Journal of Machine Learning Research 26 (2025) 1-39 Submitted 8/24; Revised 9/25; Published 11/25

Physics Informed Kolmogorov-Arnold Neural Networks for Dynamical Analysis via Efficient-KAN and WAV-KAN

Subhajit Patra EEY247540@IITD.AC.IN
Department of Electrical Engineering
Indian Institute of Technology
Delhi, India

Sonali Panda SP.SONALPANDA@GMAIL.COM
Department of Physics
Indian Institute of Technology
Dhanbad, India

Bikram Keshari Parida PARIDA.BIKRAM90.BKP@GMAIL.COM
Artificial Intelligence and Image Processing Lab.
Sun Moon University
Asan-Si, South Korea

Mahima Arya ARYAMAHIMA@GMAIL.COM
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore 641112, India

Kurt Jacobs DR.KURT.JACOBS@GMAIL.COM
United States DEVCOM Army Research Laboratory
Adelphi, Maryland 20783, USA
&
Department of Physics
University of Massachusetts at Boston
Boston, Massachusetts 02125, USA

Denys I. Bondar DBONDAR@TULANE.EDU
Abhijit Sen ASEN1@TULANE.EDU
Department of Physics and Engineering Physics
Tulane University
New Orleans, LA 70118, USA

Editor: Stephan Mandt

Abstract

Physics-informed neural networks have proven to be a powerful tool for solving differential equations, leveraging the principles of physics to inform the learning process. However, traditional deep neural networks often face challenges in achieving high accuracy without incurring significant computational costs. In this work, we implement the Physics-Informed Kolmogorov-Arnold Neural Networks (PIKAN) through efficient-KAN and WAV-KAN, which utilize the Kolmogorov-Arnold representation theorem. PIKAN demonstrates superior performance compared to conventional deep neural networks, achieving the same level of accuracy with fewer layers and reduced computational overhead. We explore both B-spline and wavelet-based implementations of PIKAN and benchmark their performance across various ordinary and partial differential equations using unsupervised (data-free) and supervised (data-driven) techniques. For certain differential equations, the data-free approach suffices to find accurate solutions, while in more complex scenarios, the data-driven method enhances the PIKAN's ability to converge to the correct solution. We validate our results against numerical solutions and achieve 99% accuracy in most scenarios.

Keywords: Differential Equation Solving, Physics informed models, Kolmogorov-Arnold Neural Network (KAN), Efficient-KAN, WAV-KAN

1. Introduction

The advent of deep learning and its use cases in solving complicated tasks related to computer vision, natural language processing, speech, *etc.*, has led to state-of-the-art applications in industries like healthcare, finance, robotics, to name a few. Further, using deep neural networks (DNNs) in solving differential equations through Physics Informed Neural Networks (PINNs) is another breakthrough that offered a new framework for solving partial differential equations (Raissi et al. (2019)). Since then the field of PINN has received a lot of attention (e.g., see review Cuomo et al. (2022) and is extended to solve fractional equations, integral-differential equations, and stochastic partial differential equations (Pang et al. (2019); Yuan et al. (2022); O'Leary et al. (2022)). PINN has been developed to be more robust and accurate (Wang et al. (2023)) because the original form of PINN has drawbacks (Lawal et al. (2022); Krishnapriyan et al. (2021); Ji et al. (2021); Fuks and Tchelepi (2020); Dwivedi et al. (2021)), which are emanate from deep networks.

Recently, a promising alternative to the traditional multilayer perceptron has been proposed: the Kolmogorov-Arnold Neural Network (KAN) (Liu et al. (2024)). While the universal approximation theorem (Cybenko (1989)) is foundational to deep learning, KAN is based on the Kolmogorov-Arnold representation theorem. This theorem establishes that any multivariate continuous function can be decomposed into sums of univariate continuous functions. KANs leverage the power of splines and multi-layer perceptrons, as KAN is essentially a combination of the two. KAN offers a distinct advantage over traditional Multi-layer Perceptrons (MLPs) by incorporating learnable activation functions in addition to learnable weights (Liu et al. (2024)). This characteristic enhances the ability of KANs to approximate solutions to differential equations with greater accuracy and fidelity compared to conventional neural networks (e.g., PINNs). By allowing the activation functions themselves to adapt during training, KANs can better capture the intricate dynamics and behaviors inherent in differential equations, thereby producing solutions that closely align with actual solutions.

A KAN-based physics-informed neural network (PIKAN) has been shown to solve the 2D Poisson equation (Liu et al. (2024)) for a variety of geometries as well as other partial differential equations (Wang et al. (2024)). KAN was also successfully applied to system identification (Koenig et al. (2024)). A thorough comparison of PIKAN based on different variations of KAN have been made in Shukla et al. (2024). However, more recent implementations of KAN, such as WAV-KAN (Bozorgasl and Chen (2024)) or efficient-KAN (Blealtan (2024)), have not been utilized for solving differential equations.

In this article, we present a PIKAN analysis of various differential equations via efficient-KAN and WAV-KAN. We explore the data-free approach – Data-Free PIKANs (DF-PIKANs), which solve equations without relying on external data, making them ideal for problems where data is scarce or unavailable. Also, the data-driven approach is utilized – Data-Driven PIKANs (DD-PIKANs), which uses existing datasets to enhance solution accuracy. The use of data provides guidance during training, helping the model to learn more accurately. We will demonstrate the performance of both approaches, drawing parallels to similar concepts PINNs. Note that the key advantage of using PIKANs is that they require less experimentation with different architectures. This is because PIKANs (via efficient-KAN and WAV-KAN) can effectively capture the solutions of differential equations with simpler, lower-complexity architectures.

The rest is organized as follows: In Section 2, we provide an introduction to KAN, discussing its theoretical foundation and advantages over traditional neural networks. Section 3 focuses on the implementation details of the PIKAN, explaining the construction and training process of PIKANs, including both B-spline and wavelet-based approaches. Sections 4, 5, 6 and 7 present a series of case studies that showcase the effectiveness of PIKANs in achieving high accuracy across different types of differential equations, validated against numerical solutions. Finally, in Section 9, we summarize our findings, highlighting the benefits and potential future directions for PIKANs in the context of differential equations analysis.

2. Introduction to Kolmogorov Arnold Networks (KAN)

Recall that DNNs is based on the Universal Approximation Theorem, which reads as follows: Let σ be any continuous sigmoidal function. Then finite sums of the form

$$g(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \sigma\left(\mathbf{w}_j^T \mathbf{x} + b_j\right), \text{ where } \mathbf{x}, \mathbf{w}_j \in \mathbb{R}^n, \alpha_j, b_j \in \mathbb{R}$$

are dense in the set of continuous functions on $[0,1]^n$ (Cybenko (1989)). Thus, for any continuous function $f(\mathbf{x})$, there is a sum $g(\mathbf{x})$ of the above form such that

$$|g(\mathbf{x}) - f(\mathbf{x})| < \varepsilon.$$

In neural network context, this theorem shows that a sufficiently smooth function $f(\mathbf{x})$ can be approximated arbitrarily closely on a compact set using a two-layer neural network (NN) with appropriate weights and activation functions (Funahashi (1989); Hornik et al. (1989)). A two layer NN is a network that consists of an input layer, one hidden layer, and an output layer. The hidden layer has neurons with activation functions, and the output

layer typically performs a weighted sum of the hidden layer outputs.

KAN is based on the Kolmogorov-Arnold representation theorem stating that any continuous function of multiple variables can be represented as a superposition of continuous functions of a single variable and addition (Givental et al. (2009); Braun and Griebel (2009); Kolmogorov (1957)). Formally, it states that, given a continuous function $f : [0,1]^n \to \mathbb{R}$, there exist univariate continuous functions $\phi_{i,j}$ and ψ_i such that

$$f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{2n+1} \psi_i \left(\sum_{j=1}^{n} \phi_{i,j}(x_j) \right).$$

The above formulae consists of two layers of univariate continuous functions. The inner layer applies univariate functions $\phi_{i,j}(x_j)$ to each input variable, and their outputs are summed. The outer layer then applies another set of univariate functions ψ_i to the sums.

Note that the potential of utilizing KAN for constructing neural networks has been attempted previously (Lin and Unbehauen (1993); Lai and Shen (2021); Sprecher and Draghici (2002); Leni et al. (2013); Montanelli and Yang (2020); Fakhoury et al. (2022)). However, in Liu et al. (2024), the shortcomings of the previous attempts were finally resolved, resulting in the comprehensive construction of neural networks (KANs) utilizing the Kolmogorov-Arnold theorem. In KANs, nodes perform the function of summing incoming signals without introducing any non-linear transformations. Conversely, edges within the network incorporate learnable activation functions, which are weighted combination of basis function and a B-splines. Throughout the training process, KANs optimize these spline activation functions to match the target function. Further, the learning parameters are fundamentally different from those in traditional neural networks, which use weights and biases. In KANs, the primary parameters are the coefficients of the learnable uni-variate activation functions. In KANs, the primary parameters are the coefficients of the learnable uni-variate activation functions.

In this paper, we will use efficient-KAN (for code implementation refer to Blealtan (2024)), a reformulation of originally proposed KAN (Liu et al. (2024)) which significantly reduces the memory cost and make computation faster. Further, we also utilize an alternative to KAN (or efficiemt-KAN) called as Wavelet Kolmogorov-Arnold Networks (WAV-KAN) (Bozorgasl and Chen (2024)) which uses wavelets instead of B-splines.

In this paper, we will leverage both the use of efficient-KAN and WAV-KAN in solving various differential equations. Note that we will use the name PIKAN consistently, regardless of the type of KAN being used. However, we will make it clear if we use the efficient-KAN or WAV-KAN for solving the differential equation.

3. Implementation of PIKAN

The implementation of PIKAN (either with efficient KAN or WAV-KAN) is similar to that of PINN except for the fact that instead of using DNNs, one uses KANs. For simplicity, we will define it for an ordinary differential equation and introduce DF-PIKAN. For the differential equation subject to the initial condition as follows:

$$\frac{dy(x)}{dx} - f(y(x), x) = 0, \qquad y(a) = y_a \tag{1}$$

where $x \in [0, 1]$, the aim is to train a neural network which finds a function $\hat{y}_\theta(x)$ such that satisfies the equation in an approximate manner, i.e.,

$$\frac{d\hat{y}_\theta(x)}{dx} - f(\hat{y}_\theta(x), x) \approx 0, \qquad \hat{y}_\theta(a) = y_a \tag{2}$$

within the specified domain of the original differential equation. Note that θ denotes the tunable parameters (e.g, weights, biases) of neural networks. In KANs, these parameters are the coefficients of the learnable uni-variate activation functions applied to each input. In order to achieve the approximate accurate form $\hat{y}_\theta(x)$, the main trick is to introduce the correct loss function for the neural network. Let us define the residual of the differential equation as

$$\mathcal{R}_\theta(x_{\mathrm{r}}^i) = \frac{d\hat{y}_\theta(x_{\mathrm{r}}^i)}{dx} - f(\hat{y}_\theta(x_{\mathrm{r}}^i), x_{\mathrm{r}}^i) \tag{3}$$

where $\{x_r^i\}_{i=1}^{N_r}$ represents specific points in the domain of the variable x. These points are either vertices of a fixed mesh or points that are randomly sampled during each iteration of a gradient descent algorithm. Now using the residual loss defined above, physics loss is defined as:

$$\mathcal{L}_{\mathrm{r}} = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{R}_\theta(x_{\mathrm{r}}^i) \right|^2. \tag{4}$$

To ensure that the right-hand side (RHS) of Eq 2 is approximately zero, we need to minimize the physics loss. This involves reducing the discrepancy between the derivative of the neural network output $\frac{d\hat{y}_\theta(x)}{dx}$ and the function $f(\hat{y}_\theta(x), x)$ at the mesh points x_r^i. By making the physics loss as small as possible, we ensure that the neural network solution adheres closely to the underlying physical laws represented by the differential equation.

However, we have not yet considered how to incorporate the information about boundary condition/initial conditions. This can be done as either hard or soft constraints. In the hard constraint style, to incorporate the initial condition $y(a) = y_0$, the output of the neural network is modified such that $\hat{y}_\theta(x) \mapsto y_a + (x - a)\hat{y}_\theta(x)$. In such a case, the physics loss is the only term in the loss function. In the soft constraint style, the initial condition information is included in the loss function. Thus, including boundary condition as a soft constraint, the final loss function for the neural network becomes

$$\begin{aligned} \mathcal{L}_{\text{final}} &= \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{bc}} \\ &= \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{R}_\theta(x_{\mathrm{r}}^i) \right|^2 + \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} \left| \hat{y}_\theta(x_{\mathrm{bc}}^i) - y_a^i \right|^2, \end{aligned} \tag{5}$$

Where N_r is the maximum number of collocation points used to evaluate the residuals and N_{bc} is the maximum number of boundary collocation points. In our example, $N_{bc} = 1$, since we have only one boundary condition.

However, as straightforward as the formulation may appear, it is not without subtleties. First, note that different terms in the loss function can have different magnitudes and, therefore, some terms might dominate the training process, while others are neglected. In order to ensure that all parts of the loss contribute appropriately to the training process, the weights are often multiplied in the loss function to balance different terms. In such a

case, the loss function with multiplicative weight (also including initial condition if any) takes the following form

$$\mathcal{L}_{\text{final}} = \lambda_{\text{r}}\mathcal{L}_{\text{r}} + \lambda_{\text{bc}}\mathcal{L}_{\text{bc}} + \lambda_{\text{ic}}\mathcal{L}_{\text{ic}} \tag{6}$$

The DF-PIKANs utilize the loss function as in Eq (6). Note that

$$\mathcal{L}^{DF}_{\text{final}} = \mathcal{L}_{\text{final}} \tag{7}$$

where the superscript DF in $\mathcal{L}^{DF}_{\text{final}}$ denotes that it pertains to a Data-Free approach.

In this section, we will now delve into DD-PIKANs (similar to DD-PINNs as in Refs. Raissi et al. (2019); Yang et al. (2024)) where the loss function Eq (6) is modified to include observational or simulated data. In other words, DF-PIKANs rely on governing physical equations for learning, while DD-PIKANs use additional data in the loss function to improve accuracy and robustness during training. The loss function is given by

$$\mathcal{L}^{DD}_{\text{final}} = \lambda_{\text{r}}\mathcal{L}_{\text{r}} + \lambda_{\text{bc}}\mathcal{L}_{\text{bc}} + \lambda_{\text{ic}}\mathcal{L}_{\text{ic}} + \lambda_{\text{data}}\mathcal{L}_{\text{data}} \tag{8}$$

where

$$\mathcal{L}_{\text{data}} = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} \left|\hat{u}_\theta\left(x^i_{\text{data}}\right) - u\left(x^i_{\text{data}}\right)\right|^2 \tag{9}$$

which ensures the solution fits the observed collocation data points $\left(x^i_{\text{data}}, u\left(x^i_{\text{data}}\right)\right)$.

