MACHINE LEARNING
Science and Technology

PURPOSE-LED PUBLISHING™ Part of the MACHINE LEARNING Series

PAPER • OPEN ACCESS

SPIKANs: separable physics-informed Kolmogorov–Arnold networks

To cite this article: Bruno Jacob *et al* 2025 *Mach. Learn.: Sci. Technol.* **6** 035060

IOP Publishing Mach. Learn.: Sci. Technol. 6 (2025) 035060 https://doi.org/10.1088/2632-2153/ae05af

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
27 May 2025

REVISED
1 August 2025

ACCEPTED FOR PUBLICATION
10 September 2025

PUBLISHED
22 September 2025

PAPER

SPIKANs: separable physics-informed Kolmogorov–Arnold networks

Bruno Jacob*, Amanda A Howard and Panos Stinis

Pacific Northwest National Laboratory, Richland, WA 99354, United States of America
* Author to whom any correspondence should be addressed.

E-mail: bruno.jacob@pnnl.gov, amanda.howard@pnnl.gov and panagiotis.stinis@pnnl.gov

Keywords: physics-informed neural networks, Kolmogorov–Arnold networks, separable physics-informed neural networks

Abstract

Physics-Informed Neural Networks (PINNs) have emerged as a promising method for solving partial differential equations (PDEs) in scientific computing. While PINNs typically use multilayer perceptrons (MLPs) as their underlying architecture, recent advancements have explored alternative neural network structures. One such innovation is the Kolmogorov–Arnold Network (KAN), which has demonstrated benefits over traditional MLPs, including faster neural scaling and better interpretability. The application of KANs to physics-informed learning has led to the development of Physics-Informed KANs (PIKANs), enabling the use of KANs to solve PDEs. However, despite their advantages, KANs often suffer from slower training speeds, particularly in higher-dimensional problems where the number of collocation points grows exponentially with the dimensionality of the system. To address this challenge, we introduce Separable Physics-Informed Kolmogorov–Arnold Networks (SPIKANs). This novel architecture applies the principle of separation of variables to PIKANs, decomposing the problem such that each dimension is handled by an individual KAN. This approach drastically reduces the computational complexity of training without sacrificing accuracy, facilitating their application to higher-dimensional PDEs. Through a series of benchmark problems, we demonstrate the effectiveness of SPIKANs, showcasing their superior scalability and performance compared to PIKANs and highlighting their potential for solving complex, high-dimensional PDEs in scientific computing.

1. Introduction

Physics-Informed Neural Networks (PINNs) have gained widespread attention for their ability to incorporate physical laws into machine learning models, allowing solutions to forward and inverse problems involving partial differential equations (PDEs) [1]. However, traditional PINNs struggle with computational costs when solving multi-dimensional, highly complex PDEs. This challenge is exacerbated by the need for a large number of collocation points, leading to high memory overhead and inefficient scaling as the problem's dimensionality increases [2].

As an alternative to the traditional multi-layer perceptron (MLP) architecture, Kolmogorov–Arnold networks (KANs) introduced by [3] and extended in [4] have prompted further exploration in the context of physics-informed learning. These networks, featuring trainable activation functions, have demonstrated promising results and superior performance compared to MLPs in terms of interpretability, robustness against catastrophic forgetting [5, 6], and resilience to noisy data [7]. To leverage these benefits within the PINNs framework, [3, 8] have proposed physics-informed KANs (PIKANs) and deep operator networks (DeepOKANs [9]). PIKANs have shown success on benchmark problems for physics-informed machine learning and applications. In [10], a fast JAX-based implementation of grid-dependent PIKANs was developed, achieving up to 84 times faster training times than the original KAN implementation through an adaptive state transition technique that avoids loss function peaks between grid extensions. GINN-KANs, introduced in [11], combines Growing Interpretable Neural Networks (GINN) and KANs to outperform both individual methods on Feynman symbolic regression benchmarks and shows improved performance

when applied to solving 15 different PDEs. In [12], the Kolmogorov–Arnold-Informed Neural Network (KINN) framework was proposed and formalized, applying KANs to different PDE forms (strong form, energy form, and inverse form) and demonstrating significant performance improvements over MLPs in accuracy and convergence speed for various PDEs including multiscale, stress concentration and nonlinear hyperelasticity problems. In [13], the Artificial Intelligence Velocimetry-Thermometry (AIVT) method was developed using physics-informed KANs to infer hidden turbulent temperature fields from experimental 3D Lagrangian velocity measurements in Rayleigh-Bénard convection, successfully reconstructing continuous temperature and velocity fields from sparse experimental data. In [14], an efficient wavelet-based (WAV-KAN) is used to construct improved PIKANs, demonstrating superior performance compared to conventional deep neural networks by achieving the same level of accuracy with fewer layers and reduced computational overhead for solving various ODEs and PDEs. In [15], PIKANs were used for power system dynamics, achieving superior accuracy in predicting rotor angle and frequency dynamics while employing fewer learnable parameters than conventional PINNs and successfully identifying inertia and damping coefficients. In [16], Higher-order-ReLU-KANs (HRKANs) were proposed, introducing simpler activation functions that both improve the efficiency of matrix operations and yield smooth non-zero higher-order derivatives, thus improving training stability, accuracy and convergence on the linear Poisson and nonlinear Burgers equations.

