**DISPLACED INFO:**

The general first-order differential equation is represented as:

The general first-order differential equation is represented as:

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

Finally, the new weights can be calculated using the error defined previously:

**Abstract**

**WRITE THIS WHEN THE WORK IS DONE**

1. **Introduction**

The majority of the engineering and science systems can be modeled and analytically studied using ordinary differential equations that naturally arise in many applications in science and engineering such as fluid mechanics, population dynamics, chemical kinetics and structures [1-5, 13]. However, since Ordinary differential equations (ODEs) can be complex, they present great difficulty in their solution as these equations often do not have analytical solutions, thus requiring the use of numerical or approximation methods. Over the years, the finite difference method, shooting method and the runge-kutta method have been the methods of choice to try to overcome these challenges. Their Still, however, does exist a limitation, chisels and holes cannot provide closed solutions and have scrupulous computing power requirements that have resulted in the investigation of alternative methods for solving ODEs [1-3, 5, 12, 13].

Artificial Neural Networks (ANNs) however introduced new constructs for the global solution to such ODEs by recontextualizing the problem as an optimization framework. Initial studies indicate that multilayer perceptrons can be efficient in approximation solutions to Initial Value Problems (IVPs) Models and also Boundary Value Problems (BVPs) Models.

These methods based on neural networks are much more efficient than the classical numerical techniques. In particular, ANNs are capable of formulating analytic solutions which eliminate the necessity for performing interpolation over discretized computational intervals, hence more flexibility in solving IVPs and BVPs [1, 3, 6, 7, 11, 13 -17]. On the other hand, the first generation of models based on ANNs had several challenges 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, 12] and Wavelet Neural Networks (WNNs) [1] have emerged. These approaches have been recorded to have shorter convergence times and a higher accuracy compared to traditional techniques when applied to complex expressions of differential equations. Furthermore, WNNs have attracted considerable interest because 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 said to be achieved by neural networks[1]. Furthermore, the implementation of sophisticated training methodologies, such as Extreme Learning Machines (ELM) and metaheuristic optimization techniques, including Particle Swarm Optimization (PSO), has substantially enhanced both the efficiency and accuracy of these neural network models [1, 2, 4, 5, 14].

In response to this assertion, the Kolmogorov-Arnold Network (KAN) architecture evolves a novel architecture which is robust and function approximation, which shows potential for solving ODEs. ODEs are appreciably accounted for in this architecture. 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 [6-8, 13, 15-18]. This inbuilt KAN Thus, the KAN proves to be especially useful in terms of complex mathematical models that are developed using ODEs [6, 7, 13, 16-18]. This paper attempts to use KAN’s systematic functioning on KAN as a function decomposition architecture on neural networks to overcome the shortcomings that are posed by the current neural network architectures on higher order differential equations.

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, 18]. 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, 7 ,13 ,15 ,17 ,18].  
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 [ 7, 8, 13, 17, 18]. 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 [18], 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]. Moreover, RBFNNs trained via extreme learning methodologies demonstrate the high rates of convergence and high accuracy regarding fractional differential equations [2, 3, 12]. 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 [15 - 18], 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 both first- and second-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 distinctive architectural framework and theoretical foundations establish it as a formidable alternative to prevailing ANN methodologies for function approximation. This research aims to enhance the current capabilities of neural network-based approaches in addressing first- and second-order ODEs by leveraging KAN, thereby facilitating advancements in computational mathematics and related fields.

