Kolmogorov-Arnold Networks: A rebirth of artificial intelligence towards solving differential equations?

Gregorio Pérez Bernal[a,1] **and Nicolás Guarín Zapata**[a,2]

This manuscript was compiled on November 21, 2024

Kolmogorov-Arnold Networks (KANs) represent a groundbreaking framework in numerical analysis and machine learning, offering a novel approach to approximate functions. Unlike traditional neural networks, KANs leverage the Kolmogorov-Arnold representation theorem to enhance accuracy and interpretability by directly learning complex functions. As this emerging architecture holds significant promise for applications where traditional methods struggle, it is essential to study its potential. In this study, the focus is in the numerical approximation of differential equations, such as the Helmholtz equation. This research aims to explore the effectiveness of physics-informed KANs (PIKANs) and variational physics-informed KANs (VPIKANS) in solving partial differential equations, laying the foundation for their broader applications in computational physics and engineering.

Kolmogorov Arnold Networks | Physically informed networks | Differential equations | Numerical methods | Approximation theory

Significance Statement

Kolmogorov-Arnold Networks (KANs) represent a groundbreaking framework in numerical analysis and machine learning, offering a novel approach to approximate functions. Unlike traditional neural networks, KANs leverage the Kolmogorov-Arnold representation theorem to enhance accuracy and interpretability by directly learning complex functions. As this emerging architecture holds significant promise for applications where traditional methods struggle, it is essential to study its potential. In this study, the focus is in the numerical approximation of differential equations, such as the Helmholtz equation. This research aims to explore the effectiveness of physics-informed KANs (PIKANs) and variational physics-informed KANs (VPIKANS) in solving partial differential equations, laying the foundation for their broader applications in computational physics and engineering.

Studying differential equations is crucial in physics as it allows us to model and predict the behavior of dynamic systems in various fields where changes in a function can be modeled more simply than the function itself (1). The Helmholtz equation is a differential equation studied in different areas of physics, such as seismology, electromagnetism, and even quantum mechanics (2). Given that the vast majority of differential equations cannot be solved analytically, there is a need to study ways to solve these problems numerically (1).

In addition to classical numerical methods, a new approach was proposed in 2019 to solve differential equations: physics-informed neural networks (PINNs) (3). This framework allows approximating the solution of a partial differential equation using a multilayer perceptron (MLP) with a cost function to minimize based on physical laws, such as differential equations and their variational forms.

In 2024, Kolmogorov-Arnold Networks (KANs) were introduced as an alternative to MLPs, offering greater interpretability and, for certain problems, higher accuracy in learning functions (4). These networks are effective thanks to the Kolmogorov-Arnold representation theorem, which establishes that any continuous multivariable function can be represented as a sum of continuous univariate functions. Hence, it is natural to ask: can physics-informed Kolmogorov-Arnold Networks (PIKANs) and their variational form (VPIKANS) improve the numerical approximation of the Helmholtz equation in one dimension?

1. Mathematical formulation of the problem

To properly comprehend the problem to solve, a short framework on the different mathematical tools that are used for solving differential equations using KAN.

A. Helmholtz Equation. Here, we are interested in the solution of the one-dimensional Helmholtz equation. This equation arises in various physical contexts, including the study of wave propagation, vibrations, and acoustics. In the context of boundary value problems, the Helmholtz equation models the steady-state behavior of physical systems, making it particularly useful for analyzing systems that exhibit periodic or wave-like behavior. Solving the one-dimensional Helmholtz equation accurately is crucial in applications like electromagnetic wave analysis, optical fibers, and quantum mechanics.

This differential equation is of the form:

$$\frac{d^2u}{dx^2} + k^2u = f(x)\,, \qquad [1]$$

Author affiliations: [a]School of Applied Science and Engineering, Universidad EAFIT, Medellín, Colombia

[1] nguarinz@eafit.edu.co
[2] gperezb1@eafit.edu.co

where $u(x)$ is the unknown function, k is a constant known as the *wave number*, and $f(x)$ is a source term. Subject to $u(0) = u(1) = 0$.