Recent advances have continued to enhance PIKANs through methodological improvements and hybrid architectures. Scalable Bayesian Physics-Informed KANs [17] combine dropout Tikhonov ensemble Kalman inversion with Chebyshev KANs and provide robust uncertainty quantification while addressing computational inefficiency for large-scale physics problems. Physics-informed KAN PointNet [18] allows the solution of multigeometry inverse problems for incompressible flow at a significantly lower computational cost compared to traditional PINNs. In biomedicine, the tanh-cPIKAN architecture [19] uses Chebyshev polynomials with additional non-linearity layers, providing systematic insights for the selection of representation models for systems pharmacology. For optimization in PIKANs, [20] demonstrate that the self-scaled BFGS and Broyden algorithms achieve orders of magnitude better accuracy without requiring adaptive weights. In addition, a novel Hybrid Parallel KAN and MLP PINN (HPKM-PINN) [21] connects KANs and MLPs through an adaptive scale factor, reducing relative error by up to two orders of magnitude compared to standalone architectures on several PDE benchmarks. A survey in [22] provides a comprehensive review of advances in network architecture, feature extension, and optimization techniques for PINNs and PIKANs.

Despite the advantages of KANs in physics-informed learning, the use of B-splines as basis functions for the activations in the original formulation leads to a significant increase in computational cost. Although previous work has proposed alternative basis functions (e.g. wavelets [23], radial basis functions [24], Chebyshev polynomials [25], Sinc [26]) that reduce computational time per iteration, the fundamental issue of exponential growth in the number of training points and network evaluations remains. Consequently, the curse of dimensionality associated with this architecture, combined with slower performance in KANs, limits the application of these networks to solving high-dimensional initial boundary value problems.

To address these limitations, we present Separable Physics-Informed Kolmogorov–Arnold Networks (SPIKANs). Inspired by Separable PINNs [27], we propose a network architecture that decomposes the solution of multi-dimensional PDEs into separable components, enabling more efficient training and inference. For a d-dimensional PDE with N^d sampled collocation points, instead of training one KAN with a cloud of $O(N^d)$ training points, SPIKANs decompose the problem into d KANs, each receiving $O(N)$ points as inputs. This approach substantially reduces computational costs in terms of time and memory, at the cost of requiring a factorizable mesh of collocation points rather than the traditional unstructured point-cloud used in PINNs.

This paper is organized as follows: in section 2, we briefly introduce KANs and PIKANs, along with the description of the proposed method, SPIKANs. In section 3, we demonstrate the use of SPIKANs in four benchmark problems, in increasing order of dimensionality, comparing gains in accuracy and speedup. Finally, we summarize the findings and discuss the limitations in section 4.

2. Methods

2.1. Kolmogorov–Arnold networks (KANs)

Kolmogorov-Arnold Networks (KANs) are a type of neural architecture proposed by [3]. They are inspired by the Kolmogorov–Arnold representation theorem, which states that a multivariate continuous function $u(\boldsymbol{x}) = u(x_1, x_2, \ldots, x_n)$ defined on a bounded domain can be expressed as a composition of continuous

univariate functions and sums [8]. In practice, [3] proposes approximating u as

$$u(x_1,\ldots,x_n) = \sum_{i_{L-1}=1}^{n_{L-1}} \phi_{L-1,i_L,i_{L-1}} \left(\sum_{i_{L-2}=1}^{n_{L-2}} \cdots \left(\sum_{i_0=1}^{n_0} \phi_{0,i_1,i_0}(x_{i_0}) \right) \right), \tag{1}$$

where L denotes the number of layers of the KAN, n_j are the number of nodes of the j-th layer, and $\phi_{i,j,k}$ are the activation functions. In the original KAN formulation [3], each activation function is parameterized as $\phi(x) = w_b b(x) + w_s \text{spline}(x)$, where $b(x)$ is a fixed basis function (typically $\text{silu}(x) = x/(1+e^{-x})$) and $\text{spline}(x) = \sum_i c_i B_i(x)$ is a linear combination of B-splines of order k. The learnable parameters are the weights w_b, w_s and the B-spline coefficients c_i. While alternative basis functions for the spline component have been explored, such as wavelets [23], radial basis functions [24] and Chebyshev [25] polynomials, this work uses B-splines following the original formulation. This structure imparts KANs with parameter efficiency and interpretability, especially for functions exhibiting compositional structures, outperforming traditional MLPs in these scenarios [8].

2.2. Physics-informed Kolmogorov–Arnold networks

In PIKANs, we seek to approximate the solution of a general initial-boundary value problem (IBVP), consisting of a partial differential equation (PDE) defined over a spatial domain Ω and a temporal domain Γ, along with corresponding initial and boundary conditions. Formally, the IBVP can be expressed as:

$$\mathcal{D}[u(\boldsymbol{x},t)] = f(\boldsymbol{x},t), \quad \boldsymbol{x}\in\Omega,\ t\in\Gamma, \tag{2}$$

$$\mathcal{I}[u(\boldsymbol{x},t)] = u_0(\boldsymbol{x}), \quad \boldsymbol{x}\in\Omega,\ t=0, \tag{3}$$

$$\mathcal{B}[u(\boldsymbol{x},t)] = g(\boldsymbol{x},t), \quad \boldsymbol{x}\in\partial\Omega,\ t\in\Gamma, \tag{4}$$

where $\mathcal{D}$ represents the differential operator defining the PDE, $u(\boldsymbol{x},t)$ is the solution, $f(\boldsymbol{x},t)$ is the source term, $u_0(\boldsymbol{x})$ denotes the initial condition at time $t=0$, and $g(\boldsymbol{x},t)$ specifies the boundary conditions on the boundary $\partial\Omega$ of the spatial domain. The operators $\mathcal{I}$ and $\mathcal{B}$ represent the application of initial and boundary conditions, respectively.

