**A Novel Approach Using Kolmogorov-Arnold Networks for Solving Non-linear Ordinary Differential Equations**

Morteza Farrokhnejad1[](https://orcid.org/0009-0004-6231-2402), Ali Farrokhnejad1[](https://orcid.org/my-orcid?orcid=0009-0008-0749-3377), Ahmet Rizaner2[](https://orcid.org/0000-0001-8419-5308)

1Department of Computer Engineering, Eastern Mediterranean University, 99628, Famagusta, North Cyprus, via Mersin 10, Turkey

2Department of Information Technology, School of Computing and Technology, Eastern Mediterranean University, 99628, Famagusta, North Cyprus, via Mersin 10, Turkey

ARTICLE INFO

-----------------------------

Keywords: Kolmogorov-Arnold Networks (KAN), Ordinary Differential Equations (ODEs), Function Approximation, Neural Network Architectures, Computational Mathematics, Machine Learning in Mathematical Modeling

**Abstract**

Ordinary Differential Equations (ODEs) are crucial in modeling a plethora of scientific and engineering systems, but their solution is often computationally challenging, especially when closed-form solutions are unavailable. Traditional numerical methods such as Runge-Kutta and finite differences have been employed, however, such methodologies are accompanied by constraints regarding their efficiency and associated computational expenses. This manuscript explores the application of the Kolmogorov-Arnold Network (KAN) for the numerical resolution of first-order ODEs, an innovative based on the Kolmogorov-Arnold theorem (KAT). KAN's distinctive approach for function approximation, using a decomposition of multivariate functions into univariate components, allows for high precision with fewer parameters and expedited convergence than traditional neural networks such as Radial Basis Function Neural Networks (RBFNNs) and Wavelet Neural Networks (WNNs). The efficacy of the KAN model is substantiated through comparison with other methods on example ODEs, showing comparable, if not superior accuracy. The results indicate that KAN presents a promising alternative for solving complex ODEs, offering enhanced computational efficiency and robustness over traditional techniques and other neural network models.

1. **Introduction**

Ordinary Differential Equations (ODEs) serve as crucial instruments in the mathematical modeling and analytical study of various scientific and engineering systems. They normally emerge in diverse applications, including fluid dynamics, chemical reaction kinetics, population dynamics, and structural analysis [1 - 8]. The inherent complexity of ODEs presents significant challenges in their solution, as numerous cases do not yield closed-form solutions, which makes the utilization of numerical or approximation techniques necessary. Currently available numerical methodologies, including the Runge-Kutta technique, finite difference approach, and shooting method, have historically been used to address these challenges. However, their drawbacks, such as considerable computational demands and the inability to produce closed-form solutions have motivated the investigation of alternative strategies for ODE resolution [1 - 3, 5 - 10].

The introduction of Artificial Neural Networks (ANNs) has facilitated novel methodologies for the numerical resolution of ODEs by recontextualizing the problem as an optimization framework. Preliminary investigations have demonstrated the efficacy of Multilayer Perceptrons (MLPs) in approximating solutions to both Initial Value Problems (IVPs) and Boundary Value Problems (BVPs). Based on neural networks, these methods are much more efficient than the classical numerical techniques. In particular, ANNs are capable of formulating analytical solutions which eliminate the necessity for performing interpolation over discretized computational intervals, hence more flexibility in solving IVPs and BVPs [1, 3, 6, 8, 11 - 17]. On the other hand, the first generation of models based on ANNs had several challenges [7] among which were the pronounced vulnerability to convergence at local minima and suboptimal rates of convergence [1].

As a solution to the shortcomings of the ANNs mentioned, new generation advanced architectures such as Radial Basis Function Neural Networks (RBFNNs) [2, 3, 9] and Wavelet Neural Networks (WNNs) [1, 7, 8] have emerged. These approaches have been known to have shorter convergence times and a higher accuracy compared to traditional techniques when applied to complex expressions of differential equations. Additionally, WNNs have attracted considerable interest, since their activation functions are concentrated so that the size of the network can be kept small which allows faster training while preserving the ability of any approximation that is expected by neural networks [1, 8]. Furthermore, the implementation of advanced training methodologies, such as Extreme Learning Machines (ELM) and metaheuristic optimization techniques, including Particle Swarm Optimization (PSO), has substantially enhanced both the efficiency and effectiveness (accuracy) of these neural network models [1, 2, 4, 5, 14].

In response to this assertion, the Kolmogorov-Arnold Network (KAN) architecture showcases a novel architecture which is efficient for function approximation. This architecture is promising for solving ODEs. The KAN model is based on the Kolmogorov-Arnold representation theorem which states that every multivariate continuous function can be expressed as a finite sum of univariate functions [6, 10 - 12, 15 - 29]. This inbuilt universality renders KAN particularly adept at approximating intricate mathematical models, including those characterized by ODEs [6, 11, 12, 16, 17, 19, 28].

This paper attempts to overcome the shortcomings that are posed by the current neural network architectures on differential equations by capitalizing on KAN’s systematic approach to function decomposition. The primary aim of the study is to make use of the KAN architecture for the approximation of the solutions of first order ODEs. This investigation is a major breakthrough in the fusion of sophisticated machine learning techniques with computational mathematics. The ANN-based approach is contrasted to the KAN structure, which is able to build a process-specific problem space and in this way improve the approximation of the results [17, 19, 20, 21, 23, 24, 26, 28, 29]. It is also different from any other design in that the network can become more precise with a decrease in the number of parameters, which in turn makes it relatively faster while preserving the accuracy [6, 10 - 12, 15, 17, 19 - 22, 24 - 26, 29].   
The rationale behind the implementation of KAN in this framework is the existed capabilities to handle the critical ODEs solution. Mostly, first order ODEs are noted to be not serious with the many complicated boundary conditions. However, they may now and then show some anomalously typical of nonlinear dynamics which affects the conventional numeric technique [6, 12, 17 - 19]. The unique modification of KAN, characteristic of the agility of the system to the situation, along with an expressive mode of representation of the mentioned challenges, definitely leads to very good results in their solution. Furthermore, KAN can be easily upgraded by including advanced optimization algorithms [19, 10, 20, 22 - 29], hence it is enhanced in solving ODEs with its robustness.