For completeness, we write the DD-PIKAN loss function for differential equation with two variables, for example, Burger equation. Burgers' equation in one spatial dimension x and time t is given by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

where $u = u(x,t)$ is the velocity field, and ν is the kinematic viscosity – a given parameter. The initial condition specifying the value of u at the initial time $t = 0$ is given by

$$u(x,0) = u_0(x)$$

and the boundary condition that specifies the behavior of u at the boundaries of the spatial domain x, often $x = 0$ and $x = L$:

$$u(0,t) = u_L(t), \quad u(L,t) = u_R(t)$$

The residual term, physics loss, initial condition loss, boundary loss and the total loss are the following

$$\begin{aligned}
R_\theta(x,t) &= \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2},\\
\mathcal{L}_{\mathrm{r}} &= \frac{1}{N_r}\sum_{i=1}^{N_r} |\mathcal{R}_\theta(x_{\mathrm{r}}^i, t_{\mathrm{r}}^i)|^2,\\
\mathcal{L}_{\mathrm{ic}} &= \frac{1}{N_{ic}}\sum_{j=1}^{N_{ic}} \Big(\hat{u}(x_{\mathrm{ic}}^j, 0) - u_0(x_{\mathrm{ic}}^j)\Big)^2,\\
S_1 &= \frac{1}{N_{bc}}\sum_{k=1}^{N_{bc}} \Big(\hat{u}(0, t_{\mathrm{bc}}^k) - u_L(t_{\mathrm{bc}}^k)\Big)^2,\\
S_2 &= \frac{1}{N_{bc}}\sum_{k=1}^{N_{bc}} \Big(\hat{u}(L, t_{\mathrm{bc}}^k) - u_R(t_{\mathrm{bc}}^k)\Big)^2,\\
\mathcal{L}_{bc} &= S_1 + S_2,\\
\mathcal{L}_{\mathrm{final}}^{DF} &= \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{ic}} + \mathcal{L}_{\mathrm{bc}}.
\end{aligned} \tag{10}$$

We note that it is well-known that the data-free approach of PINNs presents some limitations in finding a correct solution to many differential equations. For example, for one-dimensional convection problems, PINNs achieve a good solution only for small values of convection coefficients (Krishnapriyan et al. (2021); Yang et al. (2024); Huang and Agarwal (2023); Rohrhofer et al. (2023)). In scenarios like this, adopting a data-driven approach yield a significant improvement. By incorporating a data loss term, the training process is guided more effectively toward achieving correct solutions . We will demonstrate that the same applies to the Burger equation below.

4. Linear ordinary differential equation

Consider the following initial value problem:

$$\frac{dy}{dx} = 3x^2, \qquad y(0) = 1. \tag{11}$$

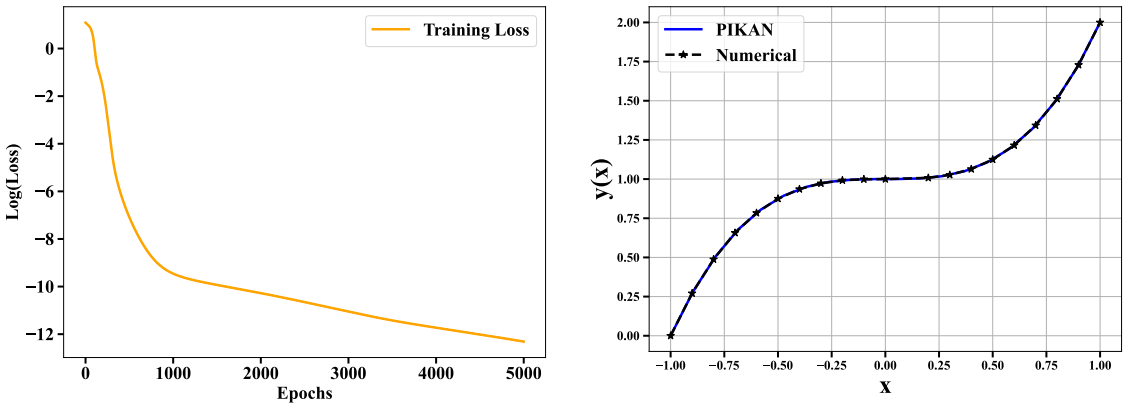
Let us define the residual loss function as

$$\mathcal{R}_\theta(x) = \frac{d\hat{y}_\theta(x)}{dx} - 3x^2,$$

where $\hat{y}_\theta(x)$ is the PIKAN solution which we intend to learn. From Eq (6) and Eq (7), the final loss function reads as

$$\mathcal{L}_{\mathrm{final}}^{DF} = \frac{1}{N_r}\sum_{i=1}^{N_r} \big|\mathcal{R}_\theta(x_{\mathrm{r}}^i)\big|^2 + |\hat{y}_\theta(0) - 1|^2 \tag{12}$$

where x_{r}^i are the collocation points with the weights $\lambda_{\mathrm{r}} = \lambda_{\mathrm{ic}} = 1$ and $\lambda_{\mathrm{bc}} = 0$.



(a) (b)

Figure 1: (a) Illustrates the relationship between loss (Eq. (12)) and epoch. (b) A comparison between the PIKAN predicted solution and numerically exact solution of Eq. (11).

Architecture	Spline-order	Grid size	Basis Activation	Loss
$[1, 5, 4, 3, 1]$	3	5	sin	10^{-6}

Table 1: The architecture details for Sec. 4.

We utilized the efficient-KAN method to solve the differential equation, employing collocation points $N_x = 100$ equally distributed on the interval $-1 \leq x \leq 1$. The model was trained for 5000 epochs with the learning rate $\eta = 0.001$. The base activation function is sin. At the end of the 5000 epochs (see Fig. 1a), the recorded mean squared error (MSE) loss was on the order of 10^{-6}. Further details are listed in Table 1. The comparison of the PIKAN Predicted solution and the exact Numerical result of Eq (11) are given in Fig. 1b.

5. System of Ordinary Differential Equations

In this section, we employed PIKAN to solve coupled linear and non-linear differential equations that have been previously solved using PINN (Uddin et al. (2023)) and by Galerkin Method with Cubic B-Splines (Kasi Viswanadham (2019)).

We begin by considering the simple case

$$\begin{aligned} \frac{dv(x)}{dx} &= u, \\ \frac{du(x)}{dx} &= -v, \end{aligned} \qquad \begin{cases} u(0) = 1, \\ v(0) = 0. \end{cases} \tag{13}$$

For which the residuals are defined as

$$\mathcal{R}^u_\theta(t) = \frac{d\hat{u}(x)}{dx} + \hat{v}(x), \quad \mathcal{R}^v_\theta(t) = \frac{d\hat{v}(x)}{dx} - \hat{u}(x).$$

Note that $\hat{u}(x)$ and $\hat{v}(x)$ are the PIKAN solution to be learned. Note that henceforth, we drop the parameter θ in the notation of the PIKAN solution. Hereinafter, the weights $\lambda_{\mathrm{r}}, \lambda_{\mathrm{ic}}$ and λ_{bc} in the final loss function (8) are all set to unity. As a result, the final loss becomes

$$\mathcal{L}_{\mathrm{final}}^{DF} = \frac{1}{N_r}\sum_{i=1}^{N_r}\left(\left|\mathcal{R}_\theta^u(x_{\mathrm{r}}^i)\right|^2 + \left|\mathcal{R}_\theta^v(x_{\mathrm{r}}^i)\right|^2\right) + (\hat{u}(0)-1)^2 + (\hat{v}(0)-0)^2. \tag{14}$$

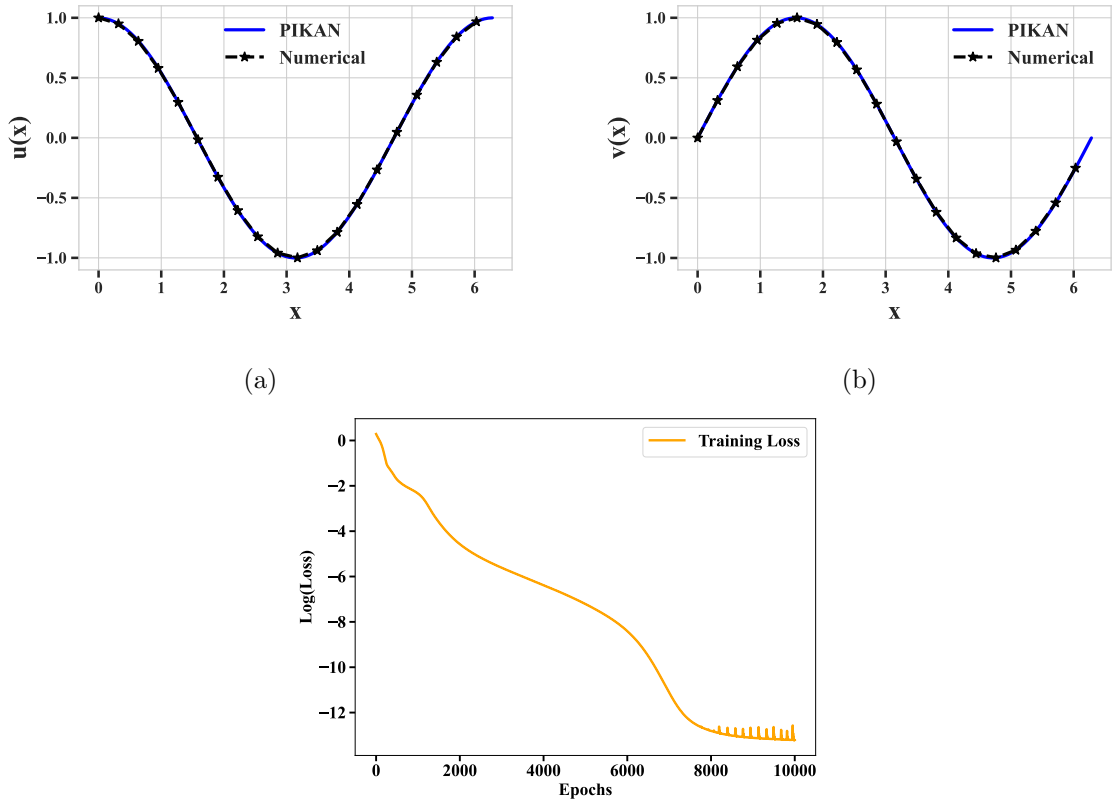


Figure 2: (a) and (b) Illustrates the comparison between the PIKAN-predicted solution and the numerically exact solution of Eq. (13). (c) Illustrates the evolution of the loss, defined in Eq. (14), with training epochs.

In coupled differential equation we have seen that the convergence of the efficient-KAN is not satisfactory and to resolve this, we have used WAV-KAN employing collocation points $N_x = 100$ equally distributed on the interval $0 \leq x \leq 2\pi$. The model was trained for 10000 epochs and Adam optimizer with a learning-rate $\eta = 0.001$ is used.The recorded mean squared error (MSE) loss was of the order of 10^{-6}. Further details can be obtained from Table 2. The comparison of the PIKAN predicted solution and the numerically calculated solution of Eq. (13) is shown in Fig 2a & 2b.

Architecture	Wavelet type	Loss
$[1, 3, 2]$	sin	10^{-6}

Table 2: Architecture details of the model used in section 5.

.

5.1 System of Linear Ordinary Differential Equations

Consider the linear coupled system with boundary conditions (Uddin et al. (2023)):

$$\begin{aligned} u''(x) + xu(x) + 2v'(x) &= u_1, \\ u(x) + v''(x) + 2v(x) &= u_2, \\ \begin{cases} u(0) & = 0 \\ v'(1) + v(1) & = \cos(1) + \sin(1) \\ u(1) & = 2 \\ v'(0) & = 1. \end{cases} \end{aligned} \tag{15}$$

Here, $0 \leq x \leq 1$ and the non-homogeneous terms are $u_1(x) = x^3 + x^2 + 2 + 2\cos(x)$ and $u_2(x) = x^2 + x + \sin(x)$, respectively.

The residuals read

$$\begin{aligned} \mathcal{R}^1_\theta(x) &= \hat{u}''(x) + x\hat{u}(x) + 2\hat{v}'(x) - u_1(x), \\ \mathcal{R}^2_\theta(x) &= \hat{u}(x) + \hat{v}''(x) + 2\hat{v}(x) - u_2(x). \end{aligned}$$

As a result, we have the physics loss, the boundary loss and the final loss as

$$\begin{aligned} \mathcal{L}_{\mathrm{r}} &= \frac{1}{N_r} \sum_{i=1}^{N_r} \left(\left| \mathcal{R}^1_\theta(x^i_{\mathrm{r}}) \right|^2 + \left| \mathcal{R}^2_\theta(x^i_{\mathrm{r}}) \right|^2 \right), \\ \mathcal{L}_{\mathrm{bc}} &= \left(\hat{u}(0) - 0\right)^2 + \left(\hat{v}'(1) + \hat{v}(1) - (\cos(1) + \sin(1))\right)^2 \\ &\quad + \left(\hat{u}(1) - 2\right)^2 + \left(\hat{v}'(0) - 1\right)^2, \\ \mathcal{L}^{DF}_{\mathrm{final}} &= \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{bc}}. \end{aligned} \tag{16}$$

We have utilized the efficient-KAN method to solve the coupled-linear differential equation (15), employing collocation points $N_x = 100$ equally distributed on the interval $0 \leq x \leq 1$ (Uddin et al. (2023)). Note that we have observed that there is not much difference between the performance of WAV-KAN and efficient-KAN for this problem, but the convergence of WAV-KAN is faster than efficient-KAN. The model was trained for 10000 epochs and AdamW optimizer is used with a learning rate $\eta = 0.001$. At the end of the 10000 epochs, the recorded loss was of the order of 10^{-5}. Further details about the model used are listed in Table 3. The comparison of the PIKAN predicted solution and the numerically exact solution of $u(x)$ and $v(x)$ is shown in Fig 3a & 3b.

5.2 System of Nonlinear Ordinary Differential Equations

Consider the problem (Uddin et al. (2023)):

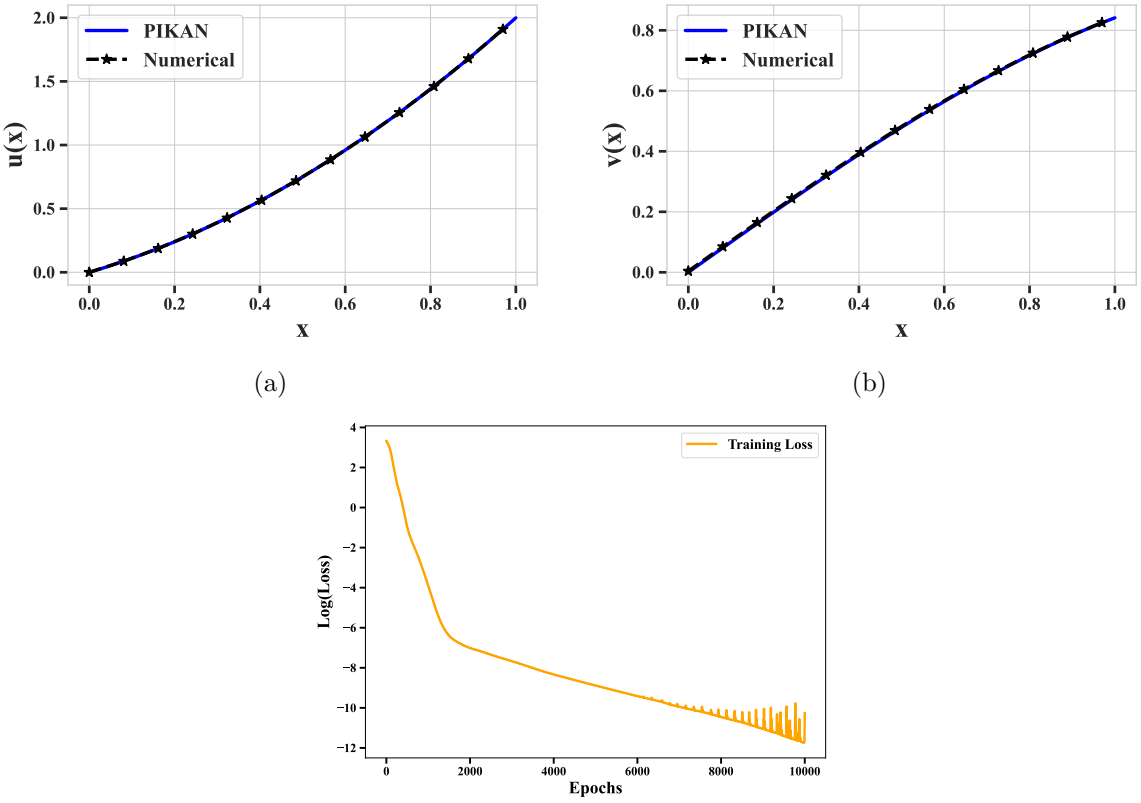


Figure 3: (a) and (b) Shows the Comparison between the PIKAN-predicted solution and the numerically exact solution of Eq. (15). (c) shows the plot of loss vs Iterations, defined in Eq. (16), with training epochs.