Using the KAN approximation as defined in equation (1), we seek to find an approximate solution $\hat{u}$ by minimizing the physics-informed loss functions. These loss functions, introduced by [1], are constructed as three distinct residuals obtained by summing over the interior, boundary, and initial collocation points of the domain. The residuals, denoted as $\mathcal{L}_{\text{pde}}$, $\mathcal{L}_{\text{ic}}$, and $\mathcal{L}_{\text{bc}}$, are defined as:

$$\mathcal{L}_{\text{pde}}(\theta) = \frac{1}{N_{\text{pde}}} \sum_{i=1}^{N_{\text{pde}}} \left| \mathcal{D}\left[\hat{u}\left(\boldsymbol{x}^i_{\text{pde}}, t^i_{\text{pde}};\, \theta\right)\right] - f\left(\boldsymbol{x}^i_{\text{pde}}, t^i_{\text{pde}}\right) \right|^2, \tag{5}$$

$$\mathcal{L}_{\text{ic}}(\theta) = \frac{1}{N_{\text{ic}}} \sum_{i=1}^{N_{\text{ic}}} \left| \mathcal{I}\left[\hat{u}\left(\boldsymbol{x}^i_{\text{ic}}, 0;\, \theta\right)\right] - u_0\left(\boldsymbol{x}^i_{\text{ic}}\right) \right|^2, \tag{6}$$

$$\mathcal{L}_{\text{bc}}(\theta) = \frac{1}{N_{\text{bc}}} \sum_{i=1}^{N_{\text{bc}}} \left| \mathcal{B}\left[\hat{u}\left(\boldsymbol{x}^i_{\text{bc}}, t^i_{\text{bc}};\, \theta\right)\right] - g\left(\boldsymbol{x}^i_{\text{bc}}, t^i_{\text{bc}}\right) \right|^2 \tag{7}$$

where θ is the set of trainable parameters of the KAN. The total loss function $\mathcal{L}$ used for training the PIKAN is constructed as a weighted sum of these individual loss components:

$$\mathcal{L}(\theta) = \lambda_{\text{pde}}\mathcal{L}_{\text{pde}}(\theta) + \lambda_{\text{ic}}\mathcal{L}_{\text{ic}}(\theta) + \lambda_{\text{bc}}\mathcal{L}_{\text{bc}}(\theta), \tag{8}$$

where the weighting parameters can be adjusted manually or in a self-adaptive manner (see, e.g. [28–30]).

For forward problems, we seek to find the optimal network parameters θ^* that minimize the total loss function $\mathcal{L}(\theta)$, ensuring that the predicted solution $\hat{u}(\boldsymbol{x},t;\theta^*)$ satisfies the governing PDE, initial conditions, and boundary conditions. This optimization problem can be formally expressed as:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta). \tag{9}$$

By solving the minimization problem in equation (9), we obtain the network parameters that yield the approximate solution $\hat{u} \approx u$. The optimization process is typically performed using gradient-based methods, where the gradients of the loss function with respect to the network parameters are computed via automatic differentiation.

2.3. Separable physics-informed Kolmogorov–Arnold networks

In the original construction of the KAN approximation, as presented in equation (1), all input variables are processed jointly within a single network architecture. Specifically, a KAN approximates a multivariate function $u : \mathbb{R}^n \to \mathbb{R}$, where $u(\boldsymbol{x}) = u(x_1, x_2, \ldots, x_n)$, by considering all variables simultaneously within a composite functional mapping. While this approach is effective for low-dimensional problems, it becomes computationally prohibitive for high-dimensional cases due to the exponential growth in the number of required collocation points and the increased complexity of the network architecture-a manifestation of the so-called curse of dimensionality.

One way to mitigate this is by decomposing the multivariate function $u(\boldsymbol{x}, t)$ into a separable representation. This idea, introduced by [27] for MLP-based PINNs, allows for a significant reduction in memory and computational cost, and can be seen as a tensor factorization. In the present work, we introduce the separable formulation for KANs: specifically, we approximate the solution $u(\boldsymbol{x}, t)$ as a finite sum of products of univariate functions, each depending on a single input variable:

$$u(\boldsymbol{x}, t) \approx \hat{u}(\boldsymbol{x}, t) = \sum_{j=1}^{r} \prod_{i=1}^{n} f_j^{(\theta_i)}(x_i), \tag{10}$$

where $r \in \mathbb{N}$ is the rank of the approximation (also referred to as the latent dimension), n is the number of spatial dimensions, $f_j^{(\theta_i)} : \mathbb{R} \to \mathbb{R}$ are univariate functions parameterized by θ_i, and x_i represents the i-th input variable. The time variable t can be included as an additional dimension.

By adopting this separable representation, we effectively decompose the multivariate function into a sum over r terms, each of which is a product of univariate functions. Each univariate function $f^{(\theta_i)}(x_i)$ can be approximated using a separate KAN or another suitable univariate approximator. This decomposition transforms the high-dimensional approximation problem into multiple one-dimensional problems, significantly reducing computational complexity.

The theoretical foundation for this approach, which applies to general (and not necessarily separable) PDEs, is established through the universal approximation property of SPIKANs, presented in theorem 1 of appendix B. This theorem guarantees that for any square-integrable function and tolerance $\varepsilon > 0$, there exists a SPIKAN of sufficient rank r and univariate KAN complexity that can approximate the target function within the specified tolerance. However, we emphasize that theorem 1 only proves existence, and does not provide constructive bounds on the required rank r or grid parameters. In practice, the rank required for accurate approximation depends on the intrinsic separability of the target function, which can be prohibitively large for highly non-separable problems.

To incorporate this separable approximation into the PIKAN framework, we adjust the physics-informed loss functions accordingly. The predicted solution $\hat{u}(\boldsymbol{x}, t)$ is defined as per equation (10). The partial derivatives of $\hat{u}$ with respect to each input variable x_i are computed using the product rule:

$$\frac{\partial \hat{u}}{\partial x_i} = \sum_{j=1}^{r} \left(f_j^{(\theta_i)\prime}(x_i) \prod_{k=1,\, k\neq i}^{n} f_j^{(\theta_k)}(x_k) \right), \tag{11}$$

where $f_j^{(\theta_i)\prime}(x_i)$ denotes the derivative of $f_j^{(\theta_i)}$ with respect to x_i. Higher-order derivatives can be computed similarly by applying the product rule iteratively. We also define $f^{(\theta_i)} = (f_1^{(\theta_i)}, \ldots, f_r^{(\theta_i)})$.

