psi(x) \tag{2}$$

where $\hbar$ is the reduced Planck constant, $\psi(x)$ is the wavefunction, and E is the energy of the particle.

Solving this equation reveals that the energy levels are quantized, given by equation (3).

$$E_n = \left(n + \frac{1}{2}\right)\hbar\omega \tag{3}$$

where n is a non-negative integer. These discrete energy levels illustrate the quantum nature of the harmonic oscillator, with the lowest energy state (ground state) having an energy of $\frac{1}{2}\hbar\omega$ due to the zero-point energy. The corresponding wavefunctions $\psi_n(x)$ are described by Hermite polynomials and Gaussian functions, ensuring orthogonality and completeness, meaning they form a basis for any function within the system's space.

This model is relevant since it is applicable in many physical systems: from vibrating atoms in a molecule to quantized modes of electromagnetic fields in a cavity [16]. The analytic solution of the harmonic oscillator illustrates how omnipresent quantum mechanics is in controlling the behavior of particles in any potential well (zero-point energy), signaling the Heisenberg uncertainty [17] that the position and momentum of particles are inherently indefinite.

B. Boundary Conditions

The wave function at the borders approaches zero as x approaches either positive or negative boundary value. This ensures that the wave function is well-behaved and is a popular choice for boundary conditions in quantum mechanics. Let, boundary value in x-axis is a and $-a$, then the boundary condition will be $\psi(-a) = 0$ and $\psi(a) = 0$.

III. Methodology

This section outlines the step-by-step methods used in this approach.

A. Neural Networks

A neural network can be fundamentally described as a mathematical model designed to approximate a function through a set of parameters that can be learned. Formally, the network is represented as

$$u(x, \theta) : \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_u},$$

where x denotes the input to the network, θ represents the set of parameters to be optimized, and d_x, d_θ, and d_u correspond to the dimensionalities of the network's inputs, parameters, and outputs, respectively. In a typical supervised learning scenario, the goal is to adjust the parameters θ by training the network on a dataset containing input-output pairs. The training process involves minimizing a loss function that penalizes discrepancies between the network's predictions and the actual data, leading to parameter updates [18].

The specific structure of the network function is determined by the architecture of the neural network. In this work, we focus exclusively on feedforward fully connected networks (FCNs). The network function is defined as follows:

$$u(x, \theta) = f_n \circ \ldots \circ f_i \circ \ldots \circ f_1(x, \theta) \tag{4}$$

where $x \in \mathbb{R}^{d_x}$ is the input to the FCN, $u \in \mathbb{R}^{d_n}$ is the output, and n represents the number of layers (or depth) in the FCN. Each layer $f_i(x, \theta)$ is defined by:

$$f_i(x, \theta) = \sigma_i(W_i x + b_i) \tag{5}$$

Here, $\theta_i = (W_i, b_i)$ denotes the learnable parameters for the i-th layer, where $W_i \in \mathbb{R}^{d_i \times d_{i-1}}$ are the weight matrices and $b_i \in \mathbb{R}^{d_i}$ are the bias vectors. The activation functions σ_i, which introduce nonlinearity into the network, are typically nonlinear functions such as the rectified linear unit (ReLU) or the hyperbolic tangent function, which is the specific choice in our case.

B. Physics-Informed Neural Networks (PINNs)

Physics-informed neural networks (PINNs) [19] are utilized to address differential equation problems, specifically focusing on boundary value problems of the form:

$$\mathcal{D}[u](x) = f(x), \quad x \in \Omega \subseteq \mathbb{R}^d, \tag{6}$$

$$\mathcal{B}_k[u](x) = g_k(x), \quad x \in \Gamma_k \subseteq \partial\Omega, \tag{7}$$

where $\mathcal{D}[u](x)$ represents a differential operator, and $u(x)$ is the function to be determined. The conditions $\mathcal{B}_k(\cdot)$ enforce boundary conditions (BCs) that ensure the solution's uniqueness. Here, BCs are considered in a broad sense, without explicit distinction between initial and boundary conditions. The variable x can span time as well. Equations (6) and (7) cover various differential equation problems, including linear and nonlinear types, as well as time-dependent and independent cases, and those with irregular, higher-order, and cyclic boundary conditions. [20]

To resolve these equations, PINNs utilize a neural network to approximate the solution directly, denoted as $u(x, \theta) \approx u(x)$, where θ represents the neural network's parameters. For simplicity, the same notation is used for both the true solution and the neural network's approximation throughout this discussion. It is critical to highlight that PINNs approximate the solution functionally rather than providing a discretized solution, distinguishing them from traditional methods such as finite difference methods. As such, PINNs act as a mesh-free method to solve differential equations. [21]

The training process for the PINN involves minimizing the following loss function:

$$\mathcal{L}_p(\theta) = \frac{\lambda_p}{N_p} \sum_{i=1}^{N_p} \left(\mathcal{D}[u(x_i, \theta)] - f(x_i)\right)^2, \tag{8}$$

$$\mathcal{L}_b(\theta) = \sum_{k=1}^{N_k} \frac{\lambda_b^k}{N_b^k} \sum_{j=1}^{N_b^k} \left(\mathcal{B}_k[u(x_j^k, \theta)] - g_k(x_j^k)\right)^2, \tag{9}$$

$$\begin{aligned} \mathcal{L}(\theta) = {} & \frac{\lambda_P}{N_P} \sum_{i=1}^{N_P} \left(\mathcal{D}[u(x_i, \theta)] - f(x_i)\right)^2 \\ & + \sum_{k=1}^{N_k} \frac{\lambda_b^k}{N_b^k} \sum_{j=1}^{N_b^k} \left(\mathcal{B}_k[u(x_j^k, \theta)] - g_k(x_j^k)\right)^2 \end{aligned} \tag{10}$$

where $\{x_i\}_{i=1}^{N_p}$ denotes a set of collocation points within the domain, and $\{x_j^k\}_{j=1}^{N_b^k}$ represents points along each boundary condition. The terms λ_p and λ_b^k are scalars that balance the different components of the loss function. The idea is that by minimizing the PDE residual, the network learns a solution that satisfies the governing PDE, and by minimizing the BC residual, it ensures that the solution conforms to the boundary conditions, leading to a unique solution. It is essential to have an adequate number of collocation and boundary points to ensure that the PINN can accurately learn a consistent solution across the domain.