**Write the organization of the paper here when the content structure is clear.**

1. **KAN Model (General info of the architecture) We can use the initial paper here, or use the existing references for citing**

The KAN model is optimally configured for function approximation tasks [15 - 18], including the resolution of ODEs, owing to its basis in the KAT. This theorem asserts that any continuous multivariate function can be expressed as a finite summation of univariate functions subjected to linear operations, thereby facilitating the ability of a KAN to approximate intricate functions with reduced network depth [6, 7, 13, 15 - 18]. By utilizing this property, KANs inherently diminish the computational complexity associated with multivariate functions while preserving accuracy, [6 - 8, 13, 17, 18] 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 [15, 17, 18], 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 - 8, 13 ,15 - 18]. The hidden layers of the network often employ Gaussian radial basis functions (RBFs) as activation functions, which have been chosen for their smoothness properties and ability to localize approximations in space [13, 17, 18]. These RBFs allow for a concentration of response from each hidden layer neuron to several areas in the input space, which is necessary due to the need for the ODEs to be well resolved considering that the overall mobile response is area driven. Unlike the WNNs, in which wavelet transforms are used in order to ensure low topology and thus enable the quickly training of the network, KANs offer a different approach, which is the theoretical assurances provided by the KAT [17, 18]. Notably, both WNNs and RBF networks be used successfully for different types of tasks [1, 2, 3, 12], however, the hierarchical univariate decomposition structure of KAN is more suitable for the task of ODE solution approximation [13]. This allows KANs to compute accurate gradient assessments which is of great value when used together with the differentiable ODE solvers. These solvers use the dynamical structures of the KANs to imitate the physical systems and retrieve hidden physical effects with the best possible thin costs of calculations.

As they are capable of handling a high-dimensional input space, KANs have certain advantages. In the implementation of KAT, application of superposition principle is used, which decreases the burden of high dimensionality by breaking complex functions into simpler constituent parts. Such decomposition improves the interpretability of the model and eases model training because there are fewer parameters to be optimized as compared to fully connected neural networks [6 - 8, 13 ,15 – 18]. Moreover, the modular structure of KANs allows their use in different types of systems [18] like Neural ODEs, where KANs are employed as gradient estimators and employed within a loop to improve the solution of the ODE. Through the composition of univariate functions, KANs are able to achieve excellent convergence rates [6, 7, 8, 13, 17, 18]. Since univariate basis functions are used to optimize specific features of the input’s data, the system is learned much faster and avoids overfitting [17, 18]. This characteristic is extremely important particularly in solving ODEs where the solution’s spatial domain has steep gradients or localized structures. 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, coupled with 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 exhibit strong approximation properties, thereby aligning optimally with the requirements of contemporary ODE-solving techniques.

1. **KAN Model for differential equations (Later we can explain more in case we choose any specific subtype of KAN – FROM REFERENCES: 7,8, AND 9)**

We can skip this part and immediately proceed with the numerical examples.

1. **Numerical Examples**

In this section, some example equations are evaluated by the KAN model. The results are then compared to other similar approaches.