A key advantage of the separable representation in SPIKANs is that each KAN approximates a univariate function $f^{(\theta_i)} : \mathbb{R} \to \mathbb{R}^r$. This design enables the efficient computation of the derivatives required to apply the differential operator $\mathcal{D}$ to the approximation $\hat{u}$, as shown in equation (11), by leveraging forward-mode automatic differentiation. Forward-mode is particularly effective for functions with a single input and multiple outputs, which aligns perfectly with the structure of each $f^{(\theta_i)}$ in the separable framework. In contrast, reverse-mode automatic differentiation is optimized for functions that have many inputs and a single output and is commonly utilized in traditional PIKANs. For a more detailed discussion on this topic, see [27].

The approximation of the solution u follows the same construction of the PINN and PIKAN loss, i.e. equations (5)–(7). A diagram illustrating the full architecture of the proposed SPIKAN method is shown in figure 1. The complete SPIKAN algorithm, including the handling of immersed boundaries, is provided in algorithm 1 in the appendix A.

2.3.1. Vector-valued outputs

Many physical systems are governed by coupled PDEs with multiple unknown fields. For instance, the incompressible Navier–Stokes equations involve velocity components and pressure, while reaction-diffusion

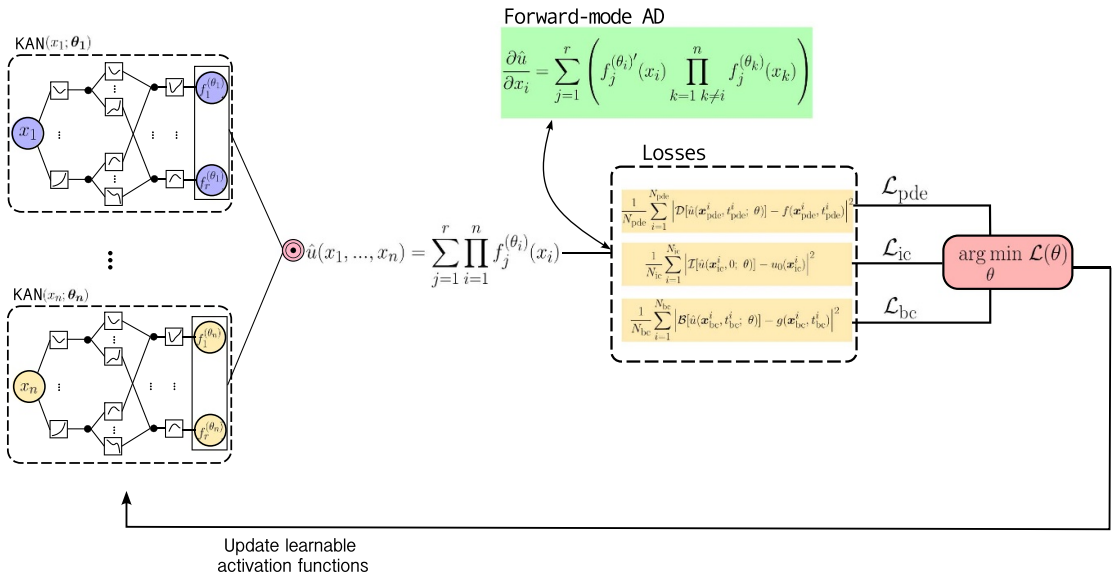

Figure 1. Schematic of the SPIKAN architecture. We decompose the n-dimensional PDE into n individual KANs. In contrast with PIKANs, which take as input points given by tuples $(x_1, \ldots, x_n)$, in SPIKANs, each network receives only one coordinate x_i at a time. Each network produces r outputs, which are then combined via an outer product and summation on the latent dimension (denoted by the $\odot$ operator) to approximate the field $\hat{u}(x_1, \ldots, x_n)$. We use forward-mode automatic differentiation to evaluate the derivatives of $\hat{u}$ needed to satisfy the differential operator $\mathcal{D}$ and any other derivative needed to construct the losses. The weights of the networks are then adjusted via gradient descent.

systems may include multiple chemical species concentrations. The SPIKAN framework naturally extends to such vector-valued problems by leveraging the separable representation for each output component.

Consider a system with m output fields $\mathbf{u} = (u^{(1)}, u^{(2)}, \ldots, u^{(m)})$, where each field $u^{(k)} : \Omega \times \Gamma \to \mathbb{R}$. The separable approximation is applied independently to each component:

$$u^{(k)}(\boldsymbol{x}, t) \approx \hat{u}^{(k)}(\boldsymbol{x}, t) = \sum_{j=1}^{r} \prod_{i=1}^{n} f_{j,k}^{(\theta_i)}(x_i), \tag{12}$$

where $f_{j,k}^{(\theta_i)}$ denotes the jth latent component for the kth output field from the ith univariate network. In practice, this is implemented by having each univariate KAN produce $r \times m$ outputs, which are then reshaped and combined appropriately.

For a coupled system of PDEs represented by the differential operator $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m)$, the physics-informed loss function becomes:

$$\mathcal{L}_{\text{pde}}(\theta) = \frac{1}{N_{\text{pde}}} \sum_{i=1}^{N_{\text{pde}}} \sum_{k=1}^{m} \lambda_k \left| \mathcal{D}_k \left[\hat{\mathbf{u}}\left(\boldsymbol{x}_{\text{pde}}^i, t_{\text{pde}}^i; \theta\right) \right] - f_k\left(\boldsymbol{x}_{\text{pde}}^i, t_{\text{pde}}^i\right) \right|^2, \tag{13}$$

where λ_k are optional weighting factors for balancing the contribution of different equations. For simplicity, in the present work we assume $\lambda_k = 1$. The computation of derivatives for each field component follows the same product rule as in equation (11), maintaining the computational efficiency of the separable formulation.