Architecture	Spline-order	Grid size	Basis Activation	Loss
$[1, 2, 3, 2]$	3	5	sin	10^{-5}

Table 3: Architecture details of the model used in section 5.1.

$$\begin{aligned} u''(x) + xu(x) + 2xv(x) + xu^2(x) &= u_3(x), \\ x^2u(x) + v'(x) + v(x) + \sin(x)v^2(x) &= u_4(x), \\ \begin{cases} u(0) = 0, \quad u(1) = 0, \\ v(0) = 0, \quad v(1) = 0. \end{cases} \end{aligned} \tag{17}$$

where,

$$\begin{aligned} u_3(x) &= 2x\sin(\pi x) + x^5 - 2x^4 + x^2 - 2, \\ u_4(x) &= x^3(1-x) + \sin(\pi x)\left(1 + \sin(x)\sin(\pi x)\right) \\ &\quad + \pi\cos(\pi x). \end{aligned}$$

Let us define the residual $\mathcal{R}^1_\theta(x)$ and $\mathcal{R}^2_\theta(x)$ using PIKAN to-be-learned solutions $\hat{u}(x)$ and $\hat{v}(x)$ as

$$\begin{aligned} \mathcal{R}^1_\theta(x) &= \hat{u}''(x) + x\hat{u}(x) + 2x\hat{v}'(x) - u_3(x), \\ \mathcal{R}^2_\theta(x) &= \hat{u}(x) + \hat{v}''(x) + 2\hat{v}(x) - u_4(x). \end{aligned}$$

As a result, the physics loss, boundary loss and final loss becomes

$$\mathcal{L}_{\mathrm{r}} = \frac{1}{N_r}\sum_{i=1}^{N_r}\left(\left|\mathcal{R}^1_\theta(x^i_{\mathrm{r}})\right|^2 + \left|\mathcal{R}^2_\theta(x^i_{\mathrm{r}})\right|^2\right), \tag{18}$$

$$\mathcal{L}_{bc} = \left(\hat{u}(0) - 0\right)^2 + \left(\hat{u}(1) - 0\right)^2 + \left(\hat{v}(0) - 0\right)^2 + \left(\hat{v}(1) - 0\right)^2, \tag{19}$$

$$\mathcal{L}^{DF}_{\mathrm{final}} = \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{bc}}. \tag{20}$$

We have utilized the WAV-KAN method to solve the coupled-nonlinear differential equation (17), employing collocation points $N_x = 100$ equally distributed on the interval $0 \le x \le 1$ (Uddin et al. (2023)). The interesting point is that there is not much difference between the performance of WAV-KAN and efficient-KAN for this particular problem but the convergence rate of WAV-KAN was faster than efficient-KAN. The model was trained for 10000 epochs and AdamW optimizer is used with a variable learning rate starting from $\eta = 0.01$, and at the end of every 1000 epochs the learning rate is updated following the given rule, $\eta_{new} = \frac{\eta_{old}}{5}$. At the end of the 10000 epochs, the recorded loss was of the order of 10^{-5}. Further details about the model used is listed in Table 4. The comparison of PIKAN predicted solution and the numerically computed solution is shown in Fig 4a, 4b.

Architecture	Wavelet type	Loss
$[1, 7, 2]$	sin	10^{-5}

Table 4: Architecture details of the model used in section 5.2.

5.3 The Lorenz Equations

The Lorenz equation is a system of ordinary differential equations that was originally developed to model convection currents in the atmosphere (Lorenz (1963)). It is an iconic example in the chaos theory and has significantly influenced our understanding of dynamic systems. We focus on exploring the non-chaotic solutions of the Lorenz system. Consider the following set of equations (Robinson et al. (2022)),

$$\begin{aligned} x'(t) &= \sigma(y(t) - x(t)), \\ y'(t) &= x(\rho - z(t)) - y(t), \\ z'(t) &= x(t)y(t) - \beta z(t). \end{aligned} \qquad \begin{cases} x(0) = 1, \\ y(0) = 1, \\ z(0) = 1, \end{cases} \tag{21}$$

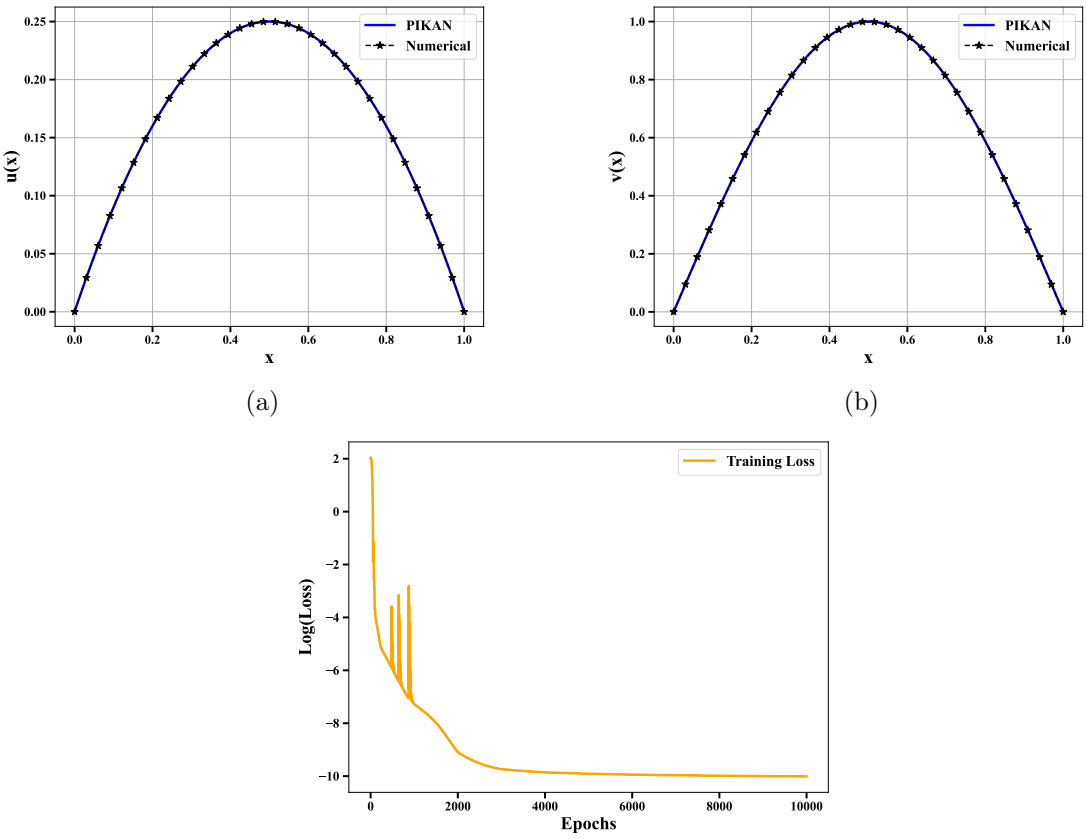


Figure 4: (a) and (b) illustrate the comparison between the PIKAN-predicted solution and the numerically exact solution of Eq. (17). (c) illustrates the evolution of the loss, defined in Eq. (20), with training epochs.

Let us define the following residuals using to-be-learned PIKAN solutions $\hat{x}(t)$, $\hat{y}(t)$ and $\hat{z}(t)$ as,

$$\begin{aligned}\mathcal{R}^1_\theta(t) &= \hat{x}'(t) - \sigma(\hat{y}(t) - \hat{x}(t)) \\ \mathcal{R}^2_\theta(t) &= \hat{y}'(t) - (\hat{x}(t)(\rho - \hat{z}(t)) - \hat{y}(t)) \\ \mathcal{R}^3_\theta(t) &= \hat{z}'(t) - (\hat{x}(t)\hat{y}(t) - \beta\hat{z}(t))\end{aligned} \tag{22}$$

and the physics loss, initial loss and final loss function takes the form:

$$\begin{aligned}\mathcal{L}_{\mathrm{r}} &= \frac{1}{N_r}\sum_{i=1}^{N_r}\left(\left|\mathcal{R}^1_\theta(t^i_{\mathrm{r}})\right|^2 + \left|\mathcal{R}^2_\theta(t^i_{\mathrm{r}})\right|^2 + \left|\mathcal{R}^3_\theta(t^i_{\mathrm{r}})\right|^2\right), \\ \mathcal{L}_{ic} &= (\hat{x}(0) - 1)^2 + (\hat{y}(0) - 1)^2 + (\hat{z}(0) - 1)^2, \\ \mathcal{L}^{DF}_{\mathrm{final}} &= \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{ic}}.\end{aligned} \tag{23}$$

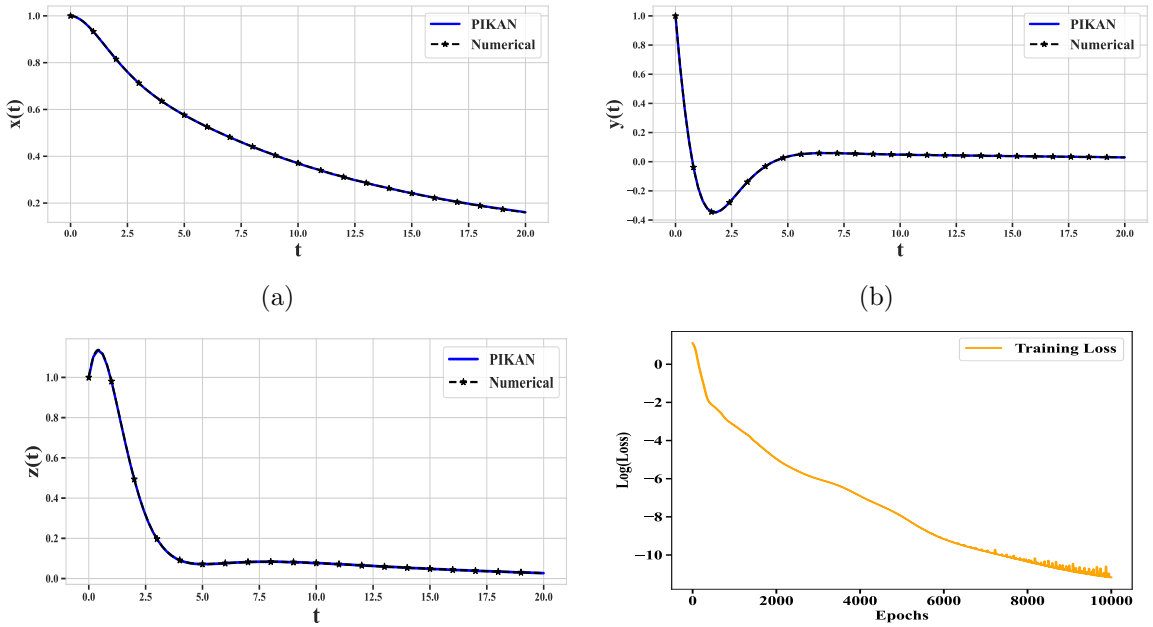


Figure 5: (a)–(c) Comparison between the PIKAN-predicted solutions and the numerically exact solution of Eq. (21). (d) Evolution of the loss, defined in Eq. (23), as a function of training epochs.

Architecture	Wavelet type	Loss
$[1, 12, 8, 3]$	sin	10^{-5}

Table 5: Architecture details of the model used in section 5.3.

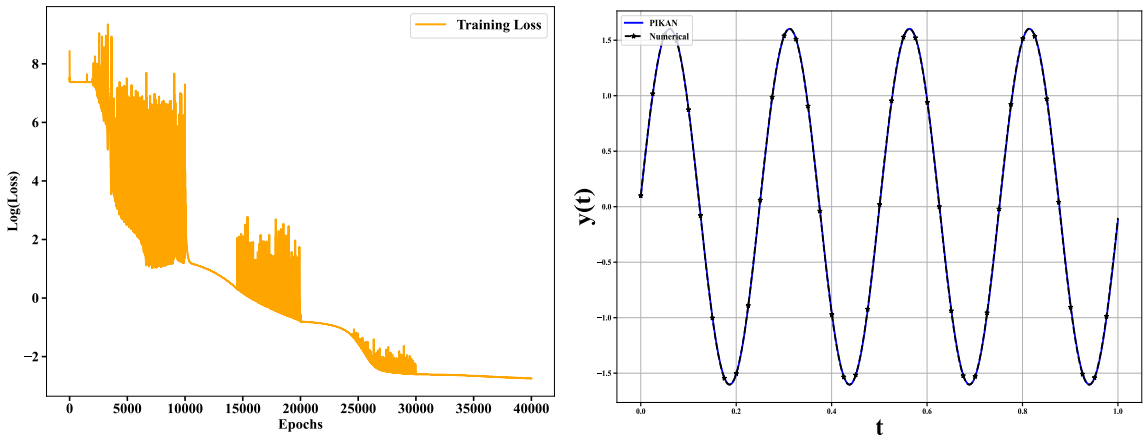
To solve this differential equation we have used both WAV-KAN and efficient-KAN, however the performance of WAV-KAN was better than the efficient-KAN. We have taken collocation points $N_r = 100$ equally distributed over the interval $0 \leq t \leq 20$. The model was trained for 10000 epochs and to minimize the loss function we have used used the AdamW optimizer with a learning rate $\eta = 0.001$. At the end of learning [Fig. 5d], the recorded mean squared loss (MSE) was on the order of 10^{-5}. Further details of the model can be obtained from the Table 5. The comparison between the PIKAN predicted solution and the numerical solutions are shown in Fig. 5a, 5b, 5c.

6. Oscillatory dynamics

6.1 Simple Harmonic Oscillator

Consider the problem

$$y''(t) + \omega_0^2 y(t) = 0, \quad \begin{cases} y(0) = 0.1, \\ y'(0) = 40, \end{cases} \tag{24}$$



(a) (b)

Figure 6: (a) Illustrates the relationship between loss (Eq. (25)) and epoch.(b) Comparison between the PIKAN predicted solution and numerically exact solution for Eq. (24).

Architecture	Wavelet type	Loss
$[1, 8, 6, 8, 1]$	sin	0.06

Table 6: Architecture details of the model used for the section 6.1.