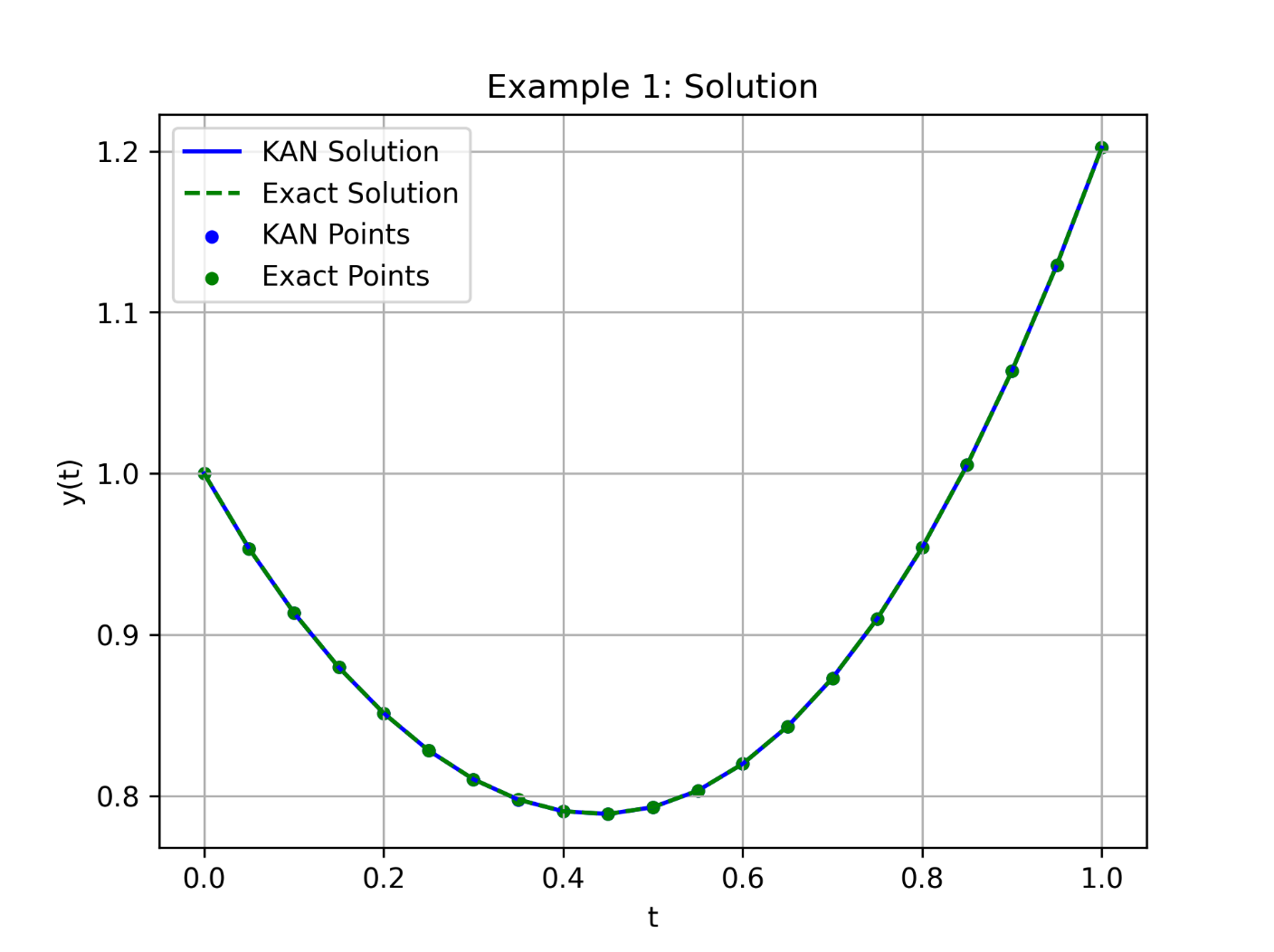
* + 1. **Example 1**

Consider the following first-order differential equation:

With the initial condition:

Compared results to other references:

| **x** | **Exact solution** | **Euler** | **RungeKutta** | **RBFNet in Rizaner et al. 2018 (n=9)** | **RBFNN solution (n=9)** |  | **KAN** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.00 | 1.0000 | 1.000 | 1.0000 | 1.0000 | 1.0000 |  | 1.0000 |
| 0.05 | 0.9536 | 0.9500 | 0.9536 | 0.9536 | 0.9536 |  | 0.9535 |
| 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 | 0.7978 |  | 0.7977 |
| 0.40 | 0.7905 | 0.7721 | 0.7907 | 0.7905 | 0.7905 |  | 0.7905 |
| 0.45 | 0.7889 | 0.7689 | 0.7890 | 0.7889 | 0.7889 |  | 0.7888 |
| 0.50 | 0.7931 | 0.7717 | 0.7932 | 0.7930 | 0.7931 |  | 0.7930 |
| 0.55 | 0.8033 | 0.7805 | 0.8035 | 0.8033 | 0.8033 |  | 0.8033 |
| 0.60 | 0.8200 | 0.7958 | 0.8201 | 0.8199 | 0.8200 |  | 0.8199 |
| 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 | 0.9101 |  | 0.9100 |
| 0.80 | 0.9541 | 0.9258 | 0.9542 | 0.9540 | 0.9541 |  | 0.9540 |
| 0.85 | 1.0053 | 0.9763 | 1.0054 | 1.0052 | 1.0053 |  | 1.0052 |
| 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 |  | 1.2021 |
| **Error** | 0 | 4.60e-10 | 1.24e-08 | 6.80e-10 | 7.56e-14 |  | 4.66e-11 |