Optimization of this loss function is generally performed using gradient-based methods, such as the Adam optimizer [22] or more sophisticated quasi-Newton methods like the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [23]. These techniques require calculating the gradient of the loss function with respect to the network parameters, which can be efficiently accomplished using automatic differentiation [24] provided by modern deep learning libraries [25]–[27]. Typically, the gradients of the network output with respect to its inputs are also necessary to evaluate the PDE residual in the loss function, which can then be used to adjust the network's parameters via automatic differentiation. [28]

C. Neural Network architecture for this research

The neural network learns complex functions, so it takes spatial coordinates as input and outputs wave function values [19]. Through training, the network can adjust itself to the training data for which labels are given, while enforcing the physics defined by the Schrödinger equation. This dual-purpose strategy allows the network to provide a physical law–informed solution to the quantum system, in which machine learning is combined with the building blocks of classical and quantum physics.

This research initializes an object of the FCN class which is written to build a simple fully connected neural network with a specific architecture [29] as shown in Figure 2. A single input feature and a single output feature. Four hidden layers with 64 neurons, having the activation function as Tanh after each layer. To represent the neural network architecture shown in Figure 2 with equations, we'll break down the feedforward process layer by layer. Here input layer is presented in equation (11), first hidden layer in equation (12), second hidden layer in equation (13), third hidden layer in equation (14), fourth hidden layer in equation (15) and output layer in equation (16).

$$f_0 = x \tag{11}$$

$$f_1 = \sigma(W_1 f_0 + b_1) \tag{12}$$

$$f_2 = \sigma(W_2 f_1 + b_2) \tag{13}$$

$$f_3 = \sigma(W_3 f_2 + b_3) \tag{14}$$

$$f_4 = \sigma(W_4 f_3 + b_4) \tag{15}$$

$$\psi = W_{out} f_4 + b_{out} \tag{16}$$

where W is the weight matrix and b is the bias vector.

Combining the Equations: The network function $\psi(x,\theta)$ can be represented as the composition of these layers:

$$\psi = W_{out}\sigma(W_4\sigma(W_3\sigma(W_2\sigma(W_1x+b_1)+b_2)+b_3)+b_4)+b_{out} \tag{17}$$

D. PINN for Schrödinger equation

Overall, solving the Schrödinger equation using a PINN approach involves formulating an appropriate loss function that incorporates the Schrödinger equation and the necessary boundary condition, and then optimizing this loss function to find the solution.

In PINN, there are typically two types of loss functions: Boundary Loss and physics loss.

1) Boundary Loss ($L_{Boundary}(x)$): To derive the boundary loss equation explicitly including the boundary values $x = -a$ and $x = a$, this expands the mean squared error calculation step. Here, a is the boundary x value. $\psi(-a) = 0$ and $\psi(a) = 0$. The boundary loss equation is given by:

$$L_{Boundary}(x) = \lambda_B \cdot \frac{1}{2}\left[(\psi_{\text{NN}}(-a))^2 + (\psi_{\text{NN}}(a))^2\right] \tag{18}$$

2) Physics Loss: This loss term ensures that the predictions of the neural network satisfy the underlying physics equations. It is formulated based on the residuals of the governing equations and is typically imposed at all points in the domain, not just the data points. For the Schrödinger's harmonic oscillator example, the physics loss would be based on the residual of the Schrödinger equation at all points in the domain. Equation (19) represents the physics loss for Schrödinger's harmonic oscillator example. Equation (19) is following the equation (2).