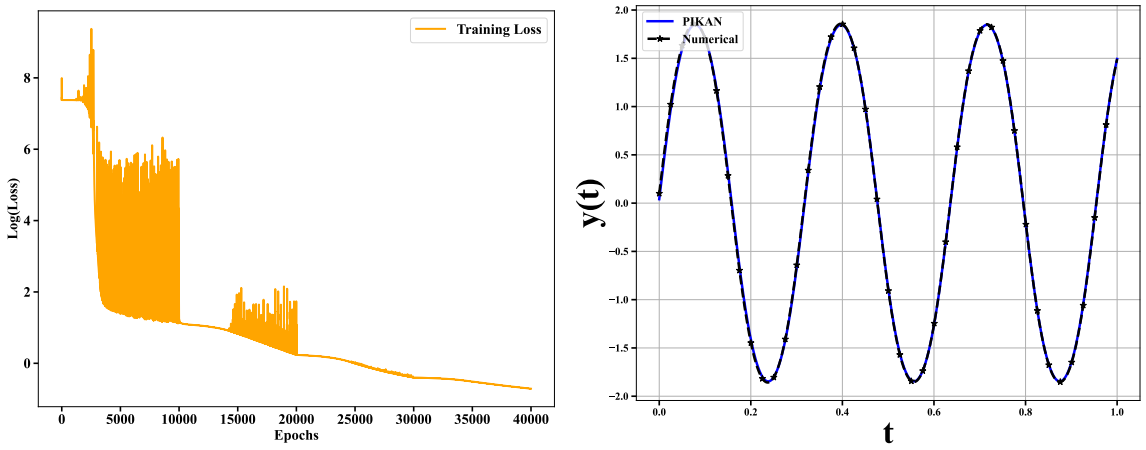
where $\omega_0 = 25$ is the normalized angular frequency , where $0 \leq t \leq 1$. Let us define the following residual

$$\mathcal{R}_\theta(t) = \hat{y}''(t) + \omega_0^2 \hat{y}(t)$$

Consequently, the physics-loss term, the initial condition loss term and the final loss terms are

$$\begin{aligned} \mathcal{L}_{\mathrm{r}} &= \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{R}_\theta(t_{\mathrm{r}}^i) \right|^2, \\ \mathcal{L}_{\mathrm{ic}} &= [(\hat{y}(0) - 0.1))^2 + \left(\hat{y}'(0) - 40\right)^2], \\ \mathcal{L}_{\mathrm{final}}^{DF} &= \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{ic}}. \end{aligned} \tag{25}$$

We have utilized WAV-KAN to solve the Eq (24) employing the collocation point $N_t = 100$ equally distributed over the interval $0 \leq t \leq 1$. We have trained the neural network for 40000 epochs and we have used Adam optimizer with a variable learning rate η starting from 0.01 and decreasing η by a factor of 10 after every 10^4 epochs. At the end of 40000 epochs, the recorded mean squared error (MSE) loss was on the order of 10^{-1}. Further details of the model can be obtained from Table 6. The comparative analysis of the PIKAN predicted solution and the numerically exact solution are given in Fig. 6b.



(a) (b)

Figure 7: (a) Illustrates the relationship between loss(Eq. (27)) and epoch.(b) Comparison between the PIKAN predicted solution and numerically exact solution of Eq. (26).

6.2 Non-linear Pendulum

The non-linear pendulum Eq. (26) with a sine function, provides an exact depiction of the pendulum's dynamics for all oscillation amplitudes;

$$y''(t) + \omega_0^2 \sin(y(t)) = 0, \quad \begin{cases} y(0) = 0.1, \\ y'(0) = 40, \end{cases} \tag{26}$$

where $\omega_0 = 25$ is the normalized angular frequency, where $0 \leq t \leq 1$.

Let us define the residual as

$$\mathcal{R}_\theta(t) = \hat{y}''(t) + \omega_0^2 \sin(\hat{y}(t))$$

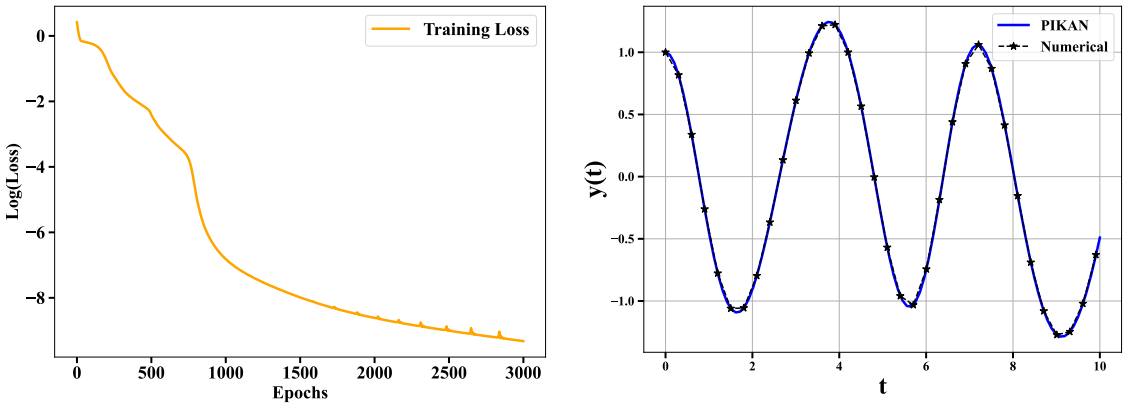
and the physics-loss, the initial condition loss as well as the the final loss are given by

$$\begin{aligned} \mathcal{L}_{\mathrm{r}} &= \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{R}_\theta(t_{\mathrm{r}}^i) \right|^2, \\ \mathcal{L}_{\mathrm{ic}} &= [(\hat{y}(0) - 0.1))^2 + \left(\hat{y}'(0) - 40\right)^2], \\ \mathcal{L}_{\mathrm{final}}^{DF} &= \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{ic}}. \end{aligned} \tag{27}$$

We have utilized WAV-KAN to solve the Eq. (26) employing the collocation point $N_t = 100$ equally distributed over the interval $0 \leq t \leq 1$. We have trained the neural network for 40000 epochs and we have used Adam optimizer with a variable learning rate η starting from 0.001 and decreasing η by 10 after every 10^4 epochs. At the end of 40000 epochs, the recorded mean squared error loss was on the order of 10^{-1}. Further details of the model can be obtained from Table 7. The comparative analysis of the PIKAN predicted solution and the numerically exact solution are given in Fig 7b.

Architecture	Wavelet type	Loss
$[1, 8, 6, 8, 1]$	sin	0.9

Table 7: Architecture details of the model used in section 6.2.



(a) (b)

Figure 8: (a) Illustrates the relationship between loss (Eq. (29)) and epoch,(b) Comparison between the PIKAN predicted solution and numerically exact solution of Eq. (28) for $a = 3, \beta = 1.2$.

6.3 Mathieu Equation

Mathieu's equation stems from his 1868 study on vibrations in an elliptical drum (Mathieu (1868)). Mathieu's equation is a linear second-order ordinary differential equation distinguished from a simple harmonic oscillator by its time-varying (periodic) stiffness coefficient (Kovacic et al. (2018); Ruby (1996)). Consider the initial value problem (Rahman et al. (2024)):

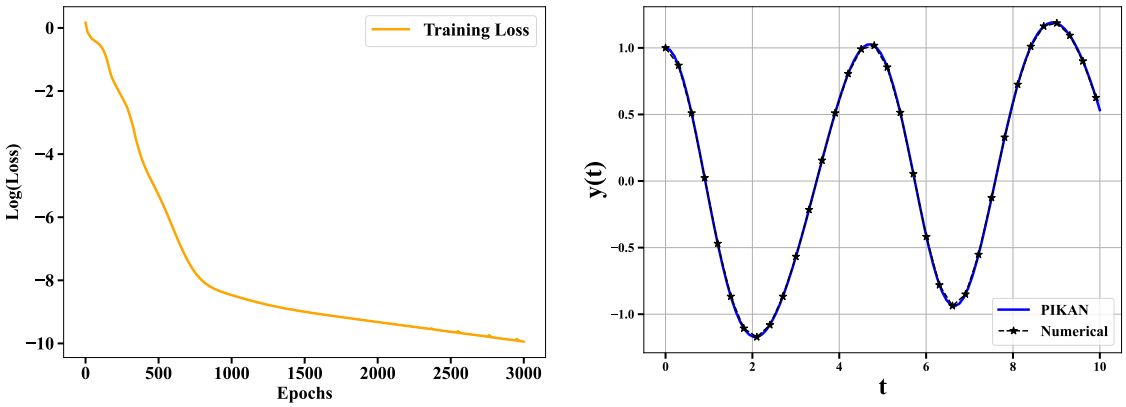
$$y''(t) + (a + \beta \cos t)y(t) = 0, \quad \begin{cases} y(0) = 1, \\ y'(0) = 0. \end{cases} \tag{28}$$

The residual corresponding to this equation is

$$\mathcal{R}_\theta(t) = \hat{y}''(t) + (a + \beta \cos t)\hat{y}(t)$$

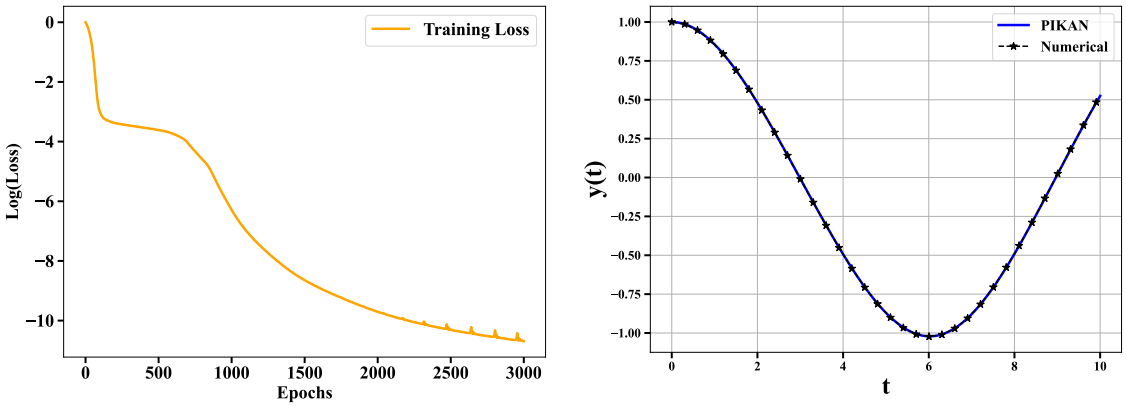
Consequently the physics-loss term, the initial condition loss term, and the total-loss terms are

$$\begin{aligned} \mathcal{L}_{\mathrm{r}} &= \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{R}_\theta(t_{\mathrm{r}}^i) \right|^2, \\ \mathcal{L}_{\mathrm{ic}} &= [(\hat{y}(0) - 1))^2 + \left(\hat{y}'(0)\right)^2], \\ \mathcal{L}_{\mathrm{final}}^{DF} &= \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{ic}}. \end{aligned} \tag{29}$$



(a) (b)

Figure 9: (a) Illustrates the relationship between loss (Eq. (29)) and epoch,(b) Comparison between the PIKAN predicted solution and numerically exact solution of Eq. (28) for $a = 2, \beta = 1$.



(a) (b)

Figure 10: (a) Illustrates the relationship between loss (Eq. (29)) and epoch, (b) Comparison between the PIKAN predicted solution and numerically exact solution of Eq. (28) for $a = 0.25, \beta = 0.05$.

We have utilized WAV-KAN to solve the Mathieu equation (28) employing ($N_r = 100$) collocation points equally spaced on the interval $0 \leq t \leq 10$. The model is trained for 3000 epochs and we have taken Adam optimizer with a learning-rate $\eta = 0.001$. At the end of 3000 epochs we have recorded mean squared error loss on the order of 10^{-6}. The further details of the model is given in Table 8. The comparison of PIKAN predicted solution and exact numerical solution for different values of the parameters α and β are shown in Figs. 8b, 9b and 10b.

Architecture	Wavelet type	Loss
$[1, 12, 8, 1]$	sin	10^{-6}

Table 8: Architecture details of the model used for section 6.3.

Architecture	Wavelet type	Loss
$[1, 12, 8, 1]$	sin	10^{-5}

Table 9: Architecture details of the model used for section 6.4.

6.4 Van der Pol Equation

The Van der Pol equation describes the behavior of a nonlinear oscillator with damping that varies with the amplitude of oscillation. Consider the following initial value problem for Van der Pol equation (Rahman et al. (2024))

$$\begin{aligned} y''(t) + \epsilon \left(c_0 + c_1 \cos(\omega t) + \alpha y^2\right) y'(t) + \omega_n^2 y(t) \\ = f_0 + f_1 \sin(\omega t) \\ \begin{cases} y(0) = 0, \\ y'(0) = 2 \end{cases} \end{aligned} \tag{30}$$

The parameters are

$$\begin{aligned} &c_0 = -1, \quad \epsilon = 0.2, \quad \omega_n = 1, \\ &c_1 = 1, \quad \alpha = 1, \quad \omega = 0.12, \quad f_0 = 0.4 \end{aligned}$$

We will consider two cases depending on the value of the parameter $f_1 = 1$ and $f_1 = 1.7$ respectively.

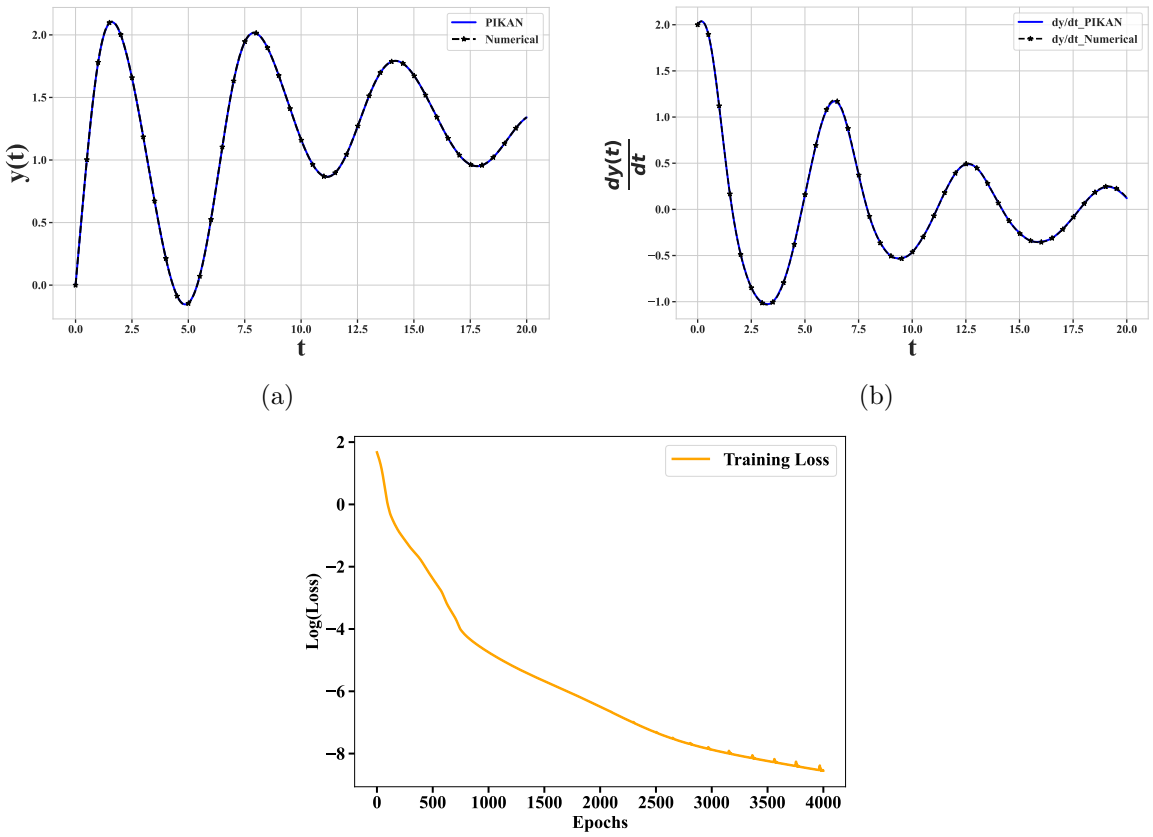
The residual term is given by

$$\begin{aligned} \mathcal{R}_\theta(t) = \big[\hat{y}''(t) + 0.2 \left(-1 + \cos(0.12t) + \hat{y}(t)^2\right) \hat{y}'(t) \\ + \hat{y}(t) - 0.4 - \sin(0.12t)\big]^2, \end{aligned}$$

and the physics loss, the initial condition loss and the total loss takes the form

$$\begin{aligned} \mathcal{L}_{\mathrm{r}} &= \frac{1}{N_r} \sum_{i=1}^{N_r} \left|\mathcal{R}_\theta(t_{\mathrm{r}}^i)\right|^2, \\ \mathcal{L}_{\mathrm{ic}} &= [(\hat{y}(0) - 0)^2 + \left(\hat{y}'(0) - 2\right)^2], \\ \mathcal{L}_{\mathrm{final}}^{DF} &= \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{ic}}. \end{aligned} \tag{31}$$

We have utilized WAV-KAN to solve the Eq (30) employing $N_t = 100$ collocation points equally spaced on the interval $0 \leq t \leq 20$. The model is trained for 4000 epochs and we have used Adam optimizer with a variable learning rate η starting from 0.001 and decreasing η by 10 after every 300 epochs. At the end of learning, we have recorded a mean squared loss on the order of 10^{-5} (see Fig. 11c and Fig. 12c, respectively). Further details are listed in Table 9. The comparison of PIKAN predicted solution and exact numerical solution are given in Figs. 11a,11b and 12a, 12b.