**** Average MSE 4.6673 x 10-11, Average MAE 1.7985 x 10-6

* + 1. **Example 2**

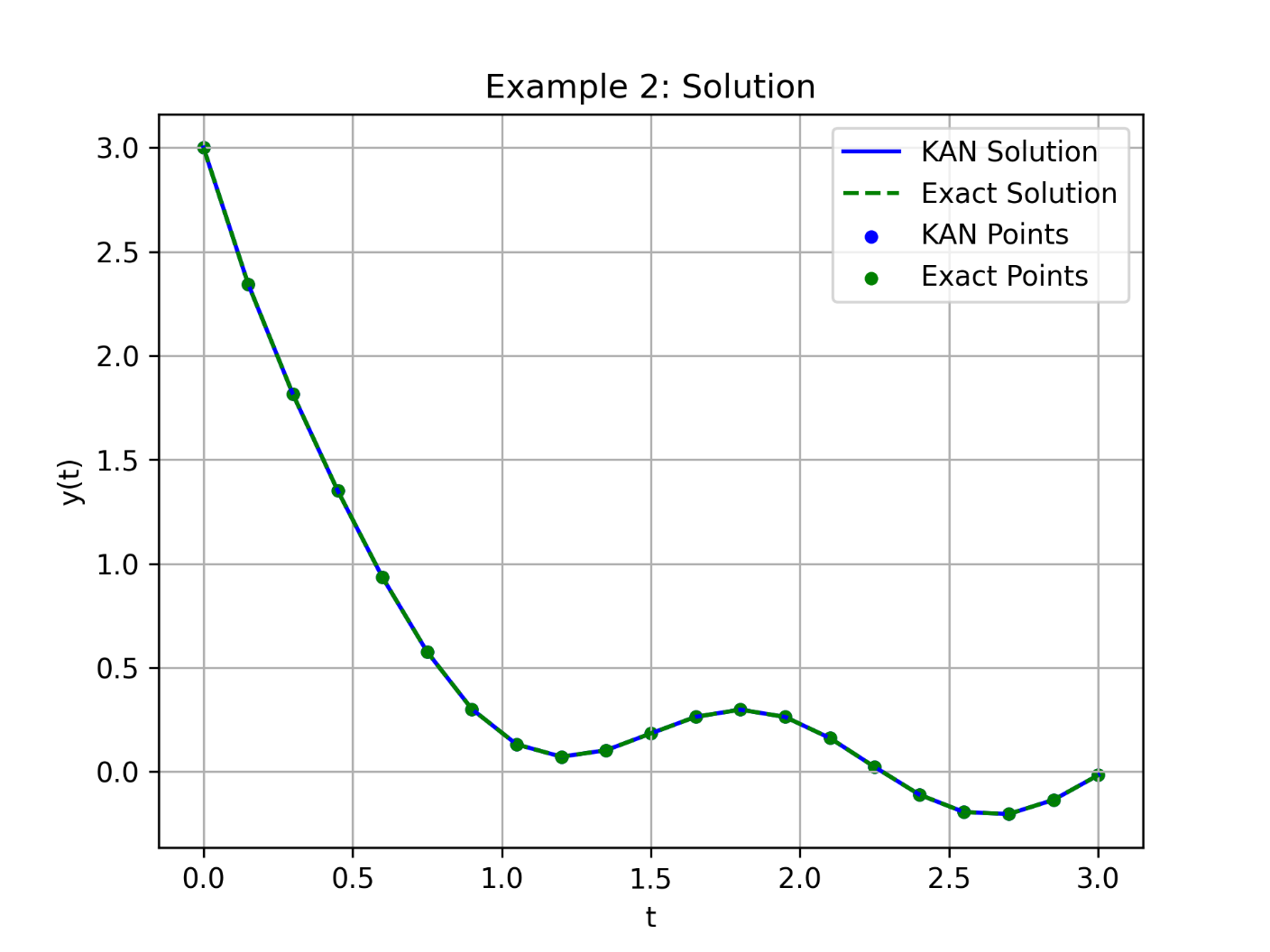
Consider another first-order linear differential equation:

With the initial condition:

Compared results to other references:

| **x** | **WNNIBOA** | **WNNBOA** | **WNNPSO** | **WNNPSOA** | **WNNMBP** | **WNNDEV** | **KAN** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.0000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.1500 | 1.94e-07 | 3.83e-01 | 4.26e-03 | 3.96e-03 | 6.06e-03 | 2.06e-01 | 1.02e-04 |
| 0.3000 | 1.14e-07 | 4.91e-01 | 8.61e-03 | 8.60e-03 | 1.37e-02 | 2.78e-01 | 7.78e-05 |
| 0.4500 | 9.81e-08 | 4.55e-01 | 4.31e-03 | 4.78e-03 | 8.48e-03 | 9.93e-02 | 6.29e-05 |
| 0.6000 | 6.96e-08 | 3.72e-01 | 1.37e-03 | 7.14e-04 | 7.51e-04 | 3.56e-02 | 4.82e-05 |
| 0.7500 | 4.66e-08 | 3.07e-01 | 2.33e-03 | 2.00e-03 | 4.37e-03 | 3.90e-02 | 4.25e-05 |
| 0.9000 | 3.22e-08 | 2.89e-01 | 8.68e-04 | 4.93e-04 | 1.09e-03 | 2.33e-01 | 3.53e-05 |
| 1.0500 | 2.35e-08 | 3.18e-01 | 3.74e-03 | 2.90e-03 | 4.05e-03 | 3.49e-01 | 9.93e-06 |
| 1.2000 | 2.71e-08 | 3.74e-01 | 3.21e-03 | 2.65e-03 | 5.86e-03 | 2.80e-01 | 8.43e-06 |
| 1.3500 | 3.27e-08 | 4.27e-01 | 1.09e-04 | 4.75e-04 | 3.18e-03 | 8.31e-02 | 6.32e-05 |
| 1.5000 | 3.05e-08 | 4.48e-01 | 2.34e-03 | 1.22e-03 | 1.17e-03 | 8.79e-02 | 3.67e-05 |
| 1.6500 | 2.46e-08 | 4.20e-01 | 1.94e-03 | 1.06e-03 | 3.50e-03 | 1.25e-01 | 6.43e-06 |
| 1.8000 | 1.66e-08 | 3.48e-01 | 4.95e-04 | 2.19e-04 | 2.37e-03 | 4.67e-02 | 8.22e-06 |
| 1.9500 | 5.45e-09 | 2.54e-01 | 2.22e-03 | 1.01e-03 | 7.16e-04 | 3.59e-02 | 4.58e-06 |
| 2.1000 | 4.24e-10 | 1.74e-01 | 1.46e-03 | 6.39e-04 | 2.95e-03 | 2.95e-02 | 4.70e-06 |
| 2.2500 | 1.37e-09 | 1.42e-01 | 7.81e-04 | 1.47e-04 | 2.55e-03 | 5.14e-02 | 1.03e-05 |
| 2.4000 | 1.17e-08 | 1.87e-01 | 1.78e-03 | 3.94e-04 | 2.58e-05 | 1.04e-01 | 1.67e-05 |
| 2.5500 | 1.30e-08 | 3.23e-01 | 2.82e-04 | 1.18e-04 | 2.30e-03 | 5.27e-02 | 2.37e-05 |
| 2.7000 | 2.57e-09 | 5.54e-01 | 1.53e-03 | 8.06e-06 | 2.05e-03 | 4.40e-02 | 6.30e-06 |
| 2.8500 | 2.61e-08 | 8.80e-01 | 4.11e-04 | 3.69e-05 | 9.74e-04 | 5.01e-02 | 1.83e-06 |
| 3.0000 | 9.59e-08 | 1.31e+00 | 1.45e-03 | 1.24e-04 | 2.23e-03 | 2.43e-03 | 9.76e-05 |

| **x** | **Exact Solution** | **RBFNet Rizaner (n=21)** | **RBFNN (n=21)** | **RBFNN (n=90)** | **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 |
| **Error** | 0 | 1.44e-06 | 5.85e-12 | 9.95e-13 | 2.10e-10 |