Note that this vector-valued extension preserves the $O(N)$ complexity of SPIKANs, as the additional cost scales linearly with the number of output fields m, which is typically small compared to the number of collocation points N. The approach has been successfully demonstrated in the solution of the incompressible Navier–Stokes equations (with $\mathbf{u} = (u, v, p)$ representing the velocity components and pressure) and in the coupled fluid-thermal problem (with $\mathbf{u} = (u, v, p, T)$ including temperature fields) shown in the Results section.

2.3.2. Extension to non-factorizable grids

While the separable representation in SPIKANs requires factorizable collocation grids, many practical applications involve complex geometries that do not naturally conform to rectangular domains. To address this limitation, we propose an extension to the SPIKAN framework for immersed boundaries that maintains the computational efficiency of the separable formulation while accommodating arbitrary geometries.

Consider a domain Ω containing an immersed body Ω_b, where the computational domain is $\Omega_c = \Omega \setminus \Omega_b$. The key insight is that we can preserve the separable structure on the regular Cartesian grid while selectively excluding points that lie within the immersed body. This is achieved through a masking strategy applied to the physics-informed loss functions. More specifically, for interior collocation points $(\boldsymbol{x}_i, t_i)$ on the factorizable grid, we define an indicator function:

$$\chi(\boldsymbol{x}_i) = \begin{cases} 1, & \text{if } \boldsymbol{x}_i \in \Omega_c, \\ 0, & \text{if } \boldsymbol{x}_i \in \Omega_b. \end{cases} \tag{14}$$

The physics-informed loss is then modified to incorporate this mask:

$$\mathcal{L}_{\text{pde}}(\theta) = \frac{1}{N_{\text{valid}}} \sum_{i=1}^{N_{\text{pde}}} \chi(\boldsymbol{x}_i) \left| \mathcal{D}[\hat{u}(\boldsymbol{x}_i, t_i; \theta)] - f(\boldsymbol{x}_i, t_i) \right|^2, \tag{15}$$

where $N_{\text{valid}} = \sum_{i=1}^{N_{\text{pde}}} \chi(\boldsymbol{x}_i)$ ensures proper normalization by counting only the valid collocation points. In this context, valid points refer to points that represent the *de facto* physical domain.

For boundary conditions on the immersed body $\partial\Omega_b$, the separable formulation must be locally relaxed. While boundaries aligned with the coordinate axes (such as the domain boundaries $\partial\Omega$) can maintain the separable structure, curved boundaries require evaluation at non-factorizable points. For these boundaries, we implement a hybrid approach where the separable networks are evaluated point-wise:

$$\hat{u}(\boldsymbol{x}_b) = \sum_{j=1}^{r} \prod_{i=1}^{n} f_j^{(\theta_i)}(x_{b,i}), \quad \boldsymbol{x}_b \in \partial\Omega_b, \tag{16}$$

where each univariate function $f_j^{(\theta_i)}$ is evaluated at the specific coordinate $x_{b,i}$ of the boundary point. We highlight that despite the increase in computational complexity from the point-wise evaluations on $\partial\Omega_b$, the overall algorithm maintains a complexity of $O(N) + O(N_b^{d-1})$, where N_b is the number of boundary points on the immersed surface and d is the spatial dimension. This remains dominated by $O(N)$ when $N \gg N_b^{d-1}$, which is typically the case for moderate dimensionality and well-resolved interior grids.

This approach has been successfully demonstrated in two benchmark problems: flow around a cylinder in a lid-driven cavity and natural convection around a heated cylinder. In both cases, the immersed cylinder is handled through masking of interior points and point-wise evaluation on the cylinder boundary, while maintaining the $O(N)$ computational complexity for the majority of the domain. The physics-informed losses for these multi-field problems naturally extend to vector-valued solutions, with the masking applied component-wise to ensure that all field variables satisfy the governing equations only in the valid computational domain.

3. Results

In this section, we present a validation study of the proposed method, as well as comparisons with the original implementation of PIKANs [8]. All the networks in this work, including the base PIKANs, are trained with Adam optimizer, with an initial learning rate $l = 10^{-3}$ and hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The total number of epochs used for training is specified in each subsection. All PIKANs and SPIKANs tested used a constant KAN grid size of $g = 3$. We leave the analysis of the multigrid aspect of KANs and PIKANs [10] via variable g for future work. We use k to denote the degrees of the B-splines in both PIKANs and SPIKANs.

For PIKANs, the size of the network is described as $[w_{\text{in}}, w_{h_1}, \ldots, w_{h_n}, w_{\text{out}}]$, where $w_{\text{in}}, w_{h_j}, w_{\text{out}}$ denotes the width of the input layer, j-th hidden layer and output layers, respectively. Similarly, for SPIKANs, a network of size $d \times [w_{\text{in}} = 1, w_{h_1}, \ldots, w_{h_n}, r * w_{\text{out}}]$ denotes width of input, hidden and output layers, with additional hyperparameters d and r indicating the dimensionality of the PDE (total number of networks used) and the latent dimension, respectively. Notice that for SPIKANs, $w_{in} = 1$ by construction (cf figure 1).

Throughout this section, for both PIKANs and SPIKANs, we have used an unbiased loss function, where $\lambda_{\text{ic}} = \lambda_{\text{bc}} = \lambda_{\text{pde}} = 1$. While various methods exist for automatic weight selection, including learning rate annealing [31, 32], self-adaptive weights [28], neural tangent kernel approaches [33], and residual-based attention [29], we chose to maintain fixed weights, in order to hopefully isolate the effect of the separable architecture. Moreover, recent work by [34] has shown that loss weights can have adversarial and somewhat unpredictable effects in PIKANs. We believe this complex interplay between loss weighting strategies and KAN architectures deserves a separate, in depth investigation, which we leave for future work.