Recent investigations emphasize the efficacy of neural network architectures in the resolution of differential equations. Specifically, WNNs, when enhanced through sophisticated optimization techniques such as the butterfly optimization algorithm, exhibit superior capabilities in approximating solutions to ODEs. [1, 8]. Moreover, RBFNNs trained via extreme learning methodologies demonstrate the high rates of convergence and high accuracy regarding fractional differential equations [2, 3, 9]. These discoveries demonstrate the resurgence and importance of neural network models in computational mathematics.

Notwithstanding the advancements made in this field, significant deficiencies persist in the literature concerning the application of KAN to ODEs. Although the Kolmogorov-Arnold Theorem (KAT) offers a theoretical framework for function approximation [10, 15 - 17, 19, 20, 21 - 23, 26, 28, 29], its practical deployment for the resolution of ODEs remains insufficiently investigated. This research endeavors to fill this lacuna by executing a thorough assessment of KAN's effectiveness in solving first-order ODEs. Through methodical experimentation, this study aims to validate KAN as a robust and efficient methodology for function approximation specifically within the context of differential equations.

The implications of this research transcend the direct utilization of KAN in the context of ODEs. By establishing its efficacy as a versatile function approximator, this investigation enriches the field of computational mathematics and neural network-based modeling. The findings derived from this study are anticipated to guide the advancement of next-generation computational methodologies adept at solving intricate scientific and engineering challenges, consequently, the KAN constitutes a significant progression in the application of machine learning techniques for the resolution of differential equations. Its unique architectural framework and theoretical foundations establish it as a suitable alternative to prevailing ANN methodologies for function approximation. This research aims to enhance the current capabilities of neural network-based approaches in approximating first-order ODE solutions using KAN, thereby facilitating advancements in computational mathematics and other related fields.

The content of this paper is organized as follows: this paper continues by detailing the theoretical foundations and architecture of the KAN, highlighting its derivation from the KAT and its efficiency for function approximation. It then outlines the methodology for applying KAN to solve ODEs, including the formulation of the trial solution and the error minimization approach. This is followed by a series of numerical examples that compare the performance of the KAN model with traditional numerical methods and other neural network architectures. The paper concludes with a discussion of the results, implications for computational mathematics, and suggestions for future research directions.

1. **Kolmogorov-Arnold Model**

The KAN model is optimally configured for function approximation tasks, including the resolution of ODEs, owing to its basis in the KAT [10, 15 - 17, 19 - 23, 26, 28, 29]. This theorem suggests that any continuous multivariate function can be expressed as a finite summation of univariate functions subjected to addition, as such:

|  |  |
| --- | --- |
|  | (1) |

Where and are univariate functions and is the number of variables. The definition of KAN was derived based on this theorem, with the assumption that the theorem represents a KAN with the shape , having inputs, followed by 2 layers with and nodes, respectively, and 1 output. This foundation on the KAT facilitates the ability of a KAN to approximate intricate functions with reduced network depth [6, 10 - 12, 15 - 17, 19, 20 - 23, 25 - 29].

In a KAN, activation functions are placed on the edges that connect nodes from successive layers. The activation function between nodes and is defined as , which has two components:

|  |  |
| --- | --- |
|  | (2) |

where is the basis function, typically set as sigmoid linear unit (silu):

|  |  |
| --- | --- |
|  | (3) |

and is a linear combination of B-splines:

|  |  |
| --- | --- |
|  | (4) |

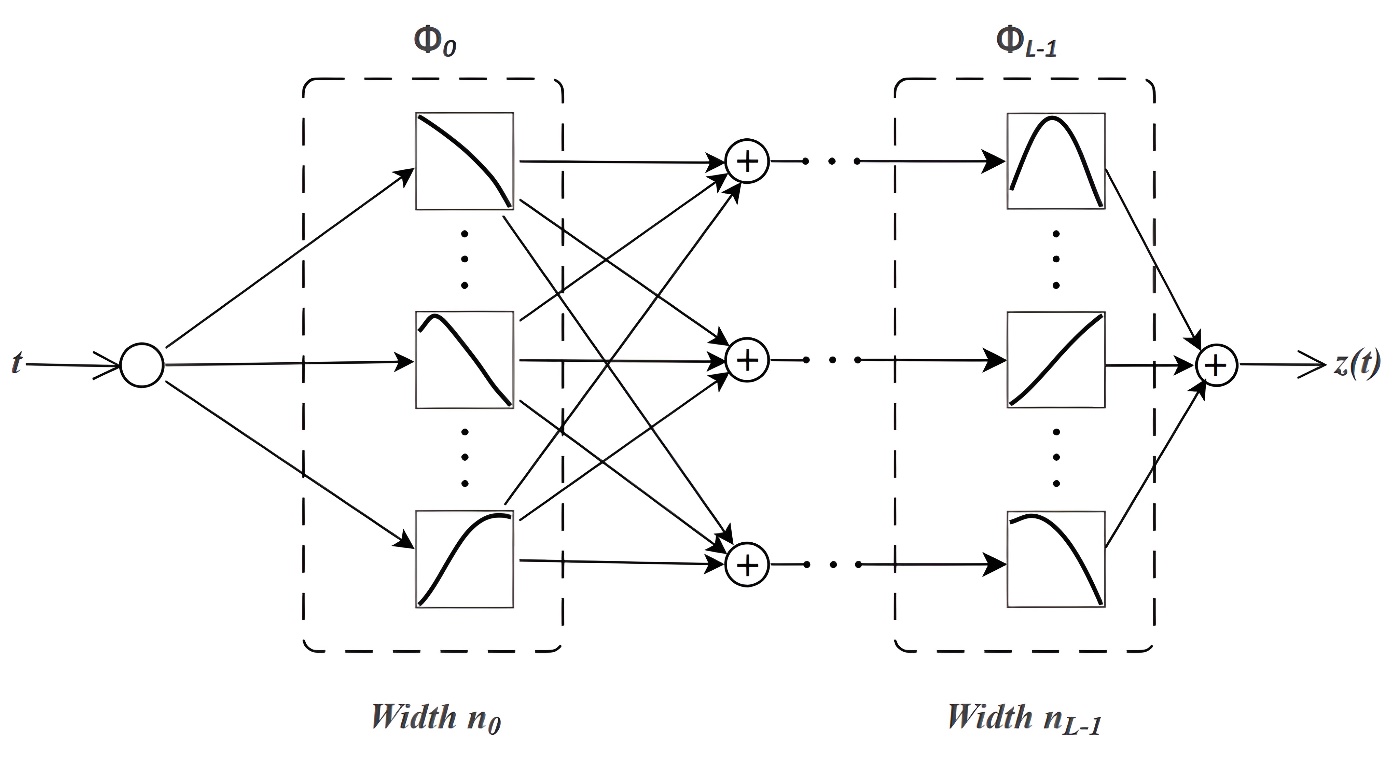
where , , and are adjusted through training, is the number of grid intervals, set at the time of definition of the B-splines, and is an individual B-spline curve at interval .

Assuming a KAN with layers, the output, denoted as for a given input vector , could then be calculated in a similar fashion to traditional machine learning models:

|  |  |
| --- | --- |
|  | (5) |

A multiplication of every , being the activation function matrices for layer , and the input vector [19].

Figure 1 illustrates the general KAN architecture with layers, having the shape , where is the number of nodes in the -th layer.



**Figure 1.** General KAN Architecture

KANs inherently diminish the computational complexity associated with multivariate functions while preserving accuracy [6, 10 - 12, 15, 17 - 22, 24 - 26, 29], a critical factor for accurately modeling the complex dynamics of ODEs. KANs are architected to optimize the advantages of the KAT by structuring layers such that univariate basis functions are hierarchically composed [10, 15, 17, 19, 20 - 28], resulting in outputs that effectively approximate multivariate functions. In contrast to conventional MLPs, which depend on universal approximation via dense layers and nonlinear activation functions, KANs leverage the structural organization offered by KAT to attain efficient and precise function representations [6, 10 - 12, 15 - 29]. The hidden layers of the network typically utilize Gaussian Radial Basis Functions (RBFs) as activation functions, selected for their smoothness properties and capacity for spatial localization of approximations [6, 17, 19, 23, 25, 26]. These RBFs facilitate a concentration of response from each hidden layer neuron to distinct regions of the input space, which is essential for the accurate resolution of ODEs where localized dynamics predominantly influence system behavior. In contrast to WNNs, which employ wavelet transformations to achieve a compact topology and facilitate efficient training, KANs present an alternative framework founded on the theoretical assurances provided by the KAT [10, 17, 19, 20 - 29]. While both WNNs and RBF networks demonstrate proficiency in distinct application domains [1 - 3, 9], the hierarchical univariate decomposition characteristic of KANs is inherently more compatible with the requirements associated with ODE approximation [6, 26, 28]. This congruence enables KANs to deliver accurate gradient evaluations, a feature that is particularly beneficial for integration with differentiable ODE solvers. These solvers exploit the structured outputs of KANs to simulate dynamical systems and extract latent physical phenomena while incurring minimal computational overhead.

A notable advantage of utilizing KANs is their ability to process high-dimensional input data effectively. The application of the superposition principle within KAT mitigates the complexity associated with high dimensionality by decomposing intricate functions into simpler, constituent components. This decomposition enhances model interpretability and streamlines the training process, as the optimization burden is reduced due to a smaller number of parameters relative to fully interconnected neural networks [6, 10 - 12, 15 - 21, 24 - 29]. Additionally, the modular architecture of KANs supports their integration into hybrid systems [19, 20, 24, 26, 28], including Neural ODEs, where KANs function as gradient evaluators to iteratively optimize solutions to ODEs.

KANs utilize univariate function composition, which results in high convergence efficiency [6, 10 - 12, 17 - 19, 20 - 28]. The univariate basis functions are designed to capture distinct characteristics of the input, facilitating expedited learning and mitigating overfitting [10, 17, 19 - 24, 26, 28]. This attribute is especially critical in addressing ODEs, where the solution landscape may present abrupt gradients or localized features. By integrating domain-specific insights into the selection of basis functions, such as Gaussian RBFs or B-splines, KANs demonstrate enhanced efficacy in approximating solutions to complex differential equations relative to alternative neural network frameworks, therefore, the KAN model serves as a powerful tool for tasks such as solving ODEs. Its theoretical foundation, alongside its efficient architectural design and adaptability to high-dimensional parameter spaces, positions it as a superior alternative to conventional neural network paradigms. By decomposing multivariate functions into their univariate components, KANs enhance computational efficiency and demonstrate strong approximation properties, thereby aligning optimally with the requirements of contemporary ODE-solving techniques.