B. Kolmogorov Arnold Networks. Kolmogorov-Arnold Networks (KANs) are neural networks inspired by the Kolmogorov-Arnold representation theorem, which asserts that any multivariate continuous function can be decomposed into a finite sum of univariate functions. Unlike traditional Multi-Layer Perceptrons (MLPs), which employ fixed activation functions at the nodes, KANs introduce learnable activation functions along the edges. This fundamental shift allows each edge to represent a non-linear transformation using a parameterized spline function, effectively eliminating the need for traditional linear weight matrices (4). In theory, any spline function will work in the construction of the activation functions, however, as purposed by (4) and (5), the most widely used spline interpolation method is **B-splines** with **k interpolation points**.
KANs leverage this architecture to enhance accuracy and interpretability, especially in applications involving Partial Differential Equations (PDEs) and data fitting (5). The network's ability to capture intricate functional relationships makes it particularly effective for low-dimensional problems where traditional networks often face overfitting or inefficiency (4).

B.1. Kolmogorov-Arnold Representation Theory. The theoretical foundation of KANs is rooted in the Kolmogorov-Arnold representation theorem (6). This theorem states that any multivariate continuous function $f(x_1, x_2, \ldots, x_n)$ on a bounded domain can be expressed as a sum of continuous univariate functions:

$$f(x_1, x_2, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right), \qquad [2]$$

where $\varphi_{q,p}$ and Φ_q are continuous univariate functions. In this context, KANs are designed to explicitly model this representation by parametrizing the univariate functions $\varphi_{q,p}$ and Φ_q as learnable B-splines, allowing for efficient approximation of complex multivariate functions.
Unlike MLPs, which rely on fixed activation functions like ReLU, sigmoid, or inverse tangent, KANs place these univariate splines on the edges of the network, allowing each edge to learn non-linear transformations directly. The output of a general KAN with L layers is then given by the following composition of activation functions:

$$\mathrm{KAN}(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_0)(x), \qquad [3]$$

By utilizing splines for the edge activations, KANs achieve superior scaling behavior and approximation accuracy, particularly in data fitting and PDE-solving tasks. This spline-based approach allows KANs to surpass the curse of dimensionality often encountered with traditional neural networks, making them highly efficient for a wide range of applications (4).

C. Artificial Neural Networks. The process of training a neural network involves optimizing a set of parameters to minimize a predefined loss function, typically the Mean Squared Error (MSE). Given a dataset with inputs x_i and corresponding target values y_i, the MSE loss function is defined as:

$$\mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2, \qquad [4]$$

where $\hat{y}_i$ represents the network's predicted output, and N is the number of data points. The optimization process seeks to adjust the network's parameters, typically using gradient-based methods, to minimize the MSE.

D. A note on function spaces, measuring error and norms on functions. A function space refers to a vector space that contains a set of functions that follow certain properties. For instance, the space L^2 is the space such that:

$$L^2 = \left\{ f \,\middle|\, f \text{ is continuous}, \int_\Omega |f(x)|^2 \, dx < \infty \right\} \qquad [5]$$

Now that we have spaces, let us think about the norm of a function, in an analogous way from the definition of a vector's norm in L^2, as defined by equation 6.

$$\|f\|_{L^2} := \left(\int_\Omega |f(x)|^2 \, dx \right)^{\frac{1}{2}} \qquad [6]$$

Naturally, given a function and approximation of it, the approximation error in L^2 can be defined as shown in equation 7.

$$\varepsilon = \|f - f_{\text{approx}}\|_{L^2} = \left(\int_\Omega |f - f_{\text{approx}}|^2 \, dx \right)^{\frac{1}{2}} \qquad [7]$$

The weak forms of a differential equation do not exist in the same spaces as their classical form. This is why the definition of the Sobolev space (equation 8) is presented.

$$H_0^1 = \left\{ f \in L^2 \,\middle|\, \nabla f \in L^2, \partial f = 0 \right\} \qquad [8]$$

As H_0^1 includes the gradient of a function, to calculate a norm in H_0^1 is defined as in equation 9.

$$\|f\|_{H_0^1} = \left(\int_\Omega |f|^2 \, dx + \int_\Omega |\nabla f|^2 \, dx \right)^{\frac{1}{2}}. \qquad [9]$$

The error on that same norm is deduced naturally, as it is done in equation 7.