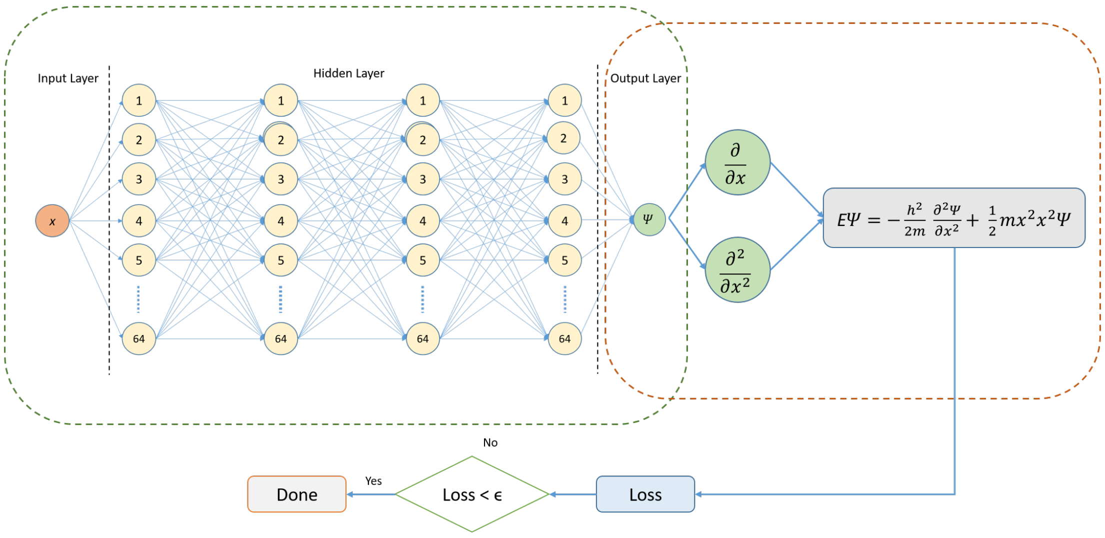

Input Layer
Hidden Layer
Output Layer
x
Ψ
$\frac{\partial}{\partial x}$
$\frac{\partial^2}{\partial x^2}$
$E\Psi = -\frac{\hbar^2}{2m}\frac{\partial^2\Psi}{\partial x^2} + \frac{1}{2}mx^2x^2\Psi$
Loss
Loss < ϵ
No
Yes
Done

Fig. 1: Flow diagram of the proposed methodology.

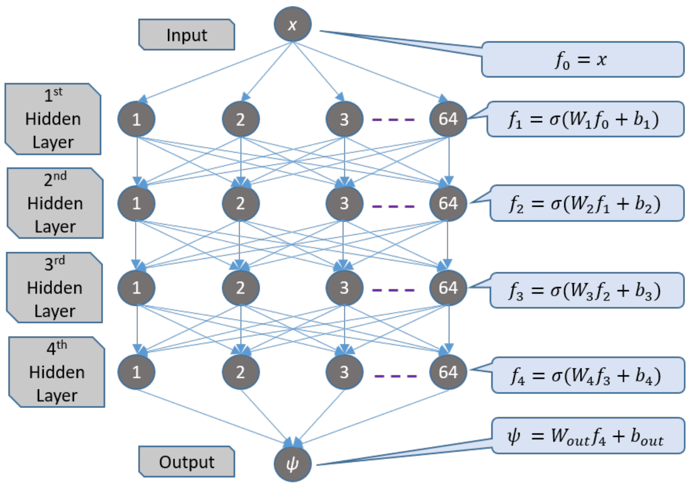

Input
x
$f_0 = x$
1st Hidden Layer
2nd Hidden Layer
3rd Hidden Layer
4th Hidden Layer
$f_1 = \sigma(W_1 f_0 + b_1)$
$f_2 = \sigma(W_2 f_1 + b_2)$
$f_3 = \sigma(W_3 f_2 + b_3)$
$f_4 = \sigma(W_4 f_3 + b_4)$
$\psi = W_{out} f_4 + b_{out}$
Output
ψ

Fig. 2: Neural Network Architecture

$$L_{Physics}(x) = \lambda_p \cdot \frac{1}{N_p} \sum_{i=1}^{N_p} \left| -\frac{\hbar^2}{2m} \mathcal{D}^2[\psi_{\text{NN}}(x_i)] + \frac{1}{2} m\omega^2 x_i^2 \psi_{\text{NN}}(x_i) - E\psi_{\text{NN}}(x_i) \right|^2 \quad (19)$$

Here, N is the number of points in the domain, x_i are the points in the domain, and λ is a hyperparameter that balances the importance of the boundary loss versus the physics loss.

3) Total Loss Function: During training, the PINN incorporates the Schrödinger equation directly into its loss function. This loss function comprises two main components: the physics loss and the boundary loss.

Based on the preceding discussion, the complete loss function for the time-independent Schrödinger equation is presented in equation (22).

$$Total\ Loss = Boundary\ Loss + Physics\ Loss \quad (20)$$

$$L_{Total}(x) = L_{Boundary}(x) + L_{Physics}(x) \quad (21)$$

$$L_{Total}(x) = \lambda_B \cdot \frac{1}{2} \left[(\psi_{\text{NN}}(-a))^2 + (\psi_{\text{NN}}(a))^2 \right] + \lambda_p \cdot \frac{1}{N_p} \sum_{i=1}^{N_p} \left| -\frac{\hbar^2}{2m} \mathcal{D}^2[\psi_{\text{NN}}(x_i)] + \frac{1}{2} m\omega^2 x_i^2 \psi_{\text{NN}}(x_i) - E\psi_{\text{NN}}(x_i) \right|^2 \quad (22)$$

Algorithm 1 Training Loop for Schrödinger Equation

```
1:  Initialization:
2:    $pinn = fcn(nn.module)$
3:    $B_k : \psi(a) = 0, \psi(-a) = 0$
4:  for loop do
5:      $L_{\text{Boundary}} = \frac{\lambda_B}{2} \left[ (\psi_{\text{NN}}(-a))^2 + (\psi_{\text{NN}}(a))^2 \right]$
6:      $L_{\text{Physics}} =$
        $\frac{\lambda_p}{N} \sum_{i=1}^{N} \left| -\frac{\hbar^2 d^2\psi}{2mdx^2} + \psi \left[ \frac{1}{2} m\omega^2 x^2 - E \right] \right|^2$
7:      $Total\_loss = L_{Boundary} + L_{Physics}$
8:      $Total\_loss.backward()$
9:      $optimiser = adam(pinn.parameters(), l_r)$
10: end for
```

This algorithm (1) trains a Physics-Informed Neural Network (PINN) to solve the Schrödinger equation.