(a) (b)

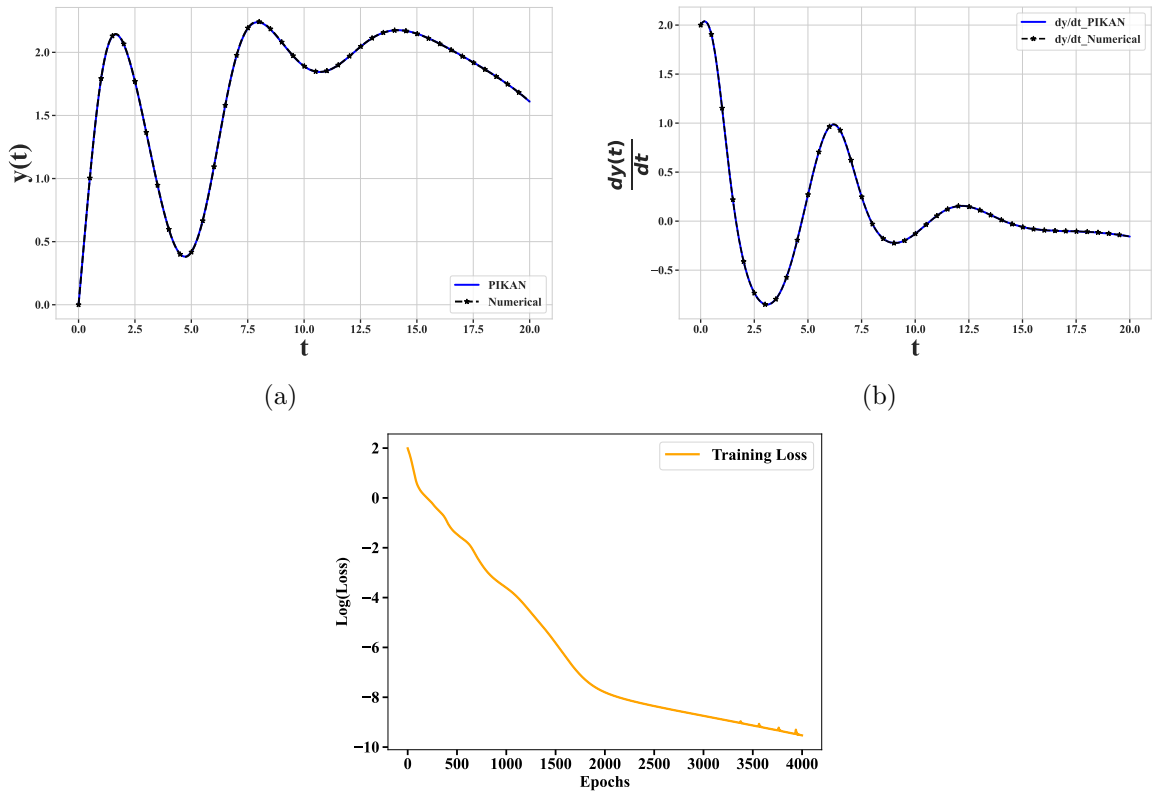
(c)

Figure 11: (a) and (b) Illustrate the comparison between the PIKAN-predicted solution and the numerically exact solution of Eq. (30) for $f_1 = 1$. (c) Illustrates the evolution of the loss, defined in Eq. (31), with training epochs.

7. Non-linear partial differential equations

7.1 Burgers' Equation

The Burger equation, first introduced by Bateman (Bateman (1915)) and later solved by Burger (Burgers (1948)), has a wide range of applications in classical dynamics, nonlinear harmonics, and plasma physics (Bonkile et al. (2018); Bec and Khanin (2007); Vallee and Moreau (2007)). An interesting aspect of this equation is its strong dependence on boundary conditions and the parameter $\nu = 0.1$ [see Eq. (32)]. Depending on the boundary condition, we have divided our investigation of Burgers' equation into two different cases because they require very different approaches: Case 1 can be resolved by minimizing MSE-Loss constructed from training examples, while Case 2 requires more precise training data generated for a specific time scale. This method is known as the data-driven approach (Raissi et al. (2019)).



(a) (b)

(c)

Figure 12: (a) and (b) illustrate the comparison between the PIKAN-predicted solution and the numerically exact solution of Eq. (30) for $f_1 = 1.7$. (c) illustrates the evolution of the loss, defined in Eq. (31), with training epochs.

Consider Burger's equation with the following boundary and initial condition (Uddin et al. (2023))

$$u_t(x,t) + u(x,t)u_x(x,t) = \nu u_{xx}(x,t), \quad \begin{cases} u(x,0) = \sin(\pi x) \\ u(0,t) = 0 \\ u(1,t) = 0, \end{cases} \tag{32}$$

where $x \in [0,1]$ and $t \in [0,1]$.

We can now define the residual term using the PIKAN to-be-learned solution $\hat{u}(x,t)$ as follows

$$\mathcal{R}_\theta(x,t) = \hat{u}_t + \hat{u}\hat{u}_x - \nu \hat{u}_{xx}.$$

The physics loss, initial condition loss, boundary loss, data-driven loss and the total loss are the following

$$\mathcal{L}_{\mathrm{r}} = \frac{1}{N_r}\sum_{i=1}^{N_r}, |\mathcal{R}_\theta(x_{\mathrm{r}}^i, t_{\mathrm{r}}^i)|^2,$$

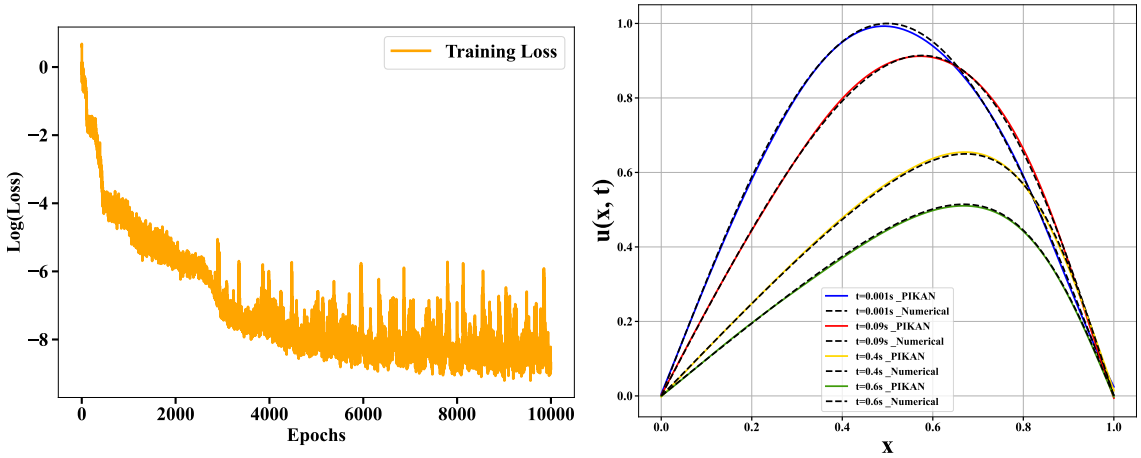
$$\mathcal{L}_{\mathrm{ic}} = \frac{1}{N_{ic}}\sum_{j=1}^{N_{ic}}\left(\hat{u}(x_{\mathrm{ic}}^i, 0) - \sin(\pi x_{\mathrm{ic}}^j)\right)^2, \tag{33}$$

$$\mathcal{L}_{\mathrm{bc}} = \frac{1}{N_{bc}}\sum_{k=1}^{N_{bc}}\left(\hat{u}(0, t_{\mathrm{bc}}^k)\right)^2 + \frac{1}{N_{bc}}\sum_{k=1}^{N_{bc}}\left(\hat{u}(1, t_{\mathrm{bc}}^k)\right)^2,$$

$$\mathcal{L}_{\mathrm{data}} = \frac{1}{N_{data}}\sum_{p=1}^{N_{data}}\left|\hat{u}\left(x_{\mathrm{data}}^p\right) - u\left(x_{\mathrm{data}}^p\right)\right|^2,$$

$$\mathcal{L}_{\mathrm{final}}^{DD} = \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{ic}} + \mathcal{L}_{\mathrm{bc}} + \mathcal{L}_{\mathrm{data}}. \tag{34}$$

where N_{r}, N_{ic}, and N_{bc} are the number of points used to calculate the residual, initial condition loss, and boundary condition loss, respectively.



(a) Loss vs Epoch.

(b) PIKAN solution vs numerical solution.

Figure 13: Results for the 1D Burgers' equation: (a) training loss vs iterations for PIKAN(b) comparison between PIKAN predicted solution vs numerically exact solution.

The solution of Burgers' Equation (32) is done using the data-driven method (8). We have computed the numerical solution first and then we have taken only 10 % of the data from the numerical solution and these data points were chosen randomly. Now we have added an extra loss term (34) that corresponds to the data-driven loss [Eqs. (34) and (8)]. To train the model we have utilized the efficientKAN by employing $N_x = 100$ and $N_t = 100$ collocation points distributed uniformly over the interval $0 \leq x \leq 1$ and $0 \leq t \leq 1$. The model is trained for 10000 epochs and to minimize the loss function we have used the

Architecture	Spline-order	Grid size	Basis Activation	Loss
$[2, 8, 4, 1]$	3	5	sin	10^{-5}

Table 10: The architecture details for Sec. 7.1.

AdamW optimizer with a variable learning rate η starting from 0.005 and decreasing η by a factor of 0.1 after every 10^3 epochs. At the end of training, we have recorded a mean squared error loss on the order of 10^{-5} (see Fig. 13a). Further details of the model is listed in Table 10. The comparison of the predicted solution of PIKAN, PINN vs the numerically exact solution of Eq. (32) computed for different times is shown in Fig. 21, especially in b and c. Figure 14 displays the full solution $u(x, t)$ obtaine via PIKANN and a standard numerical solver.

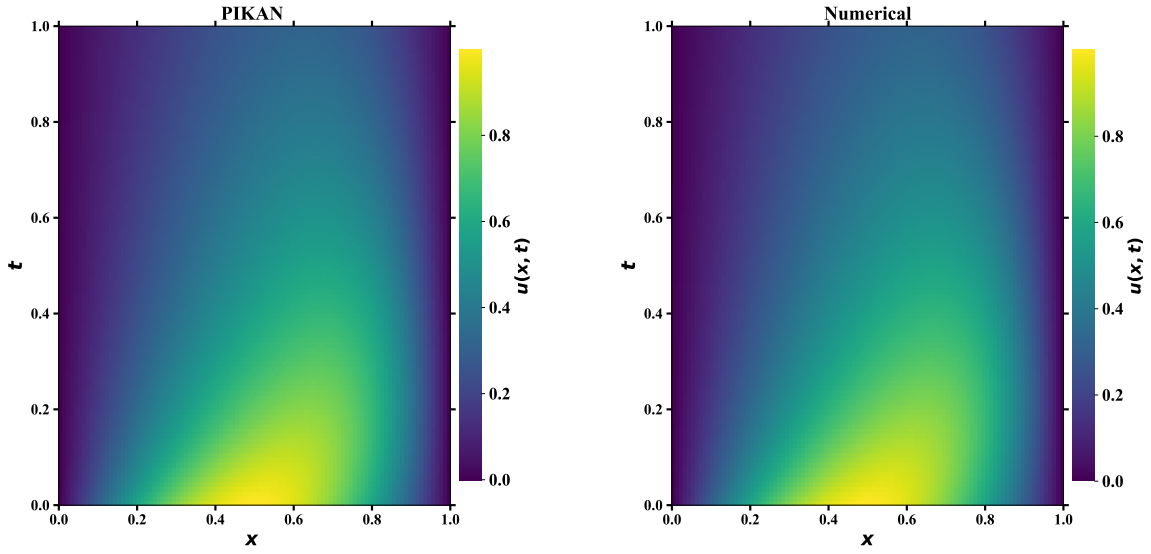


Figure 14: Comparison of the Burgers' equation solution $u(x, t)$ obtained using the proposed PIKAN and a standard numerical solver. The figure illustrates the evolution of $u(x, t)$ over the spatial domain $[0, 1]$, with the initial condition $u(x, 0) = \sin(\pi x)$ and Dirichlet boundary conditions $u(0, t) = u(1, t) = 0$.

7.2 Allen-Cahn Equation

7.2.1 Case - 1

The Allen-Cahn equation serves as a valuable model in materials science for elucidating phase separation and interface dynamics in multi-component alloy systems. This particular reaction-diffusion equation is employed to depict the progression of an order parameter that distinguishes between different phases of a material. In this section, we will delve into solutions for this highly nonlinear system using PIKAN (Hussain et al. (2019)).

Architecture	Spline-order	Grid size	Basis Activation	Loss
$[2, 12, 8, 12, 1]$	3	5	sin	10^{-6}

Table 11: Architecture details of the model used in section 7.2.1

The Allen-Cahn equation is provided below, and for this particular equation, we have selected a value of $\nu = 0.001$,

$$\begin{aligned} u_t &= \nu\, u_{xx} - u^3 + u, \\ &\begin{cases} u(x,0) & = 0.53x + 0.47\sin(-1.5\pi x) \\ u(1,t) & = 1 \\ u(-1,t) & = -1, \end{cases} \end{aligned} \tag{35}$$

where $x \in [-1, 1]$ and $t \in [0, 1]$.