E. Physically informed Networks (PIKANS). Given the Helmholtz equation, the framework established to train the KAN that solves it is by minimizing the residual of the approximation $\hat{u}(x)$ at each step. This is defined by:

$$\mathcal{R}(x) = \begin{cases} \nabla^2 \hat{u}(x) + k^2 \hat{u}(x) - f(x), & \text{for } x \in \Omega, \\ \hat{u}(x), & \text{for } x \in \partial\Omega, \end{cases} \qquad [10]$$

Given equation 10 the following loss functions are defined, analogous to the MSE.

$$\mathcal{L}_{\text{PDE}}(\theta) = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} \|\mathcal{R}_{\mathcal{F}}(x_i)\|^2, \qquad [11]$$

$$\mathcal{L}_{\text{DB}}(\theta) = \frac{1}{N_{\text{DB}}} \sum_{i=1}^{N_{\text{DB}}} \|\mathcal{R}_{\mathcal{DB}}(x_i)\|^2, \qquad [12]$$

where N_{PDE} refers to the number of discretization points in the domain and N_{DB} to those in the boundary. By adding equations 11 and 12, the following loss function is defined

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PDE}}(\theta) + \mathcal{L}_{\text{DB}}(\theta)\,. \qquad [13]$$

The algorithm's purpose is to minimize equation 13, by attempting to find a set of parameters $\theta = \text{argmin}(\mathcal{L}(\theta))$.

F. variational Physically informed Networks (VPIKANS). To solve the Helmholtz equation using VPIKANS, one must first define the variational form of said equation, which is given by equation 14.

$$-\int_\Omega \nabla u \cdot \nabla v\, dx + k^2 \int_\Omega u \cdot v\, dx = \int_\Omega v\, dx \qquad [14]$$

In this formulation $\hat{u} \in H_0^1$. $v \in H_0^1$ refers to the classically known test functions. The test functions used are constructed using a combination of the Finite Element Method (FEM) and spline interpolation techniques. Initially, a one-dimensional finite element mesh is generated over the domain. The stiffness and mass matrices are assembled using local element contributions, which were then used to compute the eigenvalues and eigenvectors of the system.

To generate the test functions, the computed eigenvectors corresponding to the smallest eigenvalues are interpolated using B-splines. Specifically, the eigenvectors were used as the basis for constructing smooth test functions, with the spline interpolation ensuring smoothness and continuity across the domain.

The process of training the VPIKAN is analogous to that of training the PIKAN, however, it minimizes the residual of the variational form, as shown in equations 15 and 16.

$$\mathcal{L}_{PDE} = \left[\int_\Omega \left(\nabla \hat{u}\ \nabla v - k^2 \hat{u} \cdot v - f \cdot v\right)\, dx\right]^2 \qquad [15]$$

$$\mathcal{L}_{DB} = \lambda \hat{u}(x) \qquad [16]$$

In equation 16 $\lambda \in \mathbb{R}$ is a constant used to add value to the loss in the boundary condition. As done in VPIKANS, the loss to minimize is given by the addition of equations 15 and 16.

2. Literature Review

Since the introduction of PINNs, various studies have been conducted on solving differential equations. Aguilar (7) carried out experiments approximating a solution to the one-dimensional Poisson equation using PINNS and VPINNS, where the residual of the Poisson equation and its variational form were used as the cost function.

In the paper introducing KANs, it is shown that, for solving partial differential equations, a two-layer KAN with 10 neurons per layer is 100 times more accurate than an MLP with four layers of width 100 (error of 10^{-7} versus 10^{-5} in terms of mean squared error) and 100 times more efficient in terms of parameters (8). However, it is noted that KANs are significantly slower than MLPs during training due to their non-linear nature.

In (4), PIKANs are formally introduced, demonstrating their ability to solve partial differential equations. An important contribution of this publication is that it shows PIKANs to be more accurate than MLPs for learning functions with significant nonlinearities. By applying different variations of KANs to solve the 2D Helmholtz equation and comparing them with an MLP, improvements were observed in the mean squared error of the approximation relative to a known solution in KAN implementations and their variations (9). In this same study, PIKANs are described as suitable for solving problems involving non-linearities, such as fluid dynamics, and it is shown that KANs outperform traditional MLPs.

3. Methodology

The project is developed in four main phases: mathematical review and understanding, computational formulation of the problem, obtaining and analyzing results, and documentation. Below, these stages are briefly described.