 Average MSE 2.1085 x 10-10, Average MAE 3.3426 x 10-6

* + 1. **Example 3**

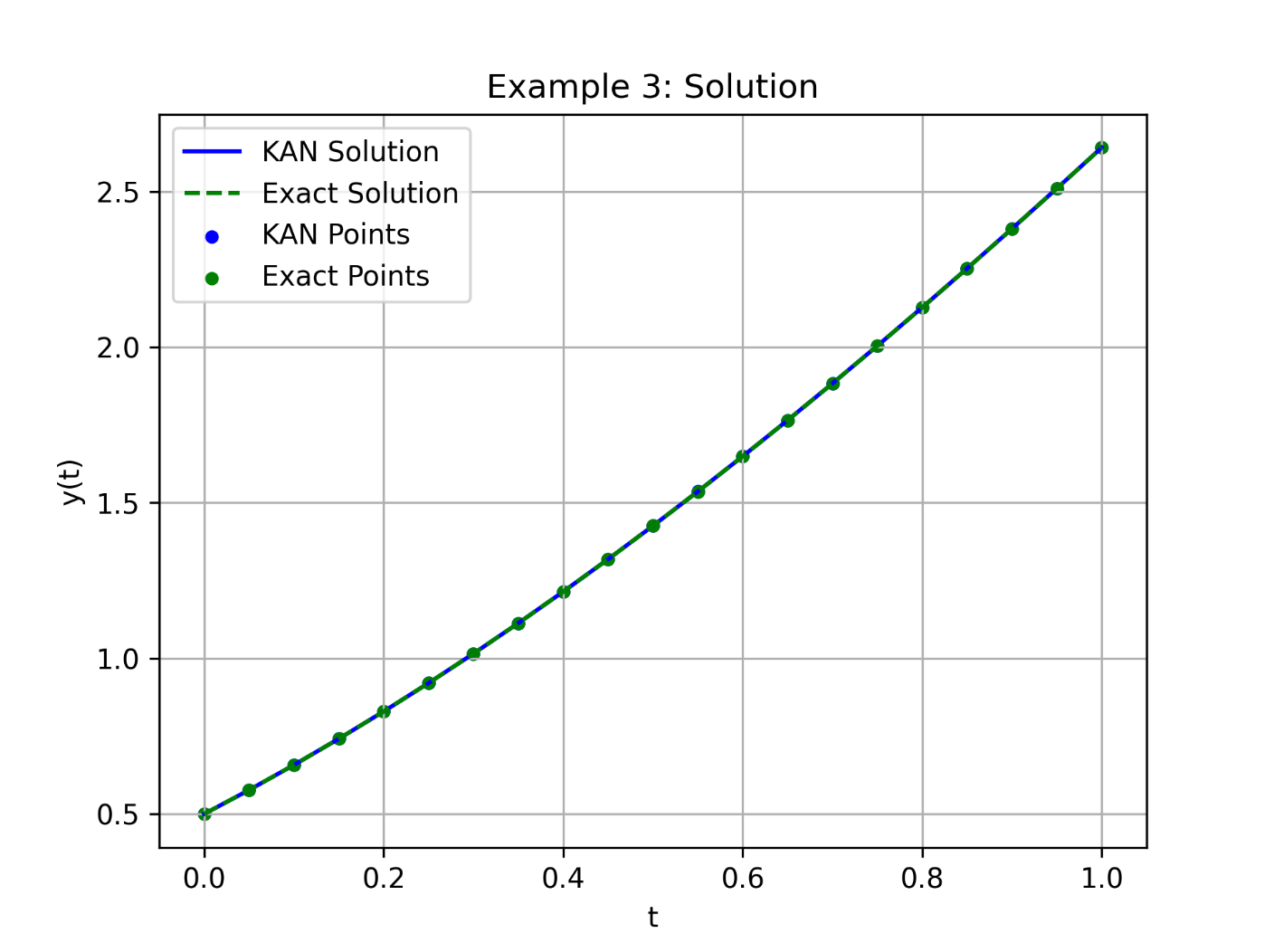
Consider another first-order linear differential equation:

With the initial condition:

Compared results to other references:

| **x** | **WNNIBOA** | **WNNBOA** | **WNNPSO** | **WNNPSOA** | **WNNMBP** | **WNNDEV** | **KAN** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.0000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.2000 | 3.72e-09 | 1.54e-01 | 5.05e-05 | 1.64e-04 | 4.73e-04 | 1.01e-02 | 7.96e-05 |
| 0.4000 | 2.29e-09 | 2.94e-01 | 9.97e-05 | 2.15e-04 | 7.12e-04 | 3.50e-03 | 6.03e-05 |
| 0.6000 | 6.12e-09 | 4.22e-01 | 6.67e-06 | 9.45e-05 | 4.75e-04 | 7.78e-03 | 4.17e-05 |
| 0.8000 | 9.51e-09 | 5.28e-01 | 1.88e-04 | 2.53e-04 | 4.12e-04 | 1.36e-02 | 1.74e-05 |
| 1.0000 | 9.65e-09 | 6.06e-01 | 6.65e-05 | 6.29e-05 | 7.53e-04 | 8.46e-03 | 3.57e-06 |
| 1.2000 | 1.42e-08 | 6.68e-01 | 1.54e-04 | 4.41e-04 | 1.18e-03 | 8.34e-04 | 9.05e-06 |
| 1.4000 | 1.23e-08 | 7.37e-01 | 5.09e-05 | 2.93e-04 | 1.28e-03 | 5.92e-03 | 2.47e-05 |
| 1.6000 | 3.07e-08 | 8.32e-01 | 2.23e-04 | 1.78e-04 | 1.17e-03 | 1.06e-02 | 2.24e-05 |
| 1.8000 | 1.52e-08 | 9.65e-01 | 1.41e-04 | 1.18e-06 | 1.57e-03 | 1.20e-02 | 4.67e-05 |
| 2.0000 | 3.51e-07 | 1.14e+00 | 3.48e-05 | 6.41e-04 | 2.47e-03 | 1.51e-02 | 2.07e-04 |

Average MSE 4.8837 x 10-10, Average MAE 4.8980 x 10-6

****

1. **Conclusion**

**WRITE THIS AND ABSTRACT IN THE END**

**Acknowledgements (if there are any)**

**References (APA for now unless we decide to change it.)**

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.
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.
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.
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.
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.
6. 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.
7. 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*.
8. 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*.
9. Liu, F., Viano, L., & Cevher, V. (2022). Understanding deep neural function approximation in reinforcement learning via $\epsilon $-greedy exploration. *Advances in Neural Information Processing Systems*, *35*, 5093-5108.
10. Heinlein, A., Klawonn, A., Lanser, M., & Weber, J. (2021). Combining machine learning and domain decomposition methods for the solution of partial differential equations—A review. *GAMM‐Mitteilungen*, *44*(1), e202100001.
11. 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.
12. 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.
13. 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.
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.
15. Selitskiy, S. (2022). Kolmogorov's Gate Non-linearity as a Step toward Much Smaller Artificial Neural Networks. In *ICEIS (1)* (pp. 492-499).
16. van Deventer, H., van Rensburg, P. J., & Bosman, A. (2022). KASAM: Spline Additive Models for Function Approximation. *arXiv preprint 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*.
18. 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*.