The residual, physics loss, initial condition loss, boundary condition loss, and total loss are the following:

$$\begin{aligned} \mathcal{R}_\theta(x,t) &= \hat{u}_t + \hat{u}^3 - \hat{u} - \nu \hat{u}_{xx}, \\ \mathcal{L}_{\mathrm{r}} &= \frac{1}{N_r}\sum_{i=1}^{N_r} |\mathcal{R}_\theta(x_{\mathrm{r}}^i, t_{\mathrm{r}}^i)|^2, \\ \mathcal{L}_{\mathrm{ic}} &= \frac{1}{N_{ic}}\sum_{j=1}^{N_{ic}} (\hat{u}(x_{\mathrm{ic}}^j, 0) - u(x_{\mathrm{ic}}^j, 0))^2, \\ \mathcal{L}_{bc} &= \frac{1}{N_{bc}}\sum_{k=1}^{N_{bc}} \left[(\hat{u}(1, t_{\mathrm{bc}}^k) - 1)^2 + (\hat{u}(-1, t_{\mathrm{bc}}^k) + 1)^2 \right], \\ \mathcal{L}_{\mathrm{final}}^{DF} &= \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{ic}} + \mathcal{L}_{\mathrm{bc}}. \end{aligned} \tag{36}$$

We have utilized efficient-KAN to solve Eq. (35), employing $N_x = 100$ collocation points for space and $N_t = 100$ for time, uniformly spread over the interval $-1 \leq x \leq 1$ and $0 \leq t \leq 1$. The model is trained for 4000 epochs and to minimize the loss function we have used the AdamW optimizer with a learning-rate $\eta = 0.001$. At the end of learning, we have recorded the MSE loss on the order of 10^{-6}. Further details of the model is listed in Table 11. The comparison between the PI-KAN predicted solution and the numerically exact solution of Eq. (35) for different time is shown in Fig. 15a, 15b & 15c (Hussain et al. (2019)).

7.2.2 CASE - 2

Now, we will solve a slightly more challenging version of the Allen-Cahn equation (Raissi et al. (2019))

$$u_t = \nu u_{xx} - 5u^3 + 5u, \begin{cases} u(x,0) & = x^2\cos(\pi x) \\ u(1,t) & = 1 \\ u(-1,t) & = -1, \end{cases} \tag{37}$$

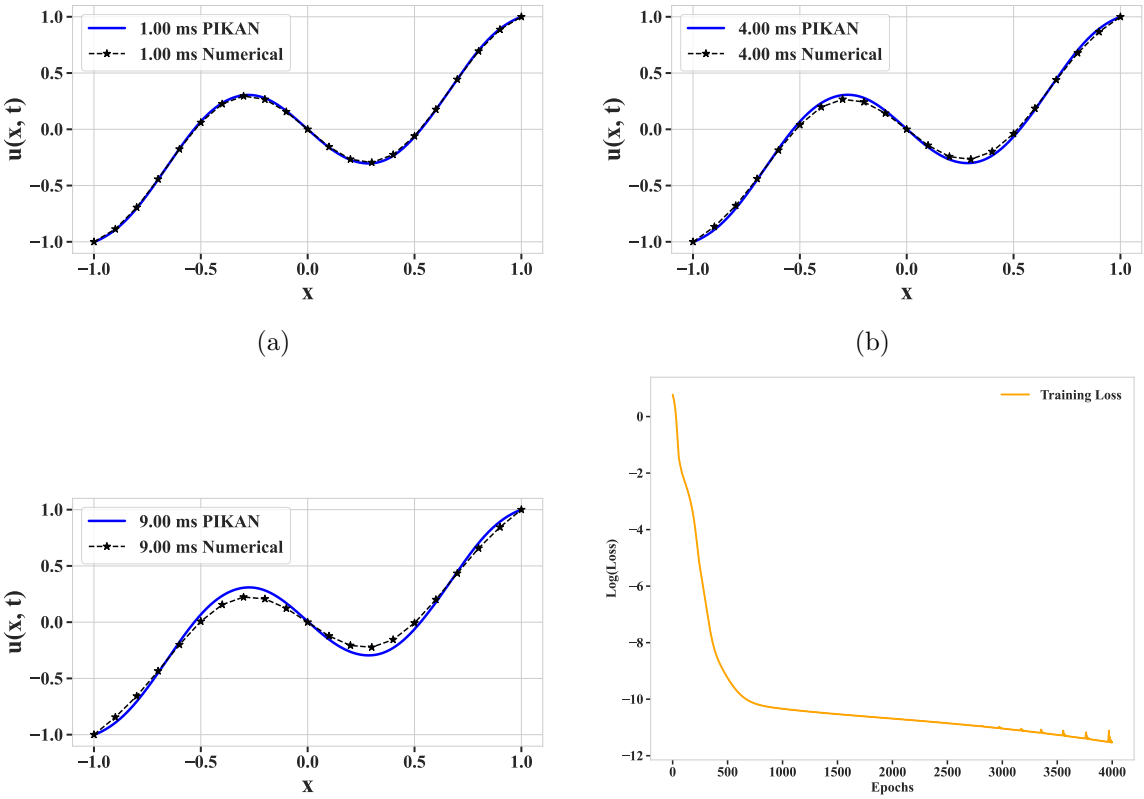


Figure 15: Panels (a) to (c) show the comparison between Allen-Cahn Case-1 PDE solutions obtained using the PIKAN method and the numerical solution obtained using the finite-difference method at $t = 1.00ms$, $t = 4.00ms$, and $t = 9.00ms$, respectively. Panel (d) illustrates the relationship between the loss (Eq. (35)) and epoch.

where $\nu = 0.0001$, $x \in [-1, 1]$, and $t \in [0, 1]$.

Architecture	Spline-order	Grid size	Basis Activation	Loss
$[2, 8, 4, 1]$	3	5	sin	0.08

Table 12: Architecture details of the model used in section 7.2.2.

The losses are

$$\begin{aligned}
\mathcal{R}_\theta(x,t) &= \hat{u}_t + 5\hat{u}^3 - 5\hat{u} - \nu\hat{u}_{xx},\\
\mathcal{L}_{\mathrm{r}} &= \frac{1}{N_r}\sum_{i=1}^{N_r} |\mathcal{R}_\theta(x_{\mathrm{r}}^i, t_{\mathrm{r}}^i)|^2,\\
\mathcal{L}_{\mathrm{ic}} &= \frac{1}{N_{ic}}\sum_{j=1}^{N_{ic}} (\hat{u}(x_{\mathrm{ic}}^j, 0) - (x_{\mathrm{ic}}^j)^2\cos(\pi x_{\mathrm{ic}}^j))^2,\\
\mathcal{L}_{\mathrm{bc}} &= \frac{1}{N_{bc}}\sum_{k=1}^{N_{bc}} \left[(\hat{u}(1, t_{\mathrm{bc}}^k) - 1)^2 + (\hat{u}(-1, t_{\mathrm{bc}}^k) + 1)^2\right],\\
\mathcal{L}_{\mathrm{data}} &= \frac{1}{N_{data}}\sum_{p=1}^{N_{data}} \left|\hat{u}(x_{\mathrm{data}}^p) - u(x_{\mathrm{data}}^p)\right|^2,\\
\mathcal{L}_{\mathrm{final}}^{DD} &= \mathcal{L}_{\mathrm{r}} + \mathcal{L}_{\mathrm{ic}} + \mathcal{L}_{\mathrm{bc}} + \mathcal{L}_{\mathrm{data}}. \qquad (38)
\end{aligned}$$

The solution of the Allen-Cahn equation (37) is obtained using the data-driven method (8). Initially, the numerical solution is computed, and then only 10% of the data from the numerical solution is randomly selected. An additional loss term (38) has been included in the loss function, corresponding to the Data-driven loss [Eqs. (38) and (8)]. To train the model, we have utilized the efficient-KAN with $N_x = 100$ and $N_t = 100$ collocation points uniformly distributed over the interval $-1 \le x \le 1$ and $0 \le t \le 1$. The model is trained for 20000 epochs, using the AdamW optimizer with a learning rate of $\eta = 0.001$ to minimize the loss function. At the end of training, a mean squared error loss on the order of 10^{-2} is recorded (see Fig. 16d). Additional details of the model are provided in Table 12. A comparison of the PIKAN predicted solution and the numerically exact solution of Eq. (37) for different time steps is shown in Fig. 16a, 16b & 16c. Figure 17 depicts the solution $u(x,t)$ for equation (37) in the domain $x \in [-1, 1]$ and $t \in [0, 1]$.

8. Comparing Physics-Informed Models: PINN and PIKAN

The PIKAN method marks a step forward in solving differential equations, offering improvements over the traditional PINN approach. A major advantage of PIKAN is its simplified hyperparameter tuning compared to PINNs. Further, PIKAN stands out for its faster convergence rates, enabling quicker and more reliable solutions. This is a crucial benefit, as it means complex differential equations can be tackled more efficiently and with higher accuracy.

To comprehensively assess the capabilities of physics-informed models, we evaluated performance across five commonly used challenging benchmark problems, each designed to highlight distinct computational difficulties. The Mathieu equation (section 6.3), with its

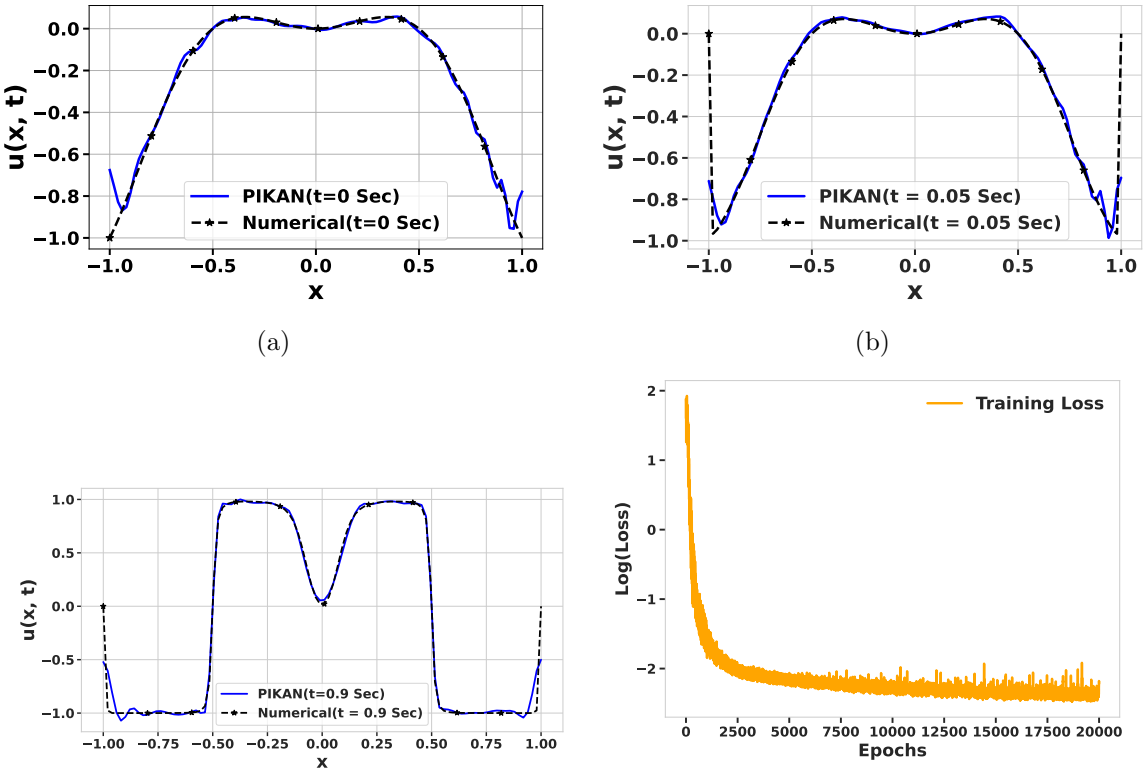


Figure 16: Panels (a) to (c) show the comparison between Allen-Cahn Case-2 PDE solutions obtained using the PIKAN method and the numerical solution obtained using the finite-difference method at $t = 0$, $t = 0.05$, and $t = 0.9$, respectively. Panel (d) illustrates the relationship between the loss (Eq. (37)) and epoch.

parametric forcing, serves to probe the stability of the model and the handling of periodic coefficients. The van der Pol oscillator (section 6.4), characterized by relaxation oscillations, presents a stiff system with fast-slow dynamics that tests the robustness of numerical methods. The Lorenz system (section 5.3), renowned for its chaotic behavior, evaluates short-horizon accuracy under conditions of extreme sensitivity to initial conditions. The Allen–Cahn equation (section 7.2.2), a reaction–diffusion problem, features thin interfaces and metastability, demanding precise resolution of sharp solution features. Finally, Burgers' equation 7.1, with its development of steep gradients and shocks, challenges models to accurately capture discontinuous phenomena. These five benchmarks cover a wide range of problems, including ODEs and PDEs, with solutions that can be smooth or sharp, and behaviors that range from stable to chaotic.

Before diving into the comparative analysis, it is crucial to understand the PIKAN architecture, which can be represented as a graph, where nodes denote summation operation and edges – learnable activation functions. Consider, as an example, the KAN architecture of form $[\mathbf{I}, \mathbf{a}, \mathbf{b}, \mathbf{O}]$, where $\mathbf{I}$ denotes the number of input variables, $\mathbf{a}$ represents the number

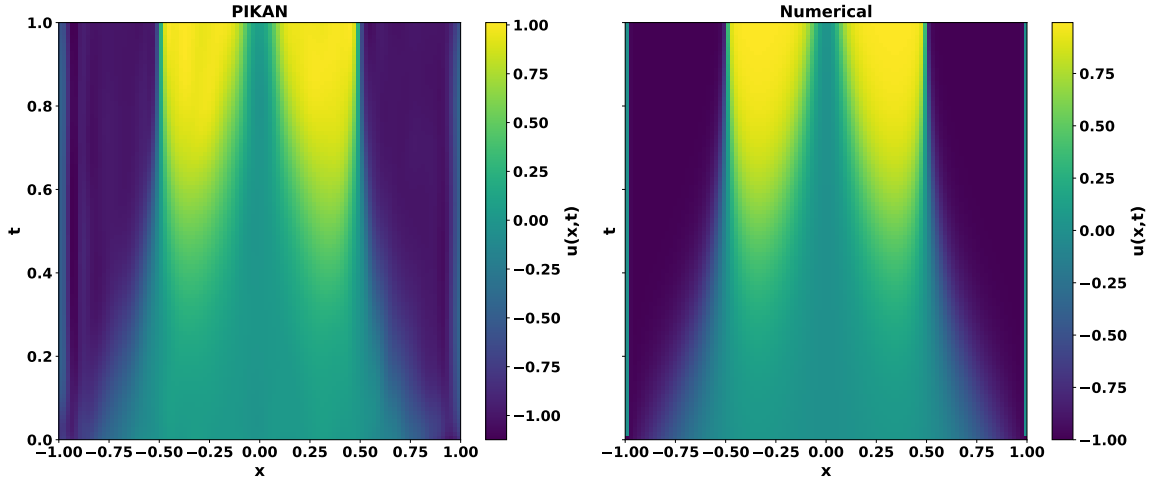


Figure 17: Side-by-side comparison of the Allen–Cohen equation solutions computed via PIKAN method (left) and a conventional finite-difference numerical scheme (right). Both panels display the solution $u(x,t)$ for equation (37) in the domain $x \in [-1,1]$ and $t \in [0,1]$. The close agreement between the two confirms the accuracy of the trained PIKAN model.

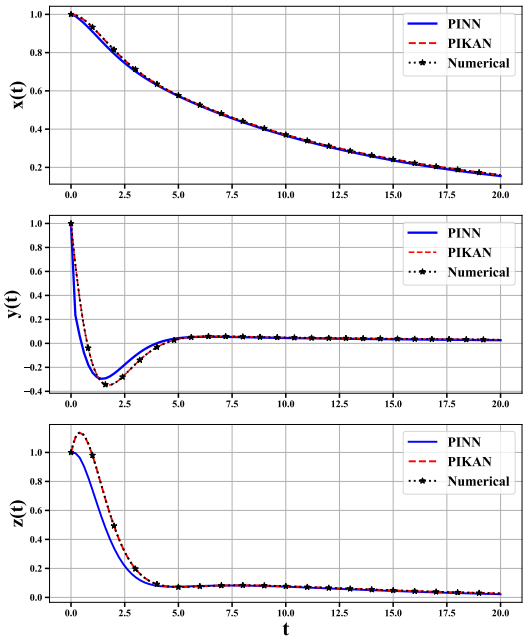
of nodes in the first hidden layer, **b** – the number of nodes in the second hidden layer, and **O** – the number of output nodes.