A. Mathematical Review and Understanding. This phase involves reviewing the literature on KAN, PINN, and PIKAN, along with a detailed study of the Helmholtz equation. Sources will include academic journals, articles, and textbooks.

B. Computational Formulation of the Problem. A suitable loss function will be selected and implemented for the problem. The variational form of the Helmholtz equation will be established, and a KAN will be implemented in Python using appropriate packages. The network will be trained using the residual of the equation, which represents the difference between the left-hand and right-hand sides when an approximate solution is substituted.

C. Obtaining and Analyzing Results. In this phase, the network will be trained to solve the Helmholtz equation, optimizing its hyperparameters (number of layers and neurons per layer). The results will be validated using metrics such as the mean squared error, and the findings will be analyzed and discussed.

To compare and contrast both methodologies, the same equation will be solved for both cases, in its strong and weak form respectively, which is shown in equation 17, subject to $u(0) = 0 = u(1)$.

$$\nabla^2 f(x) - f(x) = 1 \qquad x \in (0,1) \qquad [17]$$

To understand the configuration of both networks presented, some definitions regarding the parameters of the implementation must be clarified:

- Hidden layer: an intermediary layer positioned between the input and output layers of the neural network. It is responsible for transforming the input data into abstract representations through learnable parameters.
- Neurons per layer: Defines the number of nodes (activation functions) that are present on each hidden network of a network

Some parameters that are present only on KAN are as follows:

- Grid: Number of control points on the splines at every node.
- K: Degree of the interpolator polynomials that make up the B-splines.

C.1. Network Configuration. A KAN with the following parameters was trained for 500 iterations, using a discretization of 1000 points:

- Number of hidden layers: 3
- Number of neurons per layer: 3
- Grid: 3
- K: 3
- Total number of parameters: 288

Figure 1 shows a schematic of how a trained KAN with the structure purposed looks. Note that each "node" contains three control points, shown as functions. The shade of the plot describes how important each activation function is for the output of the model. Every network constructed was built and trained using the open source **pykan** implementation (10) on python *3.11.4*.

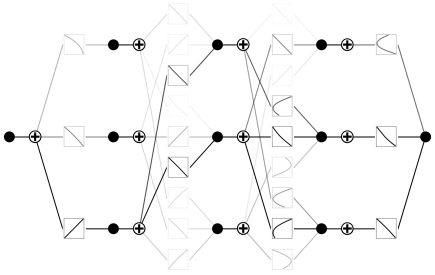

Fig. 1. Structure of a trained KAN

D. Documentation. Documentation will be conducted throughout the project, including interim reports and a final report. A GitHub repository will be used to ensure accessibility and reproducibility of the work (11).

4. Results

A. PIKANS. The PIKAN was trained for 500 iterations using equation 13 as a loss function, which took **3:32 minutes** to train. Figures 2 and 3 show instances of the training process of the network, in which one can sense the convergence of the algorithm.

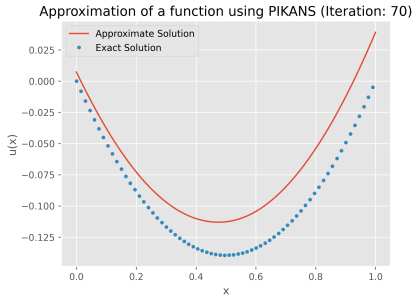

Fig. 2. Training of the network for a PIKAN, iteration 70

Figure 4 shows a Log-Log plot of how the loss function is reduced over the iterations. Note that it starts to respond around iteration 50. As well as that, considering that for this particular problem, the analytical solution is known, the L^2 error between the analytical solution and the predicted function is also plotted, noting that it also decreases over time, not as steeply as the loss function.