Initially, the PINN is created as a fully connected neural network (NN), and boundary conditions (denoted as B_k) are set to ensure that the wavefunction ψ equals zero at the domain boundaries. In each iteration of the training loop, two losses are computed: the boundary loss L_{Boundary}, which enforces these boundary conditions, and the physics loss L_{Physics}, which ensures that the neural network's output adheres to the Schrödinger equation. The total loss, a combination of these two, is minimized using backpropagation, with gradients calculated for the neural network parameters. The ADAM optimizer, an algorithm that adjusts the learning rate (l_r) dynamically, updates the network parameters based on these gradients. This process continues in a loop until the total loss is minimized, ensuring that the network accurately solves the Schrödinger equation while respecting both boundary conditions and physical laws.

IV. Result Analysis

To predict the output wavefunction for the variable n state, we employed the Physics-Informed Neural Network (PINN) method with a custom loss function, implementing it in a Python 3.8 [30] with Pytorch environment [31].

Figure 3 features a series of subplots depicting the predicted wavefunctions for different quantum states (denoted by n) at various training steps. The subplots in Figure 3 are arranged in a grid format, where each column corresponds to a specific quantum state (n), and each row represents different training steps. Specifically, the first column illustrates the wavefunctions for $n = 0$, the second column for $n = 1$ and the third column for $n = 2$.

For $n = 0$ quantum state at Figure 3, the topmost subplot (row 1) shows the predicted wavefunction at the initial training step (0). The next subplot (row 2) shows the wavefunction after 100 training steps, and the downmost subplot (row 7) shows the wavefunction after 5000 training steps. The progression of the wavefunctions across the rows illustrates how the Physics-Informed Neural Networks (PINNs) refine their predictions with more training.

As training progresses in Figure 3, the wavefunctions become more accurate and detailed. The solid green line in each subplot represents the PINN-predicted wavefunction. This visual representation highlights the efficacy of the PINNs in learning and predicting the quantum wavefunctions accurately over different training steps.

Figure 4, displays a series of subplots that illustrate the training and loss curves for a Physics-Informed Neural Network (PINN) at different quantum states ($n = 0, 1, 2$). The figure is organized into three rows, each focusing on a different aspect of the training and loss processes: Boundary Loss, Physics Loss, and total Loss.

The first row of subplots (a, b and c) displays the boundary loss curves for $n = 0, 1, 2$ respectively. The x-axis represents the training steps, and the y-axis shows the boundary loss. Similar to the training loss, the boundary loss decreases quickly initially and stabilizes thereafter. Each subplot indicates that the boundary loss reduces significantly within the first few training steps and remains minimal for the remainder of the training.

The second row of subplots (d, e and f) illustrates the physics loss curves for $n = 0, 1, 2$ respectively. The x-axis represents the training steps, and the y-axis shows the physics loss. Like the previous losses, the physics loss drops sharply in the initial training phase and then levels off, indicating effective training.

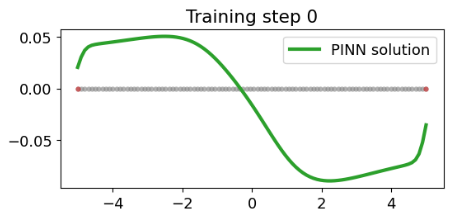
The 3rd row of subplots (g, h and i) shows the total loss curves for $n = 0, 1, 2$ respectively. The x-axis represents the training steps, ranging from 0 to 8000, while the y-axis shows the total training loss. All plots demonstrate a rapid decrease in training loss within the first few steps, followed by a plateau indicating stabilization.

Overall, the plots in Figure 4 collectively show how the boundary loss, physics loss and total loss evolve during the training process for different quantum states. Across all subplots, the rapid initial decrease in loss values followed by stabilization suggests that the PINN is effectively learning and minimizing errors. The consistent behavior across different quantum states indicates robustness in the training approach used for the PINN.

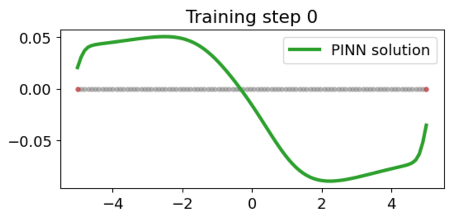
Parameter	**Value**
Number of Inputs	1
Number of Outputs	1
Number of Hidden Units	64
Number of Hidden Layers	4
Activation Function	Tanh
Learning Rate	0.0001
Boundary Conditions	x= (-5,5)
Boundary Condition Value	0
$\lambda_{Physics}$	0.0001
$\lambda_{Boundary}$	0.0001

TABLE I: Training Parameters

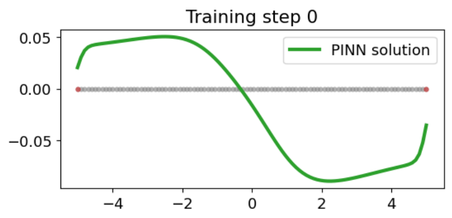
Table I provides a summary of the key hyperparameters and configurations used in training the neural network model. The model is designed with a single input feature and produces a single output feature, in-