Let us now explore how PIKAN simplifies hyperparameter tuning compared to PINNs. First, for the case of the system of two linear equations (15), two separate neural networks have been trained in Uddin et al. (2023): The PINN model approximating $u(x)$ [Eq. (15)] has 20 neurons and 4 hidden layers, and the PINN model for $v(x)$ has 10 neurons and 5 hidden layers. A single PIKAN model presented in Section 5.1 with architecture $[1,2,3,2]$ (recall that there is only two intermediate layers) already performed well. For the system of equations represented by (17), the PINN model required 8 hidden layers with 8 neurons each, as detailed in Uddin et al. (2023). In contrast, the corresponding PIKAN model (in Section 5.2) achieved similar accuracy with just a single layer having an architecture of $[1,7,2]$.

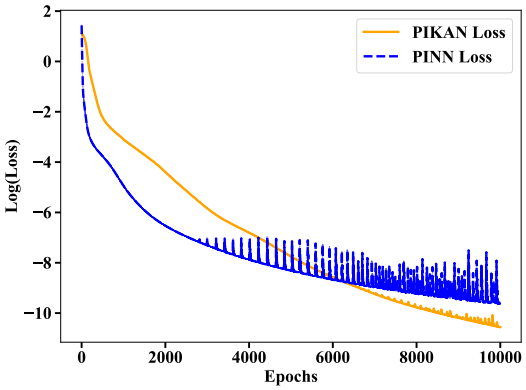
In Table 14 (see Figs. 18a, 19a, and 20a), we present a comparative analysis of five commonly used benchmarks evaluated using both PINN and PIKAN.

Processor	RAM	Operating System
12th Gen Intel® Core™ i5-1235U @ 1.30 GHz, 10 cores, 12 threads	16 GB	Windows 11

Table 13: System configuration used for all experiments.

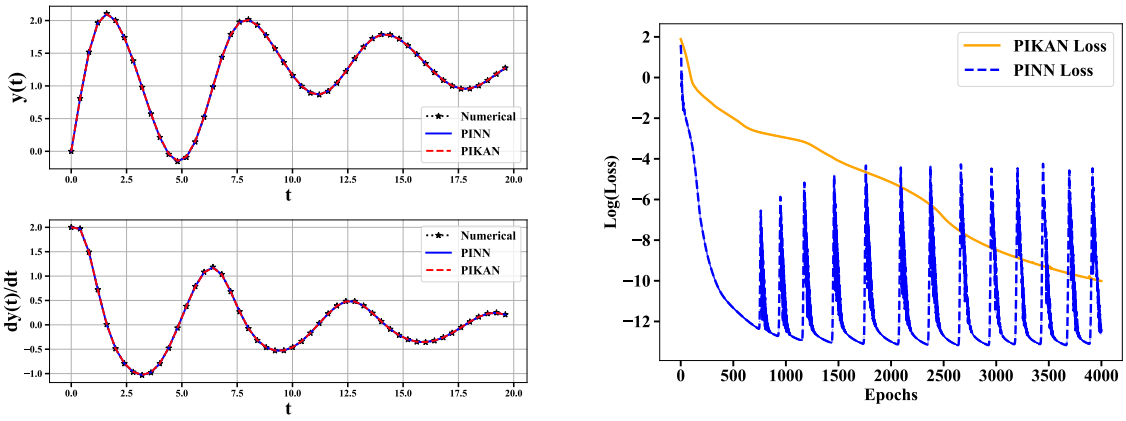


(a)



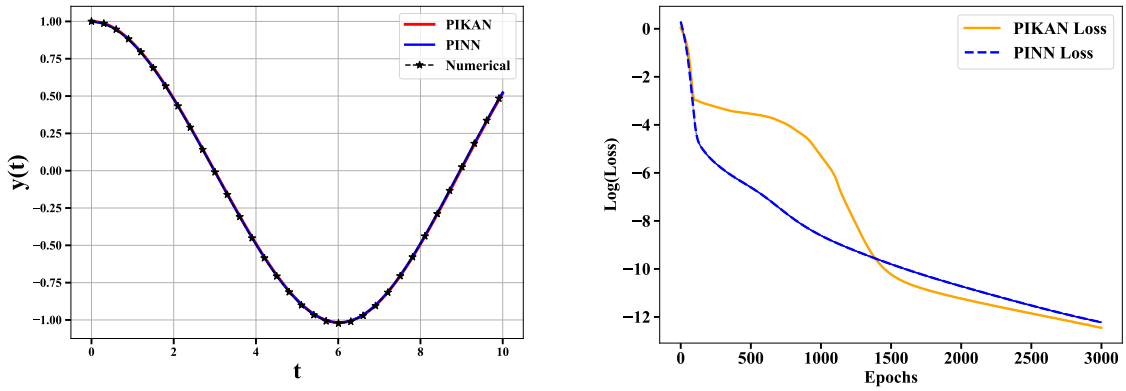
(b)

Figure 18: (a) Illustrates the Numerical vs PIKAN vs PINN sollution for $x(t), y(t), z(t)$ (b) Illustrates the training loss comparison between the PINN model and PIKAN model.



(a) (b)

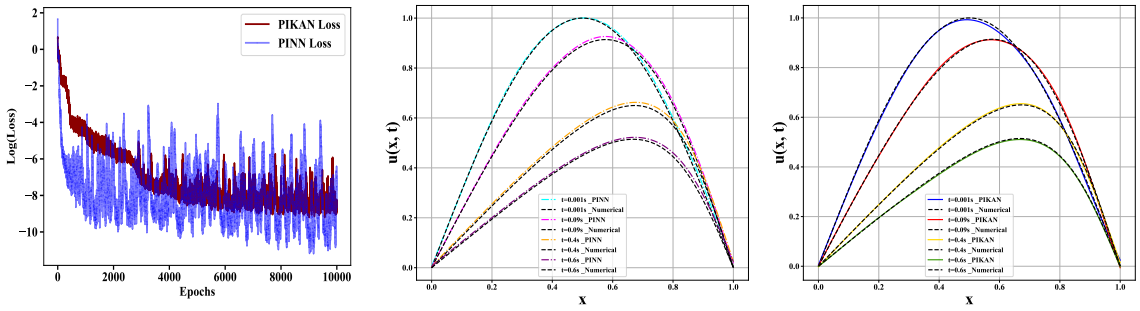
Figure 19: (a) Illustrates the comparison between the Numerical vs PINN vs PIKAN solution for two dynamical variables (b) Illustrates the training loss comparison between the PIKAN Model and PINN model.



(a) (b)

Figure 20: (a) Illustrates the Numerical vs PINN vs PIKAN sollution and (b) Illustrates the training loss vs Iterations curve of PIKAN and PINN.

As shown in Table 14, across the ODE benchmarks, PIKAN matches or surpasses the accuracy of standard PINNs while training substantially faster and with far more stable optimization dynamics. On the Mathieu equation, PIKAN attains a lower terminal residual (3.92×10^{-6} vs. 4.92×10^{-6}) in less than half the time (25.76 s vs. 58.48 s). The advantage widens on the chaotic Lorenz system, where PIKAN reduces the final loss by an order of magnitude (3.76×10^{-5} vs. 2.34×10^{-4}) and still converges sooner (124.58 s vs. 189.55 s). Burgers' equation shows the largest gap: PIKAN reaches 3.23×10^{-5} compared with 6.43×10^{-4} for PINN nearly a $20\times$ improvement while cutting wall-clock time by $\sim 1.6\times$



(a) (b) (c)

Figure 21: (a) Illustrates the relationship between loss (Eq. (34)) and epoch for both PIKAN and PINN, (b) Comparison between the PINN predicted solution and numerically exact solution of Eq. (32), (c) Comparison between the PIKAN predicted solution and the numerically exact solution of Eq. (32).

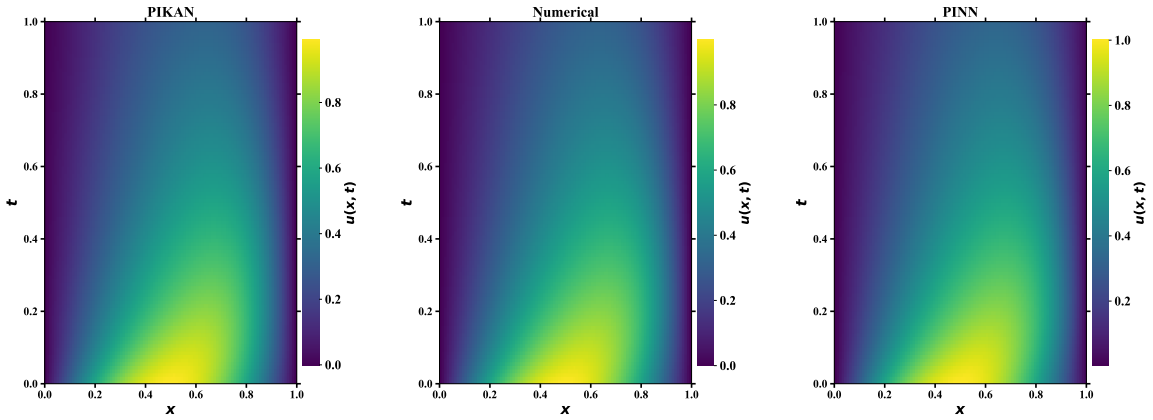


Figure 22: Comparison of the Burgers' equation solution $u(x,t)$ obtained using the proposed PINN, PIKAN and a standard numerical solver. The figure illustrates the evolution of $u(x,t)$ over the spatial domain $[0,1]$, with the initial condition $u(x,0) = \sin(\pi x)$ and Dirichlet boundary conditions $u(0,t) = u(1,t) = 0$.

(457.73 s vs. 746.21 s). The Van der Pol oscillator is the lone case where PINN achieves a smaller terminal residual ($\sim 10^{-6}$ vs. $\sim 10^{-5}$), yet the trajectories produced by PIKAN are indistinguishable from the numerical reference and PIKAN converges $\sim 3\times$ faster (86.12 s vs. 253.55 s). These quantitative gains are mirrored in the training curves: PIKAN's losses decay smoothly and monotonically, whereas PINN curves exhibit pronounced saw-tooth oscillations and plateaus, signaling optimizer-induced instabilities (Table 14; Figs. 18b–20b and 21a).

The PDE benchmarks listed in Table 14 reinforce this pattern. For the Allen–Cahn equation, PIKAN better tracks the finite-difference solution at $t = 0$, 0.05, and 0.9 with

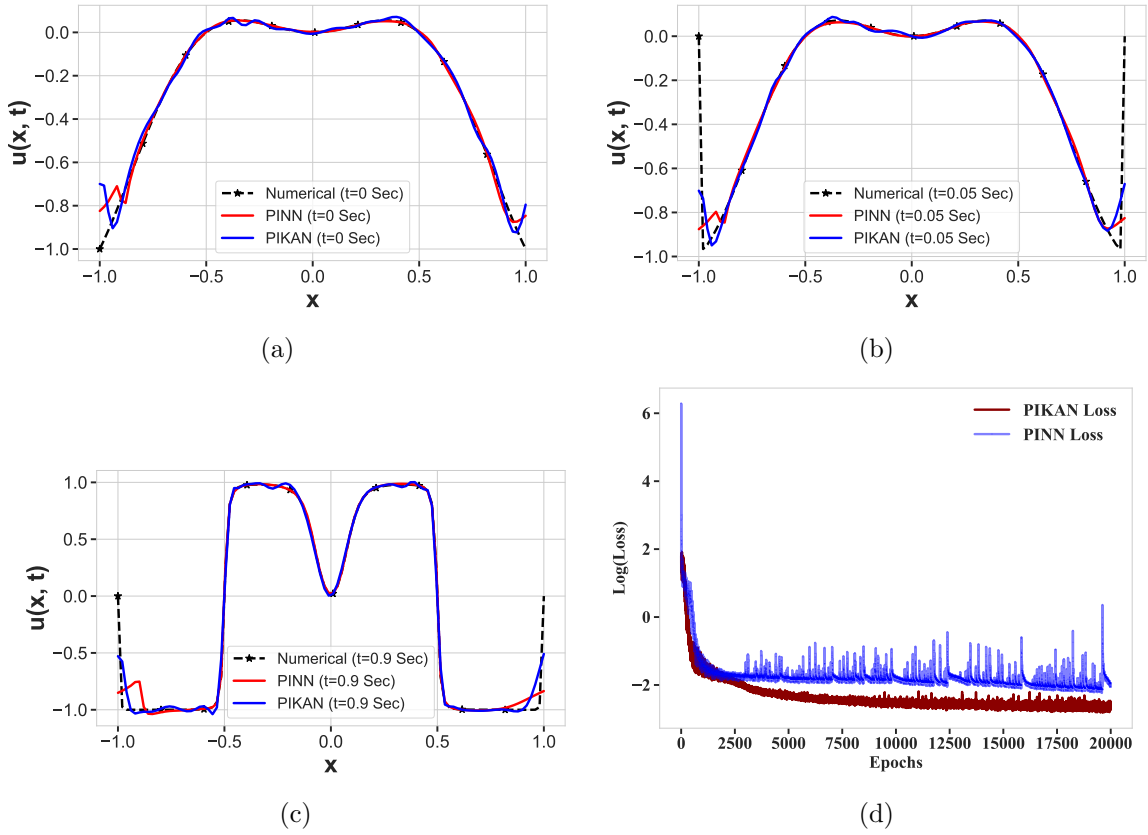


Figure 23: Panels (a) to (c) show the comparison of Allen-Cahn Case-2 PDE solutions obtained using PIKAN, PINN, and the finite-difference method at $t = 0$, $t = 0.05$, and $t = 0.9$, respectively. Panel (d) illustrates the relationship between the loss (Eq. (37)) and epoch comparing PINN and PIKAN approaches.

fewer boundary artifacts, settles to a lower and steadier loss floor, and completes training about 1.4× faster (2171.72 s vs. 3090.20 s) (Fig. 23d). For Burgers' PDE, PIKAN captures sharp features without overshoot and maintains a stable, steadily declining loss, in contrast to the high-amplitude oscillations observed for PINN (Figs. 21a– 23d). Crucially, these advantages are achieved with a compact architecture $[1, 12, 8, 1]$ for PIKAN versus the deeper, wider MLPs typically used for PINNs, yielding a better accuracy–efficiency trade-off. Summarizing across all the benchmarks, PIKAN delivers lower final residuals in three of four ODEs and across the PDEs, consistent speedups of roughly 1.4×–3×, and markedly improved training stability, establishing it as a superior alternative to standard PINNs for physics-informed learning as available in Figs. 18a– 23d.