Table 1. Comparison of methods for the 2D Helmholtz boundary value problem, equations (18)–(22).

Method	KAN Size	n_{cp}	No. Parameters	L_2(%)	Time (ms/iter)	Speedup
PIKAN (a)	[2,6,6,1]	100^2	445	37.26	20.75	1 (baseline)
SPIKAN (a)	$2 \times [1,3,3,1*5]$	100^2	454	1.29	2.65	7.81
PIKAN (b)	[2,9,9,1]	100^2	883	15.73	132.63	1 (baseline)
SPIKAN (b)	$2 \times [1,5,5,1*5]$	100^2	910	2.91	2.73	48.62
PIKAN (c)	[2,9,9,1]	200^2	883	228.84	961.54	1 (baseline)
SPIKAN (c)	$2 \times [1,5,5,1*5]$	200^2	910	2.20	3.35	287.08

The comparison between predicted and reference results using the L_2 norm is measured as:

$$L_2 = \sqrt{\frac{\sum_{i=1}^{n_{cp}} |u^{ref}(\boldsymbol{x}_i) - \hat{u}(\boldsymbol{x}_i)|^2}{\sum_{i=1}^{n_{cp}} |u^{ref}(\boldsymbol{x}_i)|^2}}, \tag{17}$$

where u^{ref} denotes the reference solution, $\hat{u}$ is the approximate solution and n_{cp} is the number of collocation points.

The implementations used in this work [35] are based on `Jax` [36], using the `Jax-KAN` package [10, 37] for the KAN implementation. All simulations were performed using an Nvidia T4 Tensor Core GPU with 16GB of vRAM.

3.1. 2D Helmholtz equation

Consider the following boundary value problem for the 2D Helmholtz equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \kappa^2 u = q(x,y), \quad (x,y) \in \Omega, \tag{18}$$

subject to the homogeneous Dirichlet boundary conditions:

$$u(-1,y) = 0, \quad -1 \leqslant y \leqslant 1 \tag{19}$$
$$u(1,y) = 0, \quad -1 \leqslant y \leqslant 1 \tag{20}$$
$$u(x,-1) = 0, \quad -1 \leqslant x \leqslant 1 \tag{21}$$
$$u(x,1) = 0, \quad -1 \leqslant x \leqslant 1 \tag{22}$$

where $\Omega = \{(x,y) \in [-1,1]^2\}$. The forcing term $q(x,y)$ is given by:

$$\begin{aligned} q(x,y) = &-(a_1\pi)^2 \sin(a_1\pi x)\sin(a_2\pi y) + \sin(a_1\pi x)\sin(a_2\pi y) \\ &-(a_2\pi)^2 \sin(a_1\pi x)\sin(a_2\pi y) + \kappa^2 \sin(a_1\pi x)\sin(a_2\pi y), \end{aligned} \tag{23}$$

which is obtained via the manufactured solution

$$u(x,y) = \sin(a_1\pi x)\sin(a_2\pi y). \tag{24}$$

We consider the case $\kappa = 1$, $a_1 = 1$ and $a_2 = 4$, as in [8].

Due to the difference in the architecture of PIKANs and SPIKANs, we perform tests with different number of neurons in the hidden layers, but focusing on a similar number of trainable parameters. Since the problem is in two dimensions, in the SPIKAN cases, the reported KAN sizes refer to the two networks that compose the SPIKAN, and the last size corresponds to the latent dimension, set to $r = 5$ for all the cases. For both PIKANs and SPIKANs, we use B-splines of degree $k = 3$. The optimization was performed using the Adam optimizer with initial learning rate 10^{-3} for 20 000 epochs. The collocation points are arranged in an equidistant manner, forming a grid of $n_{cp} = n_x \times n_y$ collocation points in the Ω domain.

Table 1 and figure 2 summarize the results of all tested cases. The iteration times were computed as an average over all the epochs, and were all produced in the same hardware environment. We note that SPIKANs achieve speedups ranging from $O(10)$ to $O(100)$, by virtue of the reduced number of network evaluations from the now one-dimensional inputs. The calculations of the L_2 error are performed with respect to the analytical solution, equation (24).

While the advantages of SPIKANs lie in decreasing the computational intesity of training, SPIKANs notably also achieved superior accuracy for all the tested cases, with significant decreases in computational time. For the PIKAN case (c), we noticed a deterioration in the quality of the solution, possibly due to the