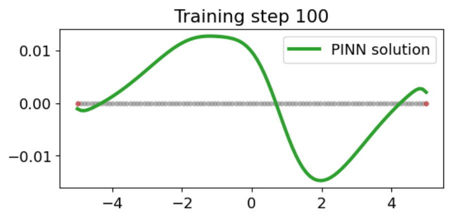
(a) n=0



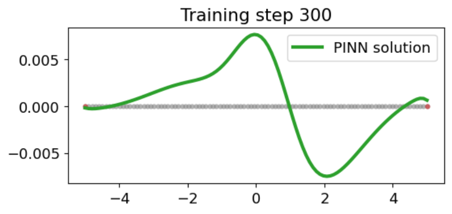
(b) n=1



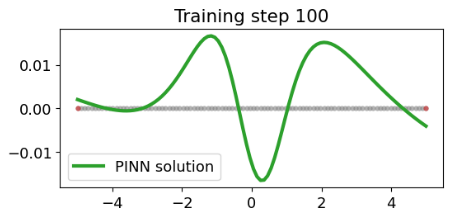
(c) n=2



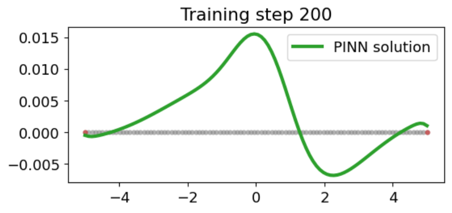
(d) n=0



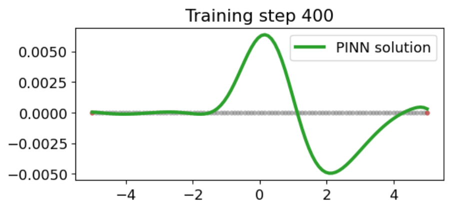
(e) n=1



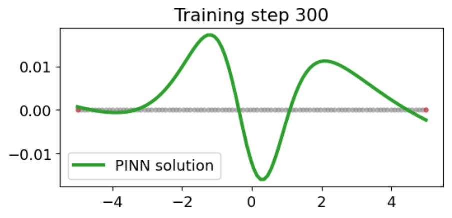
(f) n=2



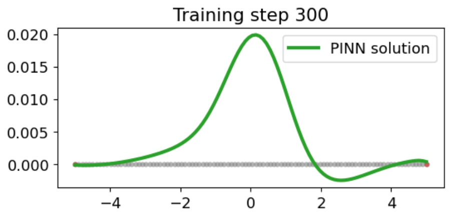
(g) n=0



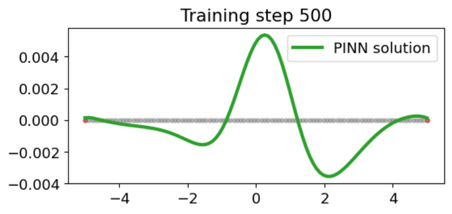
(h) n=1



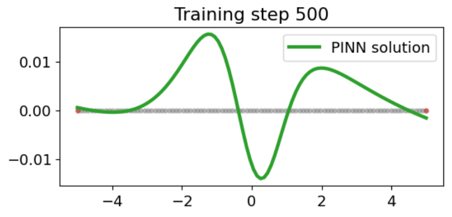
(i) n=2



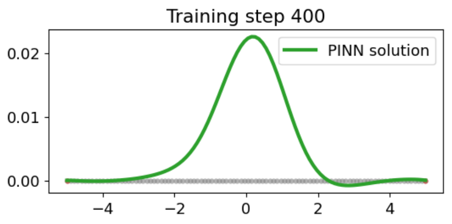
(j) n=0



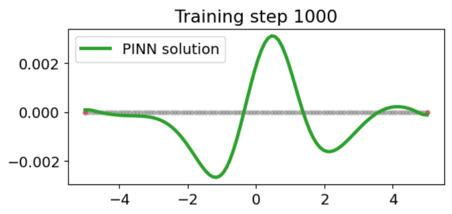
(k) n=1



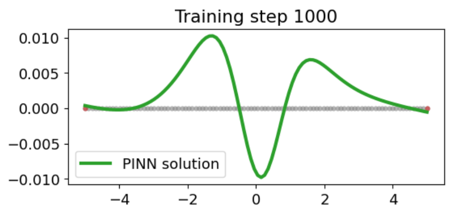
(l) n=2



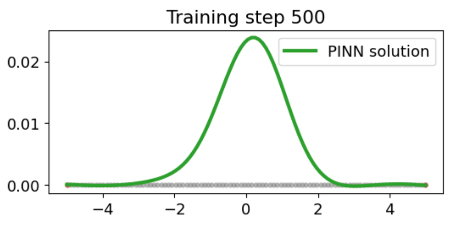
(m) n=0



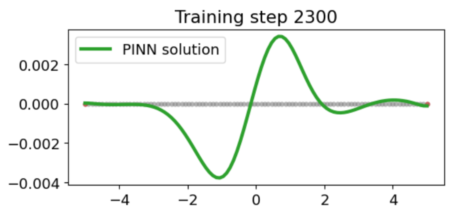
(n) n=1



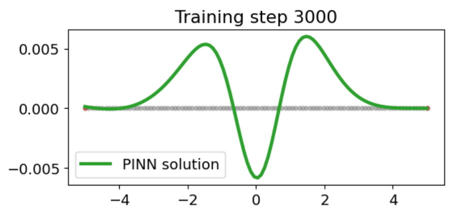
(o) n=2



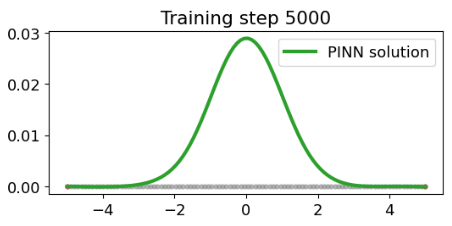
(p) n=0



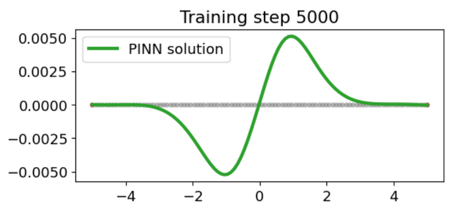
(q) n=1



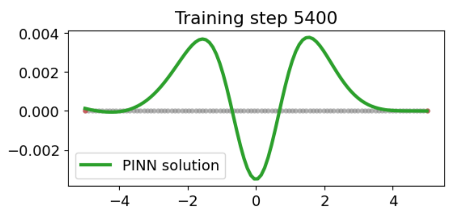
(r) n=2



(s) n=0

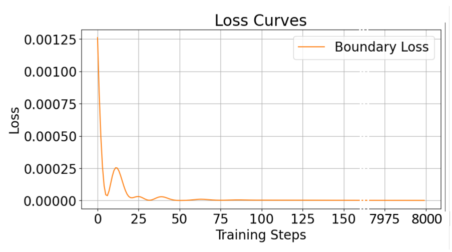


(t) n=1

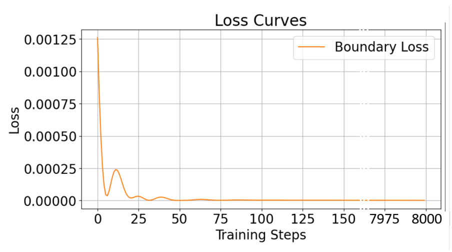


(u) n=2

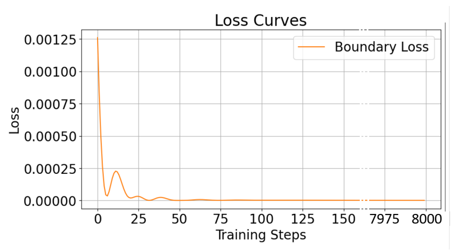
Fig. 3: Predicted wave-function for different n state and different training steps.



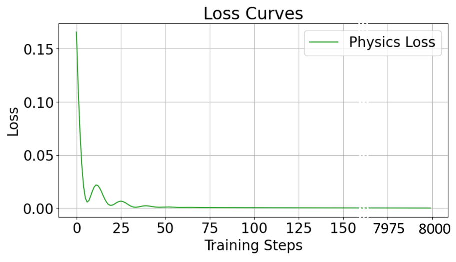
(a) n=0



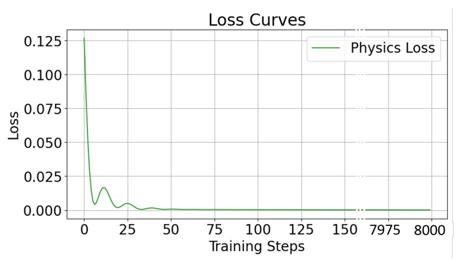
(b) n=1



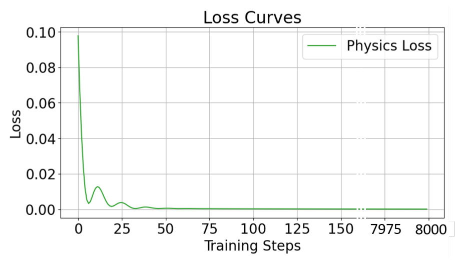
(c) n=2



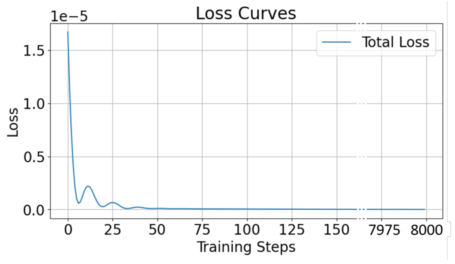
(d) n=0



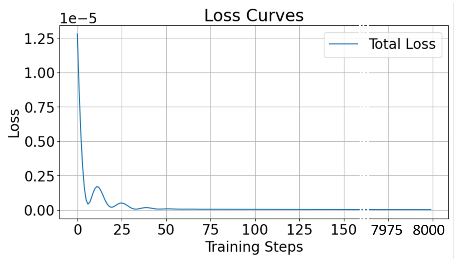
(e) n=1



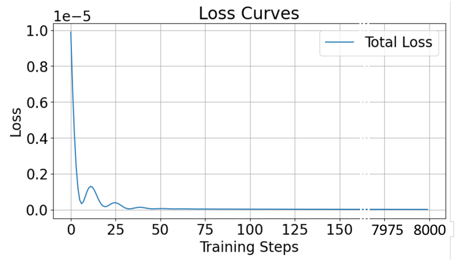
(f) n=2



(g) n=0



(h) n=1



(i) n=2

Fig. 4: Loss curves for different n states

dicating it is structured for a simple one-dimensional input-output relationship. The architecture of the neural network includes four hidden layers, each containing 64 units (neurons), which allows the model to learn complex patterns in the data.

In table I, the activation function used for the neurons in these hidden layers is the $Tanh$ (hyperbolic tangent) function, which introduces non-linearity into the model and helps in capturing intricate relationships within the data. The learning rate, a critical hyperparameter that controls the step size during the optimization process, is set to 0.0001. This relatively low learning rate helps in fine-tuning the model gradually, reducing the risk of overshooting the optimal solution. The model training also incorporates boundary conditions applied at $x = -5$ and $x = 5$, with the function value at these boundaries set to 0. These boundary conditions are essential for ensuring the model's predictions adhere to known constraints. Additionally, the loss function includes regularization terms weighted by $\lambda_{Physics}$ and $\lambda_{Boundary}$, both set to 0.0001. $\lambda_{Boundary}$ regulates the contribution of the boundary loss, while $\lambda_{Physics}$ controls the influence of the physics loss derived from the Schrödinger equation, ensuring that the model not only fits the data but also respects the underlying physical laws. These parameters collectively define the neural network's architecture and the conditions under which it is trained, crucial for replicating the results and understanding the model's behavior.