1. **Methodology of Solving ODEs using KAN**

There are various methodologies to solve ODEs using neural networks, one of which is the trial solution approach [1-3, 5]. The trial solution methodology is a crucial step in the neural network-based approximation of differential equations. The general first-order differential equation is represented as:

|  |  |
| --- | --- |
|  | (6) |

with being the initial condition.

From there, the differential equation is now an IVP and a trial solution can be constructed by considering the values from the initial condition, as such:

|  |  |
| --- | --- |
|  | (7) |

here, is the output of the KAN model, as derived in equation 5.

The error used to train the KAN model can be defined as such:

|  |  |
| --- | --- |
|  | (8) |

where is defined as:

|  |  |
| --- | --- |
|  | (9) |

Finally, the new weight, , can be calculated using the error defined in equation 8:

|  |  |
| --- | --- |
|  | (10) |

where is the learning rate, and the weight is any of the trainable parameters defined in section 2, which are , , and .

The formulaic representations of the gradients for updating these parameters are given in equations 11-13:

|  |  |
| --- | --- |
|  | (11) |

|  |  |
| --- | --- |
|  | (12) |

|  |  |
| --- | --- |
|  | (13) |

1. **Numerical Examples**

In this section, some equations are evaluated by the proposed KAN model. The results are then compared to other similar approaches. To maintain comparability, the same KAN architecture was employed to evaluate all numerical example. The shape of the architecture is (1, 7, 4, 1), consisting of 1 input node (for variable ), 1 output node (for the predicted ), and 2 layers, one with 7 nodes and another with 4 nodes between the input and output, in that order. For the spline definition of the activation functions, cubic B-splines were uniformly sampled at 100 points in the range [-5, 5]. The basis function was kept as defined in equation 3. The model was trained using LBFGS optimization with a learning rate of 0.2. Additionally, the Mean Squared Error (MSE) and Mean Absolute Error (MAE) were employed for evaluating the model.

MSE measures the average squared difference between the predicted and actual values, providing a measure of the model’s accuracy with focus on larger errors. It is defined as:

|  |  |
| --- | --- |
|  | (14) |

where represents the actual value, the predicted value, and the number of observations.

On the other hand, MAE calculates the average absolute difference between predicted and actual values, offering a more robust evaluation that is less sensitive to outliers. It is defined as:

|  |  |
| --- | --- |
|  | (15) |

* 1. **Example 1**

Consider the following first-order differential equation:

|  |  |
| --- | --- |
|  | (16) |

With the initial condition .

| **t** | **Exact solution** | **Euler** | **Runge-Kutta** | **RBFNet (n=9) [3]** | **KAN** | **RBFNN (n=9) [2]** |
| --- | --- | --- | --- | --- | --- | --- |
| 0.00 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 0.05 | 0.9536 | 0.9500 | 0.9536 | 0.9536 | 0.9535 | 0.9536 |
| 0.10 | 0.9137 | 0.9072 | 0.9138 | 0.9137 | 0.9137 | 0.9137 |
| 0.15 | 0.8798 | 0.8707 | 0.8799 | 0.8798 | 0.8798 | 0.8798 |
| 0.20 | 0.8514 | 0.8401 | 0.8515 | 0.8514 | 0.8514 | 0.8514 |
| 0.25 | 0.8283 | 0.8150 | 0.8283 | 0.8283 | 0.8283 | 0.8283 |
| 0.30 | 0.8104 | 0.7953 | 0.8105 | 0.8104 | 0.8104 | 0.8104 |
| 0.35 | 0.7978 | 0.7810 | 0.7979 | 0.7978 | 0.7977 | 0.7978 |
| 0.40 | 0.7905 | 0.7721 | 0.7907 | 0.7905 | 0.7905 | 0.7905 |
| 0.45 | 0.7889 | 0.7689 | 0.7890 | 0.7889 | 0.7888 | 0.7889 |
| 0.50 | 0.7931 | 0.7717 | 0.7932 | 0.7930 | 0.7930 | 0.7931 |
| 0.55 | 0.8033 | 0.7805 | 0.8035 | 0.8033 | 0.8033 | 0.8033 |
| 0.60 | 0.8200 | 0.7958 | 0.8201 | 0.8199 | 0.8199 | 0.8200 |
| 0.65 | 0.8431 | 0.8178 | 0.8433 | 0.8431 | 0.8431 | 0.8431 |
| 0.70 | 0.8731 | 0.8467 | 0.8733 | 0.8731 | 0.8731 | 0.8731 |
| 0.75 | 0.9101 | 0.8826 | 0.9102 | 0.9100 | 0.9100 | 0.9101 |
| 0.80 | 0.9541 | 0.9258 | 0.9542 | 0.9540 | 0.9540 | 0.9541 |
| 0.85 | 1.0053 | 0.9763 | 1.0054 | 1.0052 | 1.0052 | 1.0053 |
| 0.90 | 1.0637 | 1.0342 | 1.0638 | 1.0637 | 1.0637 | 1.0637 |
| 0.95 | 1.1293 | 1.0995 | 1.1294 | 1.1293 | 1.1293 | 1.1293 |
| 1.00 | 1.2022 | 1.1721 | 1.2022 | 1.2021 | 1.2022 | 1.2022 |
| **MSE** | 0 | 4.60e-04 | 1.24e-08 | 6.80e-10 | 1.36e-12 | 7.56e-14 |

**Table 1.** Comparison of MSE Values from References with KAN MSE for Example 1

The results in Table 1 demonstrate the superior accuracy of the KAN model compared to other numerical methods and neural network architectures in approximating the solution to the example given in equation 16. Despite performing slightly below the RBFNN approach, the KAN architecture still achieved remarkably high performance, with an MSE of the order 10-12. This highlights KAN's ability to closely approximate the exact solution with minimal error across the entire domain.