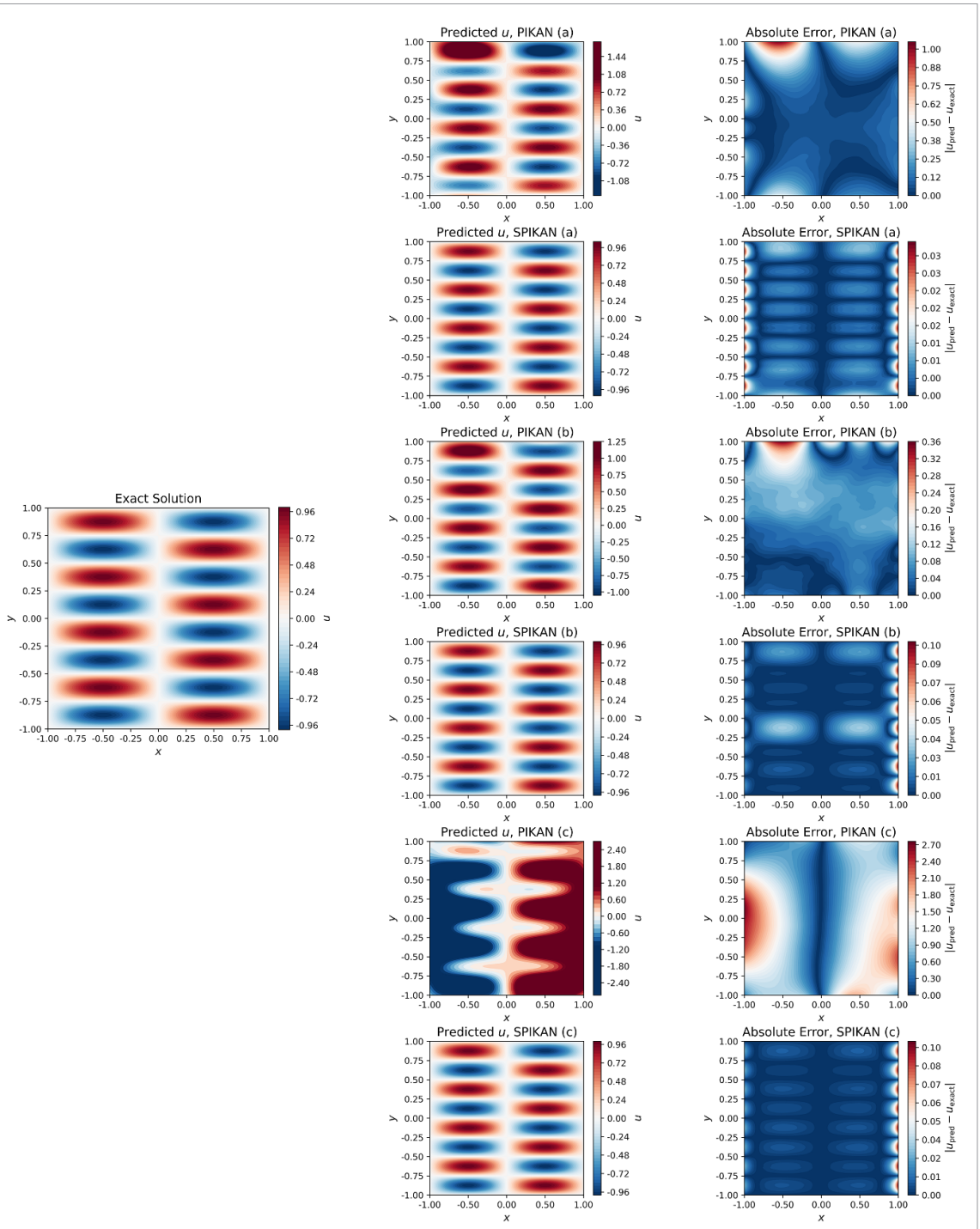

Figure 2. Contour plots and absolute error between the predicted solution u obtained with PIKANs, SPIKANs and the analytical solution of the 2D Helmholtz boundary value problem (18)–(22). A description of the cases is shown in table 1.

fixed number of epochs used for all the networks. This could be caused by the fixed number of epochs used in all the examples; larger PIKANs, exposed to larger training datasets, could require longer to converge.

It is important to highlight that the reduction in computational cost does not come at the cost of an increase in the memory; on the contrary, memory utilization is reduced from $O(n_x * n_y + \text{No. Parameters})$ to roughly $O(d * \sqrt{n_x * n_y} + \text{No. Parameters})$, with $d = 2$ for the 2D Helmholtz example. For a detailed description of the memory utilization in separable networks, we refer the reader to [27].

3.2. 2D steady lid-driven cavity flow

In this section, we demonstrate usage of SPIKANs in a multiple output setup: we solve the 2D steady, incompressible Navier–Stokes equations in an enclosed, lid-driven cavity. In this benchmark, the flow is

Table 2. Comparison of methods for the 2D steady Navier–Stokes (lid-driven cavity) for Re = 100.

Method	KAN Size	n_{cp}	No. Parameters	$L_2(u,v,p)$ (%)	Time (ms/iter)	Speedup
PIKAN	[2,9,9,3]	50^2	1029	(9.73, 11.80, 48.71)	301.20	1 (baseline)
SPIKAN (a)	$2 \times [1,5,5,3*5]$	50^2	2150	(8.66, 12.01, 35.91)	4.3	70.05
SPIKAN (b)	$2 \times [1,5,5,3*5]$	100^2	2150	(5.83, 7.18, 29.55)	6.14	49.05

driven by a constant tangential velocity applied to the lid of the cavity. The governing equations are given by

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{25}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{26}$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \tag{27}$$

where $(x,y) \in \Omega = [0,L]^2$, u and v are the velocity components in the x and y-directions, respectively, and p is the pressure field. Here, Re denotes the Reynolds number, which characterizes the ratio of inertial forces to viscous forces in the flow. In the example below, we consider $Re = 100$, $U_0 = 1$ and $L = 1$.

The boundary conditions are specified as follows:

$$u(x,L) = U_0, \quad v(x,L) = 0, \quad 0 \leqslant x \leqslant L \quad \text{(top boundary, moving lid)}, \tag{28}$$

$$u(x,0) = 0, \quad v(x,0) = 0, \quad 0 \leqslant x \leqslant L \quad \text{(bottom boundary, no-slip)}, \tag{29}$$

$$u(0,y) = 0, \quad v(0,y) = 0, \quad 0 \leqslant y \leqslant L \quad \text{(left boundary, no-slip)}, \tag{30}$$

$$u(L,y) = 0, \quad v(L,y) = 0, \quad 0 \leqslant y \leqslant L \quad \text{(right boundary, no-slip)}. \tag{31}$$

Similarly to the 2D Helmholtz example, the optimization was performed using the Adam optimizer with initial learning rate 10^{-3}. Training is performed in 50 000 epochs, with B-splines of degree $k = 5$ for both PIKAN and SPIKANs. The collocation points are arranged in an equidistant manner, forming a grid of $n_{cp} = n_x \times n_y$ points, uniformly distributed within the Ω domain.