The table II provides a detailed account of the training and validation loss values for a Physics-Informed Neural Network (PINN) at various epochs $(0, 1000, 2000, 3000, 4000, 5000, 6000, 7000, \text{ and } 8000)$ for different quantum states $(n = 0, 1, 2)$. The table is organized with columns representing the different quantum states and sub-columns distinguishing between training and validation loss values. For each quantum state, the table records the initial loss values at epoch 0 and their progressive reduction through to epoch 8000.

For instance, at $n = 0$, the training loss starts at 1.668310×10^{-5} and the validation loss at 1.655692×10^{-1}, both of which decrease to zero by epoch 8000.

Epoch	$n=0$		$n=1$		$n=2$	
	Training Loss	Validation Loss	Training Loss	Validation Loss	Training Loss	Validation Loss
0	1.668310e-05	1.655692e-01	1.279917e-05	1.267299e-01	9.896326e-06	9.770146e-02
1000	3.266200e-10	3.258319e-06	2.775923e-10	2.764913e-06	1.751631e-10	1.749061e-06
2000	1.794071e-10	1.787854e-06	1.486934e-10	1.478689e-06	1.182063e-10	1.176943e-06
3000	8.397363e-11	8.360031e-07	6.663360e-11	6.627562e-07	9.469445e-11	9.422111e-07
4000	3.802493e-11	3.783603e-07	2.495202e-11	2.483596e-07	7.013345e-11	6.976806e-07
5000	2.117348e-11	2.108264e-07	1.024353e-11	1.021418e-07	4.723562e-11	4.698080e-07
6000	1.487568e-11	1.482745e-07	6.223183e-12	6.215403e-08	2.885698e-11	2.869561e-07
7000	1.156446e-11	1.153114e-07	5.011126e-12	5.008564e-08	1.650122e-11	1.641030e-07
8000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00

TABLE II: Training and Validation Loss for different values of n across epochs

Similarly, for $n = 1$, the training loss begins at 1.279917×10^{-5} and the validation loss at 1.267299×10^{-1}, also reducing to zero by epoch 8000. The $n = 2$, where both training and validation losses decrease from their initial values to zero across the epochs.

This consistent decrease in both training and validation loss values for all quantum states demonstrates that the PINN model is effectively learning and minimizing errors throughout the training process. It also highlights the model's robustness and reliability across different energy levels, as it performs well in both training and validation phases, eventually achieving perfect training and validation results by the end of the training period.

The table III presents the predicted values of the wave function ψ at various positions x for different quantum numbers n. The positions x range from $-a$ to a, with corresponding ψ values given for quantum numbers $n = 0$, $n = 1$, $n = 2$. For each position x, the table provides the predicted ψ values for these four quantum states. For instance, at $x = -5$, the predicted ψ values are 0.000001 for $n = 0$, 0.000004 for $n = 1$, 0.0000 for $n = 2$. Similarly, at $x = 5$, the ψ values are -0.000012 for $n = 0$, -0.000024 for $n = 1$, 0.0000 for $n = 2$. This table illustrates how the wave function ψ changes with position x for different quantum numbers, highlighting the spatial distribution and behavior of ψ across different energy levels.

TABLE III: Predicted n values for x ranging from -5 to $+5$ after final training steps

x	**Predicted Psi**		
	n=0	***n=1***	***n=2***
-5	0.000001	0.000004	0.0000
-3.8889	0.000005	0.000001	0.0007
-2.7778	0.000048	-0.000057	0.0010
-1.6667	0.000414	-0.003737	0.0038
-0.5556	0.015988	-0.003849	-0.0014
0.5556	0.016005	0.004185	-0.0013
1.6667	0.004657	0.003788	0.0035
2.7778	0.000387	0.000376	0.0010
3.8889	0.000009	-0.000042	0.0008
5	-0.000012	-0.000024	0.0000

V. Discussion

In this work, we demonstrated the use of Physics-Informed Neural Networks (PINNs) for solving the time-independent Schrödinger equation, in particular the quantum harmonic oscillator. In our method, we designed a suitable loss function to improve the accuracy of wavefunction predictions, therefore directly integrating physical rules into the neural network training process.

The results confirm the capability of PINNs to solve the Schrödinger equation for different n levels and accurately predict the quantum wavefunctions portraying the quantum phenomena. By enforcing the physical laws of quantum mechanics into their learning mechanism, PINNs intrinsically produce solutions which obey the governing physical equations, all without the need for numerical methods.

Table II also visualizes low training and validation loss ranks over various quantum states, confirming the high precision of the PINNs solution of wavefunctions. The wavefunctions solved by the PINNs match the theoretical solutions, thus indicating the feasibility of our method in practice.

PINN not only have the advantage of being generally less computationally expensive than classical numerical techniques, which can be particularly limited for complex systems, but relies on state of the art techniques of deep neural networks. These capabilities can be an asset in, for example, large-scale quantum systems or higher-dimensional problems.

In the sense of flexibility, PINN can take any boundary conditions and arbitrary potential functions. This have made it applicable to various quantum mechanical problems beyond the harmonic oscillator such as systems with more complicated interactions and potentials.

VI. Conclusion

This work underscores the promise of Physics-Informed Neural Networks (PINNs) as a solver to time-independent Schrödinger equation. By incorporating physical laws in the neural network training, PINNs deliver precise answers to quantum mechanical challenges and present high-resolution wavefunctions of various energy levels. These indicate that, compared to traditional numerical tools, PINNs exhibit a promising alternative in terms of accuracy, ease of implementation, and flexibility. This work illustrates the enormous potential of physics informed neural networks to advance our knowledge of quantum mechanics and to reformulate the way we solve fundamental equations in physics.