****

**Figure 2.** Exact and KAN Solution Comparison for Example 1

* 1. **Example 2**

Consider another first-order linear differential equation:

|  |  |
| --- | --- |
|  | (17) |

With the initial condition

| **t** | **WNNPSOA [1]** | **WNNPSO [1]** | **WNNMBP [1]** | **WNNBOA [1]** | **WNNDEV [1]** | **WNNIBOA [1]** | **RBFNet [3]** | **KAN** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.15 | 3.96e-03 | 4.26e-03 | 6.06e-03 | 3.83e-01 | 2.06e-01 | 1.94e-07 | 3.35e-04 | 1.02e-04 |
| 0.30 | 8.60e-03 | 8.61e-03 | 1.37e-02 | 4.91e-01 | 2.78e-01 | 1.14e-07 | 3.97e-04 | 7.78e-05 |
| 0.45 | 4.78e-03 | 4.31e-03 | 8.48e-03 | 4.55e-01 | 9.93e-02 | 9.81e-08 | 1.01e-04 | 6.29e-05 |
| 0.60 | 7.14e-04 | 1.37e-03 | 7.51e-04 | 3.72e-01 | 3.56e-02 | 6.96e-08 | 4.16e-04 | 4.82e-05 |
| 0.75 | 2.00e-03 | 2.33e-03 | 4.37e-03 | 3.07e-01 | 3.90e-02 | 4.66e-08 | 1.10e-03 | 4.25e-05 |
| 0.90 | 4.93e-04 | 8.68e-04 | 1.09e-03 | 2.89e-01 | 2.33e-01 | 3.22e-08 | 1.39e-03 | 3.53e-05 |
| 1.05 | 2.90e-03 | 3.74e-03 | 4.05e-03 | 3.18e-01 | 3.49e-01 | 2.35e-08 | 9.82e-04 | 9.93e-06 |
| 1.20 | 2.65e-03 | 3.21e-03 | 5.86e-03 | 3.74e-01 | 2.80e-01 | 2.71e-08 | 5.99e-04 | 8.43e-06 |
| 1.35 | 4.75e-04 | 1.09e-04 | 3.18e-03 | 4.27e-01 | 8.31e-02 | 3.27e-08 | 8.12e-04 | 6.32e-05 |
| 1.50 | 1.22e-03 | 2.34e-03 | 1.17e-03 | 4.48e-01 | 8.79e-02 | 3.05e-08 | 1.12e-03 | 3.67e-05 |
| 1.65 | 1.06e-03 | 1.94e-03 | 3.50e-03 | 4.20e-01 | 1.25e-01 | 2.46e-08 | 1.09e-03 | 6.43e-06 |
| 1.80 | 2.19e-04 | 4.95e-04 | 2.37e-03 | 3.48e-01 | 4.67e-02 | 1.66e-08 | 6.07e-04 | 8.22e-06 |
| 1.95 | 1.01e-03 | 2.22e-03 | 7.16e-04 | 2.54e-01 | 3.59e-02 | 5.45e-09 | 1.06e-04 | 4.58e-06 |
| 2.10 | 6.39e-04 | 1.46e-03 | 2.95e-03 | 1.74e-01 | 2.95e-02 | 4.24e-10 | 2.78e-04 | 4.70e-06 |
| 2.25 | 1.47e-04 | 7.81e-04 | 2.55e-03 | 1.42e-01 | 5.14e-02 | 1.37e-09 | 8.27e-04 | 1.03e-05 |
| 2.40 | 3.94e-04 | 1.78e-03 | 2.58e-05 | 1.87e-01 | 1.04e-01 | 1.17e-08 | 1.23e-03 | 1.67e-05 |
| 2.55 | 1.18e-04 | 2.82e-04 | 2.30e-03 | 3.23e-01 | 5.27e-02 | 1.30e-08 | 1.48e-03 | 2.37e-05 |
| 2.70 | 8.06e-06 | 1.53e-03 | 2.05e-03 | 5.54e-01 | 4.40e-02 | 2.57e-09 | 1.78e-03 | 6.30e-06 |
| 2.85 | 3.69e-05 | 4.11e-04 | 9.74e-04 | 8.80e-01 | 5.01e-02 | 2.61e-08 | 2.49e-03 | 1.83e-06 |
| 3.00 | 1.24e-04 | 1.45e-03 | 2.23e-03 | 1.31e+00 | 2.43e-03 | 9.59e-08 | 2.86e-03 | 9.76e-05 |
| **MAE** | 1.50e-03 | 2.07e-03 | 3.26e-03 | 4.03e-01 | 1.06e-01 | 4.12e-08 | 9.52e-04 | 9.45e-07 |