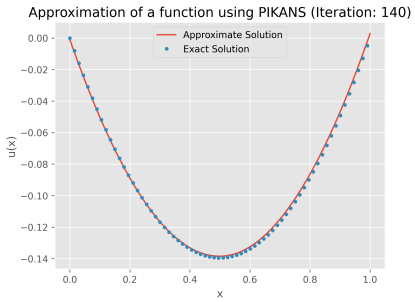

Fig. 3. Training of the network for a PIKAN, iteration 140

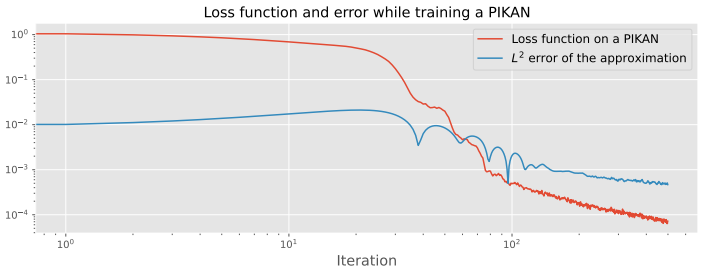

Fig. 4. Log-Log plot of the loss function

B. VPIKANS. The VPIKAN was trained for 500 iterations using the variational form of the Helmholtz equation which defines the loss described on equation **??**, which took **2:24 minutes** to train. Figures 5 and 6 show instances of the training process of the network, in which one can sense the convergence of the algorithm.

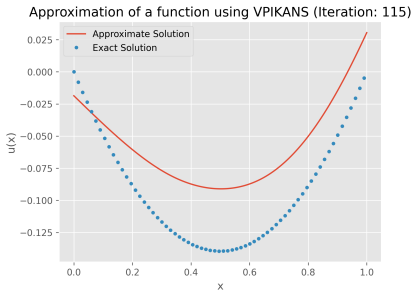

Fig. 5. Training of the network for a VPIKAN, iteration 115

Figure 7 shows a Log-Log plot of the loss function over the iterations. Note that, as the variational form of the equation lives on the H_0^1 space, the error on the H_0^1 with respect to the analytical solution can be defined. Note that the structure between the L^2 error and the H_0^1 error has a similar tendency, however, the H_0^1 norm is larger, as a result of the additional derivative term present in its definition.
Interestingly, the network only seems to start to respond after the 80^{th} iteration.

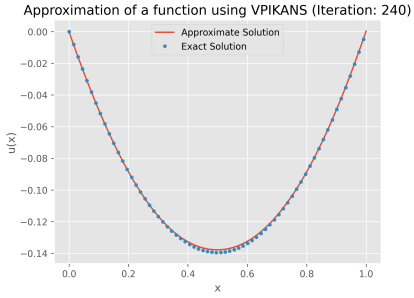

Fig. 6. Training of the network for a VPIKAN, iteration 270

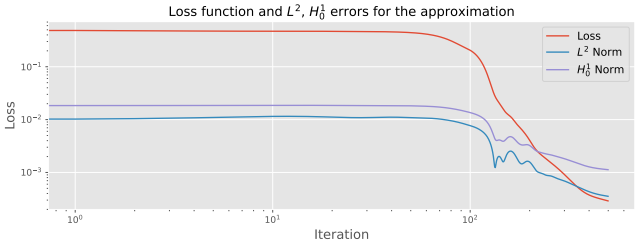

Fig. 7. Log-Log plot of the loss function and the associated errors

5. Discussion

A comparative analysis was conducted to evaluate the PIKANS and their variational counterpart (VPIKAN) in solving the Helmholtz equation. The following key observations were made for the problem under consideration:

- **PIKAN:** This method converges more rapidly (in fewer iterations) to an admissible result. Its efficiency lies in leveraging the direct residual minimization of the equation in its strong form, allowing quicker attainment of a solution that meets the accuracy threshold.
- **VPIKAN:** The variational form-based approach offers several advantages: it requires less computation time due to the reduced complexity of solving the weak form, as only a single derivative needs to be computed. Additionally, it achieves a lower error when evaluated in the L^2 norm, ensuring a more accurate overall approximation of the solution.

These findings highlight the trade-offs between the two methodologies, where PIKAN excels in convergence speed, while VPIKAN outperforms in terms of computational efficiency and accuracy under the L^2 metric, which is a metric in which both the strong and weak form of the equation make sense.

A. Comparison between a PIKAN methodology and a PINN methodology. In order to compare the PINKAN methodology established with the PINN frameworks, following the works of (7), the Helmholtz equation was solved using a PINN with roughly the same number of parameters as the PIKAN presented. Figure 8 shows the loss function on a PIKAN and a PINN, noting that the loss function is reduced nearly 10 times more in the same number of iterations when using a PIKAN framework.