References

[1] Briggs JS, Boonchui S, Khemmani S. The derivation of time-dependent Schrödinger equations. Journal of Physics A: Mathematical and Theoretical. 2007 Jan 23;40(6):1289.

[2] Band, Yehuda B., and Yshai Avishai. Quantum mechanics with applications to nanotechnology and information science. Academic Press, 2013.

[3] Dong, Jianping, and Mingyu Xu. ”Space–time fractional Schrödinger equation with time-independent potentials.” Journal of Mathematical Analysis and Applications 344.2 (2008): 1005-1017.

[4] Faddeev, Lûdvig Dmitrievič, and Stanislav Petrovich Merkuriev. Quantum scattering theory for several particle systems. Vol. 11. Springer Science & Business Media, 2013.

[5] Iten, Raban, Tony Metger, Henrik Wilming, Lídia Del Rio, and Renato Renner. ”Discovering physical concepts with neural networks.” Physical review letters 124, no. 1 (2020): 010508.

[6] Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. Nature Reviews Physics. 2021 Jun;3(6):422-40.

[7] Mao, Zhiping, Ameya D. Jagtap, and George Em Karniadakis. ”Physics-informed neural networks for high-speed flows.” Computer Methods in Applied Mechanics and Engineering 360 (2020): 112789.

[8] Raissi, Maziar, 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 378 (2019): 686-707.

[9] Myrvold, Wayne C. ”What is a Wavefunction?.” Synthese 192.10 (2015): 3247-3274.

[10] Scrinzi, Armin. ”Time-Dependent Schrödinger Equation.” Attosecond and XUV Physics: Ultrafast Dynamics and Spectroscopy (2014): 257-292.

[11] Jiang, Xiaotian, Danshi Wang, Qirui Fan, Min Zhang, Chao Lu, and Alan Pak Tao Lau. ”Physics-Informed Neural Network for Nonlinear Dynamics in Fiber Optics.” Laser & Photonics Reviews 16, no. 9 (2022): 2100483.

[12] Zhang, Chaojun, and Yuexing Bai. ”A Novel Method for Solving Nonlinear Schrödinger Equation with a Potential by Deep Learning.” Journal of Applied Mathematics and Physics 10.10 (2022): 3175-3190.

[13] Shah, Karan, Patrick Stiller, Nico Hoffmann, and Attila Cangi. ”Physics-Informed Neural Networks as Solvers for the Time-Dependent Schrödinger Equation.” arXiv preprint arXiv:2210.12522 (2022).

[14] Harcombe, Liam, and Quanling Deng. ”Physics-informed neural networks for discovering localised eigenstates in disordered media.” Journal of Computational Science 73 (2023): 102136.

[15] Schrödinger, Erwin. ”An undulatory theory of the mechanics of atoms and molecules.” Physical review 28.6 (1926): 1049.

[16] Stenholm, Stig. ”Quantum theory of electromagnetic fields interacting with atoms and molecules.” Physics Reports 6.1 (1973): 1-121.

[17] Busch, Paul, Teiko Heinonen, and Pekka Lahti. ”Heisenberg's uncertainty principle.” Physics reports 452.6 (2007): 155-176.

[18] Dolean, Victorita, Alexander Heinlein, Siddhartha Mishra, and Ben Moseley. ”Multilevel domain decomposition-based architectures for physics-informed neural networks.” Computer Methods in Applied Mechanics and Engineering 429 (2024): 117116.

[19] Cuomo, Salvatore, et al. ”Scientific machine learning through physics–informed neural networks: Where we are and what's next.” Journal of Scientific Computing 92.3 (2022): 88.

[20] Moseley, Ben, Andrew Markham, and Tarje Nissen-Meyer. ”Solving the wave equation with physics-informed deep learning.” arXiv preprint arXiv:2006.11894 (2020).

[21] Lagaris, Isaac E., Aristidis Likas, and Dimitrios I. Fotiadis. ”Artificial neural networks for solving ordinary and partial differential equations.” IEEE transactions on neural networks 9.5 (1998): 987-1000.

[22] Kingma, Diederik P., and Jimmy Ba. ”Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980 (2014).

[23] Liu, Dong C., and Jorge Nocedal. ”On the limited memory BFGS method for large scale optimization.” Mathematical programming 45.1 (1989): 503-528.

[24] Kelley, Henry J. ”Gradient theory of optimal flight paths.” Ars Journal 30.10 (1960): 947-954.

[25] Abadi, Martín, et al. ”Tensorflow: Large-scale machine learning on heterogeneous distributed systems.” arXiv preprint arXiv:1603.04467 (2016).

[26] Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen et al. ”Pytorch: An imperative style, high-performance deep learning library.” Advances in neural information processing systems 32 (2019).

[27] Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, Necula G. JAX: composable transformations of Python+ NumPy programs, v0. 3.13.

[28] Moseley, B. (2022). Physics-informed machine learning: from concepts to real-world applications [PhD thesis]. University of Oxford.

[29] Géron, Aurélien. 2017. ”Training Deep Neural Networks.” In *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media. https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html.

[30] Phillips, Dusty. Python 3 Object-Oriented Programming.: Build robust and maintainable software with object-oriented design patterns in Python 3.8. Packt Publishing Ltd, 2018.

[31] Imambi, Sagar, Kolla Bhanu Prakash, and G. R. Kanagachidambaresan. ”PyTorch.” Programming with TensorFlow: Solution for Edge Computing Applications (2021): 87-104.