**Table 2.** Comparison of MAE Values from References with KAN MAE for Example 2

| **t** | **Exact Solution** | **RBFNet (n=21) [3]** | **RBFNN (n=21) [2]** | **RBFNN (n=90) [2]** | **KAN** |
| --- | --- | --- | --- | --- | --- |
| 0.00 | 3.0000 | 3.0000 | 3.0000 | 3.0000 | 3.000 |
| 0.15 | 2.3438 | 2.2435 | 2.3438 | 2.3438 | 2.3437 |
| 0.30 | 1.8142 | 1.8138 | 1.8142 | 1.8142 | 1.8141 |
| 0.45 | 1.3511 | 1.3510 | 1.3511 | 1.3511 | 1.3511 |
| 0.60 | 0.9348 | 0.9344 | 0.9348 | 0.9348 | 0.9347 |
| 0.75 | 0.5763 | 0.5752 | 0.5763 | 0.5763 | 0.5762 |
| 0.90 | 0.3012 | 0.2998 | 0.3012 | 0.3012 | 0.3011 |
| 1.05 | 0.1318 | 0.1308 | 0.1318 | 0.1318 | 0.1317 |
| 1.20 | 0.0726 | 0.0720 | 0.0726 | 0.0726 | 0.0725 |
| 1.35 | 0.1038 | 0.1030 | 0.1038 | 0.1038 | 0.1038 |
| 1.50 | 0.1845 | 0.1834 | 0.1845 | 0.1845 | 0.1844 |
| 1.65 | 0.2643 | 0.2632 | 0.2643 | 0.2643 | 0.2642 |
| 1.80 | 0.2988 | 0.2982 | 0.2988 | 0.2988 | 0.2988 |
| 1.95 | 0.2638 | 0.2637 | 0.2638 | 0.2638 | 0.2638 |
| 2.10 | 0.1625 | 0.1622 | 0.1625 | 0.1625 | 0.1624 |
| 2.25 | 0.0235 | 0.02227 | 0.0235 | 0.0235 | 0.0235 |
| 2.40 | -0.1095 | -0.1107 | -0.1095 | -0.1095 | -0.1094 |
| 2.55 | -0.1937 | -0.1952 | -0.1937 | -0.1937 | -0.1936 |
| 2.70 | -0.2025 | -0.2043 | -0.2025 | -0.2025 | -0.2025 |
| 2.85 | -0.1348 | -0.1373 | -0.1348 | -0.1348 | -0.1348 |
| 3.00 | -0.0157 | -0.0186 | -0.0157 | -0.0157 | -0.0156 |
| **MSE** | 0 | 1.44e-06 | 5.85e-12 | 9.95e-13 | 1.49e-11 |

**Table 3.** Comparison of MSE Values from References with KAN MSE for Example 2

The comparison in Table 2 highlights the exceptional performance of the KAN model in approximating the solution of the example ODE given in equation 17. KAN achieves the smallest MAE among all tested methods, including WNN variants and RBFNs, only falling short of the WNN with an Improved Butterfly Optimization Algorithm (WNNIBOA).

While the KAN model does not have the best performance compared to other RBFNN models in Table 3, the approximation is still quite accurate, and the difference in the error level is highly negligible in the majority of use cases.



**Figure 3.** Exact and KAN Solution Comparison for Example 2

* 1. **Example 3**

Consider another first-order linear differential equation:

|  |  |
| --- | --- |
|  | (18) |

With the initial condition

| **t** | **WNNIBOA [1]** | **WNNPSO [1]** | **WNNMBP [1]** | **WNNBOA [1]** | **WNNDEV [1]** | **WNNPSOA [1]** | **CNNs [7]** | **PSNNs [7]** | **Heun [7]** | **KAN** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.20 | 3.72e-09 | 5.05e-05 | 4.73e-04 | 1.54e-01 | 1.01e-02 | 1.64e-04 | 6.01e-04 | 2.99e-04 | 2.00e-03 | 7.96e-05 |
| 0.40 | 2.29e-09 | 9.97e-05 | 7.12e-04 | 2.94e-01 | 3.50e-03 | 2.15e-04 | 3.88e-04 | 4.88e-04 | 4.19e-03 | 6.03e-05 |
| 0.60 | 6.12e-09 | 6.67e-06 | 4.75e-04 | 4.22e-01 | 7.78e-03 | 9.45e-05 | 2.34e-03 | 6.41e-04 | 6.84e-03 | 4.17e-05 |
| 0.80 | 9.51e-09 | 1.88e-04 | 4.12e-04 | 5.28e-01 | 1.36e-02 | 2.53e-04 | 1.53e-03 | 7.30e-04 | 9.63e-03 | 1.74e-05 |
| 1.00 | 9.65e-09 | 6.65e-05 | 7.53e-04 | 6.06e-01 | 8.46e-03 | 6.29e-05 | 1.74e-03 | 8.59e-04 | 1.29e-02 | 3.57e-06 |
| 1.20 | 1.42e-08 | 1.54e-04 | 1.18e-03 | 6.68e-01 | 8.34e-04 | 4.41e-04 | 4.44e-03 | 1.14e-03 | 1.64e-02 | 9.05e-06 |
| 1.40 | 1.23e-08 | 5.09e-05 | 1.28e-03 | 7.37e-01 | 5.92e-03 | 2.93e-04 | 3.50e-03 | 1.30e-03 | 2.04e-02 | 2.47e-05 |
| 1.60 | 3.07e-08 | 2.23e-04 | 1.17e-03 | 8.32e-01 | 1.06e-02 | 1.78e-04 | 5.48e-03 | 1.48e-03 | 2.47e-02 | 2.24e-05 |
| 1.80 | 1.52e-08 | 1.41e-04 | 1.57e-03 | 9.65e-01 | 1.20e-02 | 1.18e-06 | 4.28e-03 | 1.76e-04 | 2.94e-02 | 4.67e-05 |
| 2.00 | 3.51e-07 | 3.48e-05 | 2.47e-03 | 1.14e-00 | 1.51e-02 | 6.41e-04 | 1.90e-02 | 7.03e-03 | 3.42e-02 | 2.07e-04 |
| MAE | 3.13e-08 | 9.23e-05 | 9.54e-04 | 5.76e-01 | 7.99e-03 | 2.13e-04 | 3.93e-03 | 1.28e-03 | 1.46e-02 | 1.44e-06 |

**Table 4.** Comparison of MAE Values from References with KAN MAE for Example 3

As shown in Table 4, once again the KAN model outperforms traditional neural networks and numerical methods such as Heun's method and various WNN approaches, only overshadowed by the complex WNNIBOA [1, 7]. The MAE achieved by KAN affirms its capability to resolve ODEs. This performance can be attributed to the inherent design of the KAN architecture, which efficiently decomposes complex functions and adapts to localized features of the solution.