Our results demonstrate not only the accuracy (alternatively, a lower loss) but also the computational efficiency of each model, revealing that KAN consistently outperforms the neural network in terms of inference speed across all test. Furthermore, it is shown in Cui et al. (2025) that when solving the Navier–Stokes equations, KAN converges quicker and achieves better accuracy than PINN. We believe that the superiority of PIKANs is due

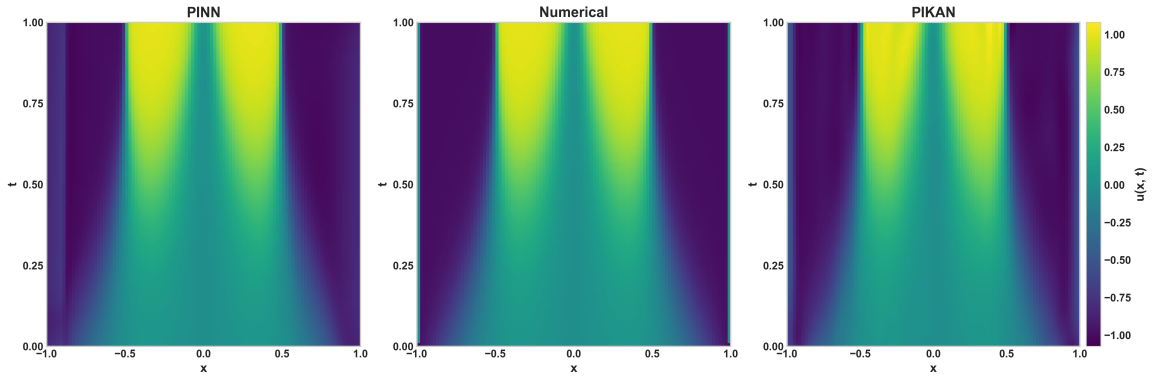


Figure 24: Side-by-side comparison of Allen–Cahn equation solutions computed via PINN (left panel), a conventional finite-difference scheme (center panel), and the PIKAN method (right panel). The three panels display the solution $u(x,t)$ for equation (37) in the domain $x \in [-1,1]$ and $t \in [0,1]$. The close agreement among all three solutions demonstrates the accuracy of the PIKAN approach.

Model	Type	Architecture	Optimizer	Epochs	Loss	Time
Mathieu Equation (28)	PINN	[1,128,64,32,16,1]	Adam	3000	4.92×10^{-6}	58.48s
	PIKAN	[1,12,8,1]	Adam	3000	3.92×10^{-6}	25.76s
Van der Pol Equation (30)	PINN	[1,128,128,128,64,32,16,1]	Adam	4000	4.22×10^{-6}	253.55s
	PIKAN	[1,12,8,1]	Adam	4000	4.49×10^{-5}	86.12s
Lorenz Equation (21)	PINN	[1,128,64,32,3]	Adam	10000	2.34×10^{-4}	189.55s
	PIKAN	[1,12,8,3]	Adam	10000	3.76×10^{-5}	124.58s
Allen-Cahn Equation (37)	PINN	[2, 512, 512, 256, 128, 64, 16, 1]	Adam	20000	1.50×10^{-1}	3090.20s
	PIKAN	[2,8,6,1]	Adam	20000	8.16×10^{-2}	2171.72s
Burger- Equation (32)	PINN	[2,64,128,32,16,1]	Adam	10000	6.43×10^{-4}	746.21s
	PIKAN	[2,8,4,1]	Adam	10000	3.23×10^{-5}	457.73s

Table 14: Comparisons of PINN and PIKAN.

to the employment of adaptive activation functions. Moreover, identifying an appropriate architecture for PIKAN is significantly more straightforward compared to PINNs, thereby reducing the time and effort typically required for architecture tuning. This advantage holds greater practical importance than mere computational efficiency, especially given that architecture search in PINNs is often time-consuming. For example, in the case of the Van der Pol equation (Sec. 6.4), the optimal PINN architecture involved numerous hidden layers and required considerable experimentation to arrive at a satisfactory configuration. Therefore, a fair and comprehensive comparison between PINN and PIKAN should not be limited to computational efficiency alone, but should also account for architectural simplicity, which directly translates to reduced development time and effort.

9. Conclusion

We have demonstrated that physics-informed modeling using KAN via efficient-KAN and wave-KAN are versatile and efficient tools. We showcased their performance on a series of

differential equations using data-free and data-driven approaches and conducted a comparative analysis in terms of architectural complexity and performance with PINN.

10. Data availability

All codes used in this study can be found in `https://github.com/AI-and-Quantum-Computing/PIKAN`.

The system configuration used by us are summarized in table 13.

Acknowledgments

D.I.B and A.S. are supported by Army Research Office (ARO) (grant W911NF-23-1-0288; program manager Dr. James Joseph). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARO or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Harry Bateman. Some recent researches on the motion of fluids. *Monthly Weather Review*, 43(4):163–170, April 1915. ISSN 1520-0493. doi: 10.1175/1520-0493(1915)43⟨163:srrotm⟩2.0.co;2. URL http://dx.doi.org/10.1175/1520-0493(1915)43<163:SRROTM>2.0.CO;2.

J Bec and K Khanin. Burgers turbulence. *Physics Reports*, 447(1–2):1–66, August 2007. ISSN 0370-1573. doi: 10.1016/j.physrep.2007.04.002. URL http://dx.doi.org/10.1016/j.physrep.2007.04.002.

Blealtan. Efficient kan: A memory-efficient kan implementation. https://github.com/Blealtan/efficient-kan, 2024.

Mayur P Bonkile, Ashish Awasthi, C Lakshmi, Vijitha Mukundan, and V S Aswin. A systematic literature review of burgers' equation with recent advances. *Pramana*, 90(6), April 2018. ISSN 0973-7111. doi: 10.1007/s12043-018-1559-4. URL http://dx.doi.org/10.1007/s12043-018-1559-4.

Zavareh Bozorgasl and Hao Chen. Wav-kan: Wavelet kolmogorov-arnold networks, 2024. URL https://arxiv.org/abs/2405.12832.

Jürgen Braun and Michael Griebel. On a constructive proof of kolmogorov's superposition theorem. *Constructive Approximation*, 30(3):653–675, May 2009. ISSN 1432-0940. doi: 10.1007/s00365-009-9054-2. URL http://dx.doi.org/10.1007/s00365-009-9054-2.

J.M. Burgers. *A Mathematical Model Illustrating the Theory of Turbulence*, page 171–199. Elsevier, 1948. doi: 10.1016/s0065-2156(08)70100-5. URL http://dx.doi.org/10.1016/S0065-2156(08)70100-5.

Shuangwei Cui, Manshu Cao, Yifeng Liao, and Jianing Wu. Physics-informed kolmogorov–arnold networks: Investigating architectures and hyperparameter impacts for solving Navier–Stokes equations. *Physics of Fluids*, 37(3), March 2025. doi: 10.1063/5.0257677. URL http://dx.doi.org/10.1063/5.0257677.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3), July 2022. ISSN 1573-7691. doi: 10.1007/s10915-022-01939-z. URL http://dx.doi.org/10.1007/s10915-022-01939-z.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989. ISSN 1435-568X. doi: 10.1007/bf02551274. URL http://dx.doi.org/10.1007/BF02551274.

Vikas Dwivedi, Nishant Parashar, and Balaji Srinivasan. Distributed learning machines for solving forward and inverse problems in partial differential equations. *Neurocomputing*, 420:299–316, January 2021. ISSN 0925-2312. doi: 10.1016/j.neucom.2020.09.006. URL http://dx.doi.org/10.1016/j.neucom.2020.09.006.

Daniele Fakhoury, Emanuele Fakhoury, and Hendrik Speleers. Exsplinet: An interpretable and expressive spline-based neural network. *Neural Networks*, 152:332–346, 2022.

Olga Fuks and Hamdi A. Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1):19–37, 2020. ISSN 2689-3967. doi: 10.1615/jmachlearnmodelcomput.2020033905. URL http://dx.doi.org/10.1615/JMachLearnModelComput.2020033905.

Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, January 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90003-8. URL http://dx.doi.org/10.1016/0893-6080(89)90003-8.

Alexander B. Givental, Boris A. Khesin, Jerrold E. Marsden, Alexander N. Varchenko, Victor A. Vassiliev, Oleg Ya. Viro, and Vladimir M. Zakalyukin, editors. *On the representation of functions of several variables as a superposition of functions of a smaller number of variables*, pages 25–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-01742-1. doi: 10.1007/978-3-642-01742-1_5. URL https://doi.org/10.1007/978-3-642-01742-1_5.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL http://dx.doi.org/10.1016/0893-6080(89)90020-8.

Archie J. Huang and Shaurya Agarwal. On the limitations of physics-informed deep learning: Illustrations using first-order hyperbolic conservation law-based traffic flow models. *IEEE Open Journal of Intelligent Transportation Systems*, 4:279–293, 2023. ISSN 2687-7813. doi: 10.1109/ojits.2023.3268026. URL http://dx.doi.org/10.1109/OJITS.2023.3268026.

Safdar Hussain, Abdullah Shah, Sana Ayub, and Asad Ullah. A approximate analytical solution of the allen-cahn equation using homotopy perturbation method and homotopy analysis method. *Heliyon*, 5(12), 2019.

Weiqi Ji, Weilun Qiu, Zhiyu Shi, Shaowu Pan, and Sili Deng. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, 125 (36):8098–8106, August 2021. ISSN 1520-5215. doi: 10.1021/acs.jpca.1c05102. URL http://dx.doi.org/10.1021/acs.jpca.1c05102.

K.N.S Kasi Viswanadham. Coupled system of boundary value problems by galerkin method with cubic b-splines. *E3S Web of Conferences*, 128:09008, 2019. ISSN 2267-1242. doi: 10.1051/e3sconf/201912809008. URL http://dx.doi.org/10.1051/e3sconf/201912809008.

Benjamin C. Koenig, Suyong Kim, and Sili Deng. KAN-ODEs: Kolmogorov-Arnold Network Ordinary Differential Equations for Learning Dynamical Systems and Hidden Physics, July 2024. arXiv:2407.04192.

Andrei Nikolaevich Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR*, 114(5):953–956, 1957.

Ivana Kovacic, Richard Rand, and Si Mohamed Sah. Mathieu's equation and its generalizations: Overview of stability charts and their features. *Applied Mechanics Reviews*, 70(2), February 2018. ISSN 2379-0407. doi: 10.1115/1.4039144. URL http://dx.doi.org/10.1115/1.4039144.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 26548–26560. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf.

Ming-Jun Lai and Zhaiming Shen. The kolmogorov superposition theorem can break the curse of dimensionality when approximating high dimensional functions, 2021. URL https://arxiv.org/abs/2112.09963.

Zaharaddeen Karami Lawal, Hayati Yassin, Daphne Teck Ching Lai, and Azam Che Idris. Physics-informed neural network (pinn) evolution and beyond: A systematic literature review and bibliometric analysis. *Big Data and Cognitive Computing*, 6(4):140, November 2022. ISSN 2504-2289. doi: 10.3390/bdcc6040140. URL http://dx.doi.org/10.3390/bdcc6040140.

Pierre-Emmanuel Leni, Yohan D Fougerolle, and Frédéric Truchetet. The kolmogorov spline network for image processing. In *Image Processing: Concepts, Methodologies, Tools, and Applications*, pages 54–78. IGI Global, 2013.

Ji-Nan Lin and Rolf Unbehauen. On the realization of a kolmogorov network. *Neural Computation*, 5(1):18–20, 1993.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks, 2024. URL https://arxiv.org/abs/2404.19756.

Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20 (2):130–141, March 1963. ISSN 1520-0469. doi: 10.1175/1520-0469(1963)020⟨0130:dnf⟩2.0.co;2. URL http://dx.doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

Emile Mathieu. Mémoire sur le mouvement vibratoire d'une membrane de forme elliptique. *Journal de Mathematiques Pures et Appliquees*, 13:137–203, 1868. URL http://eudml.org/doc/234720.

Hadrien Montanelli and Haizhao Yang. Error bounds for deep relu networks using the kolmogorov–arnold superposition theorem. *Neural Networks*, 129:1–6, 2020.

Jared O'Leary, Joel A. Paulson, and Ali Mesbah. Stochastic physics-informed neural ordinary differential equations. *Journal of Computational Physics*, 468:111466, November 2022. ISSN 0021-9991. doi: 10.1016/j.jcp.2022.111466. URL http://dx.doi.org/10.1016/j.jcp.2022.111466.

Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, January 2019. ISSN 1095-7197. doi: 10.1137/18m1229845. URL http://dx.doi.org/10.1137/18M1229845.

Jamshaid Ul Rahman, Sana Danish, and Dianchen Lu. Oscillator simulation with deep neural networks. *Mathematics*, 12(7):959, March 2024. ISSN 2227-7390. doi: 10.3390/math12070959. URL http://dx.doi.org/10.3390/math12070959.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL http://dx.doi.org/10.1016/j.jcp.2018.10.045.

Haakon Robinson, Suraj Pawar, Adil Rasheed, and Omer San. Physics guided neural networks for modelling of non-linear dynamics. *Neural Networks*, 154:333–345, October 2022. ISSN 0893-6080. doi: 10.1016/j.neunet.2022.07.023. URL http://dx.doi.org/10.1016/j.neunet.2022.07.023.

Franz M. Rohrhofer, Stefan Posch, Clemens Gößnitzer, and Bernhard C Geiger. On the role of fixed points of dynamical systems in training physics-informed neural networks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=56cTmVrg5w.

Lawrence Ruby. Applications of the mathieu equation. *American Journal of Physics*, 64 (1):39–44, January 1996. ISSN 1943-2909. doi: 10.1119/1.18290. URL http://dx.doi.org/10.1119/1.18290.

Khemraj Shukla, Juan Diego Toscano, Zhicheng Wang, Zongren Zou, and George Em Karniadakis. A comprehensive and fair comparison between mlp and kan representations for differential equations and operator networks, 2024. URL https://arxiv.org/abs/2406.02917.

David A. Sprecher and Sorin Draghici. Space-filling curves and kolmogorov superposition-based neural networks. *Neural Networks*, 15(1):57–67, January 2002. ISSN 0893-6080. doi: 10.1016/s0893-6080(01)00107-1. URL http://dx.doi.org/10.1016/s0893-6080(01)00107-1.

Ziya Uddin, Sai Ganga, Rishi Asthana, and Wubshet Ibrahim. Wavelets based physics informed neural networks to solve non-linear differential equations. *Scientific Reports*, 13(1), February 2023. ISSN 2045-2322. doi: 10.1038/s41598-023-29806-3. URL http://dx.doi.org/10.1038/s41598-023-29806-3.

Olivier Vallee and E. Moreau. The burgers equation as an electrodynamic model in plasma physics. *High Temperature Material Processes (An International Quarterly of High-Technology Plasma Processes)*, 11(4):611–617, 2007. ISSN 1093-3611. doi: 10.1615/hightempmatproc.v11.i4.130. URL http://dx.doi.org/10.1615/HighTempMatProc.v11.i4.130.

Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An expert's guide to training physics-informed neural networks, 2023. URL https://arxiv.org/abs/2308.08468.

Yizheng Wang, Jia Sun, Jinshuai Bai, Cosmin Anitescu, Mohammad Sadegh Eshaghi, Xiaoying Zhuang, Timon Rabczuk, and Yinghua Liu. Kolmogorov–Arnold Informed neural network: A physics-informed deep learning framework for solving PDEs based on Kolmogorov–Arnold networks, June 2024. URL http://arxiv.org/abs/2406.11045. arXiv:2406.11045 [cs, math].

Sunwoong Yang, Hojin Kim, Yoonpyo Hong, Kwanjung Yee, Romit Maulik, and Namwoo Kang. Data-driven physics-informed neural networks: A digital twin perspective, 2024. URL https://arxiv.org/abs/2401.08667.

Lei Yuan, Yi-Qing Ni, Xiang-Yun Deng, and Shuo Hao. A-pinn: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. *Journal of Computational Physics*, 462:111260, August 2022. ISSN 0021-9991. doi: 10.1016/j.jcp.2022.111260. URL http://dx.doi.org/10.1016/j.jcp.2022.111260.