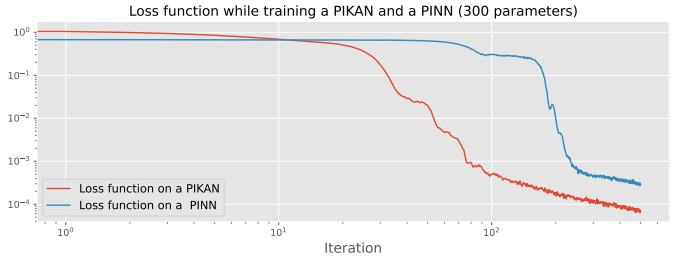

Fig. 8. Log-Log plot of the loss function using PIKAN and using PINN

B. On comparing the Finite Element Method with PIKANS for solving problems in 1D. Using the tools provided by the **FeniCs** project (12), a framework of open source code to solve differential equations using the finite element method (FEM), the Helmholtz equation was solved using roughly 250 degrees of freedom. After 500 iterations, the L^2 error between the analytical and the numerical solution was of 0.000002. This is orders of magnitude lower than the results obtained with PIKANS and shows how FEM still presents a better alternative for solving differential equations.

6. Conclusions

This study has demonstrated the potential of Kolmogorov-Arnold Networks (KANs) in addressing differential equations, specifically the Helmholtz equation, through their physics-informed variants: PIKAN and VPIKAN. The following conclusions can be drawn:

- **Comparison with Traditional Methods:** When compared to the Finite Element Method (FEM), both PIKAN and VPIKAN show promise but fall short in accuracy, with FEM outperforming in terms of error minimization. However, KAN-based methods offer significant interpretability and a framework that can potentially adapt to more complex scenarios.
- **Trade-offs Between PIKAN and VPIKAN:** PIKAN excels in convergence speed, while VPIKAN provides computational efficiency and improved accuracy. These trade-offs highlight the importance of selecting the appropriate methodology based on the specific requirements of the problem.
- **Future Implications:** The findings suggest that with further optimization and development, KANs could become a competitive alternative to traditional numerical methods, particularly in multidimensional and nonlinear problem domains.

Overall, this research contributes to the growing understanding of KANs' applicability in computational physics and engineering, providing a foundation for their use in solving partial differential equations.

7. Acknowledgements

A thanks to my wonderful advisor Nicolás Guarín-Zapata who guided me through the formulation and solution of the problem. Special thanks to Professor Elena Montilla for her guidance on how to formulate proper methodologies and objectives. I am, as always, most grateful for the unconditional support from my mother, brother, and aunt and extend my heartfelt gratitude to my family and friends.

8. References

1. G Evans, J Blackledge, P Yardley, *Numerical Methods for Partial Differential Equations*, Springer Undergraduate Mathematics Series. (Springer-Verlag, London, UK), (1999).
2. H Stoppels, “Solving the Helmholtz Equation Numerically,” Master project mathematics, Faculty of Science and Engineering (2018).
3. M Raissi, P Perdikaris, GE Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2018).
4. Z Liu, et al., Kolmogorov-arnold networks (kans). *arXiv preprint* (2024) Preprint, under review.
5. R Yu, W Yu, X Wang, Kan or mlp: A fairer comparison. *arXiv preprint* (2024) Preprint, under review.
6. J Schmidt-Hieber, The kolmogorov-arnold representation theorem revisited. *arXiv preprint* (2021) Preprint, under review.
7. JPA Calle, “Redes Neuronales Informadas por la Física para la Solución de Ecuaciones Diferenciales,” Tesis de grado, ingeniería física, Universidad EAFIT, Medellín, Colombia (2024).
8. Y Wang, et al., Kolmogorov-arnold-informed neural network: A physics-informed deep learning framework for solving pdes based on kolmogorov-arnold networks. *arXiv preprint* (2024) arXiv:2406.19756v1 [cs.LG].
9. K Shukla, JD Toscano, Z Wang, Z Zou, GE Karniadakis, A comprehensive and fair comparison between mlp and kan representations for differential equations and operator networks. *Prepr. submitted to Elsevier* (2024).
10. X Liu, Pykan: Implementation of kolmogorov-arnold networks (kans) in python (https://github.com/KindXiaoming/pykan) (2023) Accessed: 2024-11-19.
11. GP Bernal, Advanced project 1 (2024) Accessed: 2024-11-17.
12. The FEniCS Community, The fenics project (https://fenicsproject.org/) (2024) Accessed: 2024-11-19.