****

**Figure 4**. Exact and KAN Solution Comparison for Example 3

Figures show the close correspondence between the exact solutions of the differential equations and those approximated by the KAN model. In these plots, the curves representing the exact solutions and the KAN-generated solutions are almost indistinguishable across the entire domain, even for rapidly fluctuating functions like example 2. This overlay demonstrates that the KAN architecture effectively captures the dynamics of first-order ODEs, thereby validating its use for such applications.

The minor discrepancies observed in certain regions are well within the acceptable error margins and can be attributed to numerical approximation tolerances rather than any shortcomings in the KAN model itself. These results, supported by the minimal error values reported in the analysis of each example ODE, reinforce the robustness and accuracy of the KAN approach. Notably, the figures provide visual evidence that the KAN model is not only theoretically, but also practically suitable for precise function approximation in differential equation applications.

1. **Conclusion**

In this study, KAN was explored as an alternative approach for calculating solutions to first-order ODEs. The KAN model, leveraging the KAT, successfully decomposes complex multivariate functions into simpler univariate components. This quality leads to highly efficient and accurate function approximation. The results obtained from the numerical examples demonstrate that the KAN model is on par with or outperforms traditional numerical methods, as well as other neural network models such as RBFNNs and WNNs, in terms of accuracy, convergence speed, and computational efficiency. With significantly lower or comparable MSE and MAE across all tests, KAN’s potential as a robust and reliable tool for solving ODEs is demonstrated. The study highlights the advantages of KAN's ability to approximate intricate mathematical models while maintaining minimal computational requirements. As neural network-based approaches continue to advance, the KAN model is a promising method in the field of computational mathematics for resolving ODEs and other complex scientific problems. Future research may focus on optimization techniques and extending KAN's application to more complex higher-order differential equations.

**CRediT authorship contribution statement**

Morteza Farroknejad: Writing - review & editing, Writing - original draft, Formal analysis, Data curation, Conceptualization.

Ali Farrokhnejad: Conceptualization. Writing - review & editing, Resources, Methodology, Formal analysis.

Ahmet Rizaner: Supervision, Validation, Conceptualization, Writing - review & editing, Resources, Project administration, Formal analysis, Methodology.

**Declaration of Competing Interest**

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Data Availability**

No data was used for the research described in the article.

**Declaration of generative AI and AI-assisted technologies in the writing process.**

During the preparation of this work the authors used GPT-4o in order to improve the readability and language of the manuscript. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of this article.

**References**

1. Tan, L. S., Zainuddin, Z., Ong, P., & Abdullah, F. A. (2024). An effective wavelet neural network approach for solving first and second order ordinary differential equations. *Applied Soft Computing*, *154*, 111328, <https://doi.org/10.1016/j.asoc.2024.111328>
2. Liu, M., Peng, W., Hou, M., & Tian, Z. (2023). Radial basis function neural network with extreme learning machine algorithm for solving ordinary differential equations. *Soft Computing*, *27*(7), 3955-3964, <https://doi.org/10.1007/s00500-022-07529-3>
3. Rizaner, F. B., & Rizaner, A. (2018). Approximate solutions of initial value problems for ordinary differential equations using radial basis function networks. *Neural Processing Letters*, *48*, 1063-1071, <https://doi.org/10.1007/s11063-017-9761-9>
4. Dwivedi, V., & Srinivasan, B. (2020). Physics informed extreme learning machine (pielm)-a rapid method for the numerical solution of partial differential equations. *Neurocomputing*, *391*, 96-118, <https://doi.org/10.1016/j.neucom.2019.12.099>
5. Li, S., & Wang, X. (2021). Solving ordinary differential equations using an optimization technique based on training improved artificial neural networks. *Soft Computing*, *25*(5), 3713-3723, <https://doi.org/10.1007/s00500-020-05401-w>
6. Koenig, B. C., Kim, S., & Deng, S. (2024). KAN-ODEs: Kolmogorov-Arnold network ordinary differential equations for learning dynamical systems and hidden physics. *Computer Methods in Applied Mechanics and Engineering*, *432*, 117397, <https://doi.org/10.1016/j.cma.2024.117397>
7. Haweel, T. I., & Abdelhameed, T. N. (2015, February). Power series neural network solution for ordinary differential equations with initial conditions. In *2015 International Conference on Communications, Signal Processing, and their Applications (ICCSPA'15)*(pp. 1-5). IEEE, <https://doi.org/10.1109/ICCSPA.2015.7081317>
8. Sabir, Z., Wahab, H. A., Umar, M., Sakar, M. G., & Raja, M. A. Z. (2020). Novel design of Morlet wavelet neural network for solving second order Lane-Emden equation. *Mathematics and Computers in Simulation*, *172*, 1-14, <https://doi.org/10.1016/j.matcom.2020.01.005>
9. Soleymani, F., & Zhu, S. (2021). RBF-FD solution for a financial partial-integro differential equation utilizing the generalized multiquadric function. *Computers & Mathematics with Applications*, *82*, 161-178, <https://doi.org/10.1016/j.camwa.2020.11.010>
10. Shuai, H., & Li, F. (2025). Physics-informed kolmogorov-arnold networks for power system dynamics. *IEEE Open Access Journal of Power and Energy*, <https://doi.org/10.1109/OAJPE.2025.3529928>
11. Kich, V. A., Bottega, J. A., Steinmetz, R., Grando, R. B., Yorozu, A., & Ohya, A. (2024, October). Kolmogorov-Arnold Networks for Online Reinforcement Learning. In *2024 24th International Conference on Control, Automation and Systems (ICCAS)*(pp. 958-963). IEEE, DOI: [23919/ICCAS63016.2024.10773080](https://doi.org/10.23919/ICCAS63016.2024.10773080)
12. Yu, T., Qiu, J., Yang, J., & Oseledets, I. (2024). Sinc kolmogorov-arnold network and its applications on physics-informed neural networks. *arXiv preprint arXiv:2410.04096*.
13. Goswami, S., Kontolati, K., Shields, M. D., & Karniadakis, G. E. (2022). Deep transfer operator learning for partial differential equations under conditional shift. *Nature Machine Intelligence*, *4*(12), 1155-1164, <https://doi.org/10.1038/s42256-022-00569-2>
14. Chen, Y., Yu, H., Meng, X., Xie, X., Hou, M., & Chevallier, J. (2021). Numerical solving of the generalized Black-Scholes differential equation using Laguerre neural network. *Digital Signal Processing*, *112*, 103003, <https://doi.org/10.1016/j.dsp.2021.103003>
15. Selitskiy, S. (2022). Kolmogorov's Gate Non-linearity as a Step toward Much Smaller Artificial Neural Networks. In *ICEIS (1)*(pp. 492-499), DOI: 10.5220/0011060700003179
16. van Deventer, H., van Rensburg, P. J., & Bosman, A. (2022). KASAM: Spline Additive Models for Function Approximation. *arXiv preprint arXiv:2205.06376*, <https://doi.org/10.48550/arXiv.2205.06376>
17. Shukla, K., Toscano, J. D., Wang, Z., Zou, Z., & Karniadakis, G. E. (2024). A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks. *arXiv preprint arXiv:2406.02917*, <https://doi.org/10.48550/arXiv.2406.02917>
18. Jahin, M. A., Masud, M. A., Mridha, M. F., Aung, Z., & Dey, N. (2024). Kacq-dcnn: Uncertainty-aware interpretable kolmogorov-arnold classical-quantum dual-channel neural network for heart disease detection. *arXiv preprint arXiv:2410.07446*.
19. Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T.Y. & Tegmark, M. (2024). Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756,* <https://doi.org/10.48550/arXiv.2404.19756>
20. Hou, Y., & Zhang, D. (2024). A comprehensive survey on Kolmogorov Arnold networks (KAN). *arXiv preprint arXiv:2407.11075*, <https://doi.org/10.48550/arXiv.2407.11075>
21. Basina, D., Vishal, J. R., Choudhary, A., & Chakravarthi, B. (2024). KAT to KANs: A Review of Kolmogorov-Arnold Networks and the Neural Leap Forward. *arXiv preprint arXiv:2411.10622*, <https://doi.org/10.48550/arXiv.2411.10622>
22. Vaca-Rubio, C. J., Blanco, L., Pereira, R., & Caus, M. Kolmogorov-arnold networks (kans) for time series analysis. arXiv 2024. *arXiv preprint arXiv:2405.08790*, <https://doi.org/10.48550/arXiv.2405.08790>
23. Drokin, I. (2024). Kolmogorov-arnold convolutions: Design principles and empirical studies. \*arXiv preprint arXiv:2407.01092, [https://doi.org/10.48550/arXiv.2407.01092\*](https://doi.org/10.48550/arXiv.2407.01092*)
24. Danish, M. U., & Grolinger, K. (2025). Kolmogorov-Arnold recurrent network for short term load forecasting across diverse consumers. *Energy Reports*, *13*, 713-727, <https://doi.org/10.1016/j.egyr.2024.12.038>
25. Rigas, S., Papachristou, M., Papadopoulos, T., Anagnostopoulos, F., & Alexandridis, G. (2024). Adaptive training of grid-dependent physics-informed kolmogorov-arnold networks. *IEEE Access*, <https://doi.org/10.1109/ACCESS.2024.3504962>
26. Somvanshi, S., Javed, S. A., Islam, M. M., Pandit, D., & Das, S. (2024). A survey on kolmogorov-arnold network. *arXiv preprint arXiv:2411.06078*, <https://doi.org/10.48550/arXiv.2411.06078>
27. Guo, C., Sun, L., Li, S., Yuan, Z., & Wang, C. (2024). Physics-informed Kolmogorov-Arnold Network with Chebyshev Polynomials for Fluid Mechanics. *arXiv preprint arXiv:2411.04516*, <https://doi.org/10.48550/arXiv.2411.04516>
28. Hu, J., Tian, H., & Guo, Q. (2025). Discovering Dynamics with Kolmogorov Arnold Networks: Linear Multistep Method-Based Algorithms and Error Estimation. *arXiv preprint arXiv:2501.15066*, <https://doi.org/10.48550/arXiv.2501.15066>
29. Kupriyanov, G., Isaev, I., Laptinskiy, K., Dolenko, T., & Dolenko, S. (2024). Solution of an Inverse Problem of Optical Spectroscopy Using Kolmogorov-Arnold Networks. *Optical Memory and Neural Networks*, *33*(Suppl 3), S475-S482, <https://doi.org/10.3103/S1060992X24700747>