The reference fields were obtained numerically, using the finite volume method (FVM). The steady-state SIMPLE [38] algorithm was employed to resolve the incompressible Navier–Stokes equations in a equally spaced 256x256 cell mesh. Convective fluxes were approximated with a first-order upwind scheme to maintain stability, while the diffusive terms were discretized using a second-order Gauss linear scheme. A pressure reference was specified to ensure convergence, with a target tolerance of 10^{-4} for pressure and 10^{-5} for the velocity components. The iterative solver configuration utilized the Preconditioned Conjugate Gradient (PCG) method for the pressure field and a Gauss-Seidel smoother for the velocity field, with relaxation factors of 0.3 for pressure and 0.7 for velocity to enhance convergence stability. In addition, we compare the vertical and horizontal centerline velocity profiles with the benchmark results of [39].

Table 2 summarizes the results of the 2D lid-driven cavity flow at Re = 100, equations (25)–(31). We use a PIKAN with architecture of shape [2,9,9,3] as a baseline. The L_2 differences are compared with the FVM results as reference. We normalize the pressure field p by removing its mean and scaling it with the maximum value, in order to achieve a translationally invariant field across all the methods tested.

Figure 3 shows a comparison of the centerline velocity profiles of the baseline PIKAN ($[2,9,9,3], n_{cp} = 50^2$) and SPIKANs (a) ($2 \times [1,5,5,3*5], n_{cp} = 50^2$) and (b) ($2 \times [1,5,5,3*5], n_{cp} = 100^2$). Despite all results being close in terms of accuracy, the SPIKAN (b) outperforms the other cases, with the lowest L_2. Increasing the number of collocation points for training from $n_{cp} = 50^2$ to $n_{cp} = 100^2$ brings the velocity profiles closer to the references. Even though this might seem an obvious result, we highlight that doubling the number of collocation points in each dimension would lead to a quadruple ($O(2^d)$) increase in computational time in PIKANs, but only a linear increase in SPIKANs. This effect becomes more prominent in higher dimensions, as we demonstrate in section 3.4.

Another important result is that SPIKAN (a) achieved similar accuracy of the baseline PIKAN with a 70× speedup. Even with a finer resolution of $n_{cp} = 100^2$, SPIKAN (b) still outperformed PIKAN and trained nearly 50 × faster.

Contour plots of the velocity magnitude $||\mathbf{u}||$, along with the streamlines of the FVM reference, PIKAN and SPIKANs (a) and (b) are shown in figure 4. Despite having streamlines crossing the wall boundaries, notice that the SPIKAN predictions were far closer to the reference than PIKANs, especially in the lateral walls, $x = 0$ and $x = 1$. This nonphysical behavior is a consequence of using a weak formulation of boundary

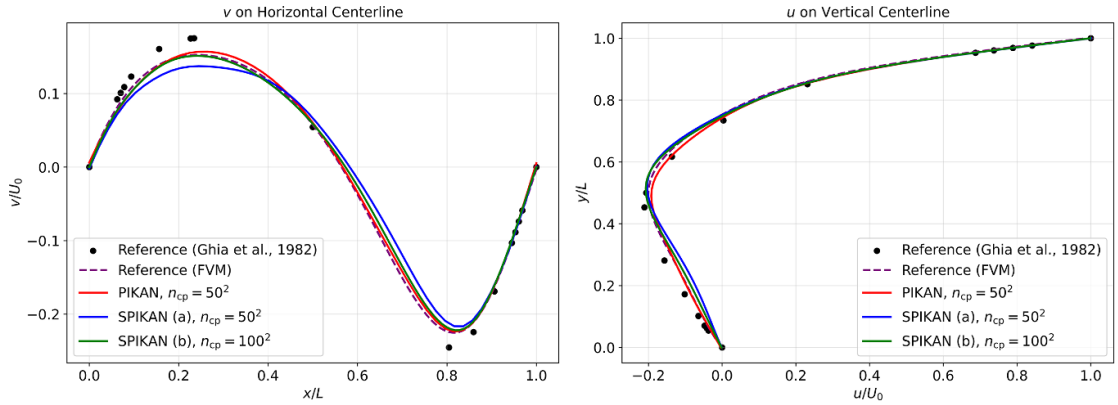

Figure 3. Comparison of horizontal and vertical velocity profiles obtained with PIKAN ($[2,5,5,3], n_{cp} = 50^2$)) and SPIKANs ($2 \times [1,9,9,3*5], n_{cp} = 50^2, 100^2$) with the [39] for the steady lid-driven cavity flow at Re = 100, equations (25)–(27). Reprinted from [39], Copyright (1982), with permission from Elsevier.

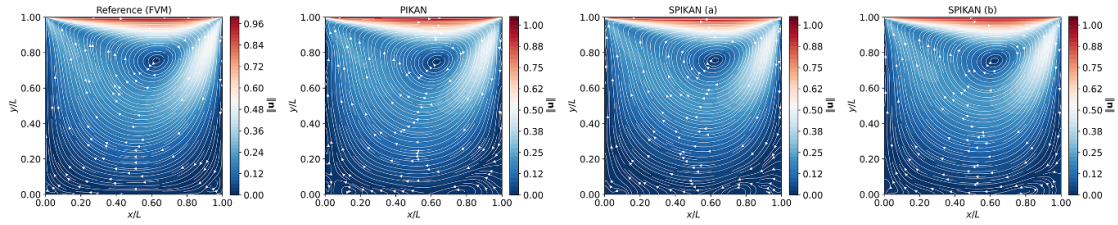

Figure 4. Comparison of the isocontours of $||\mathbf{u}|| = \sqrt{u^2 + v^2}$ and streamlines, obtained with finite volume method, PIKAN ($[2,9,9,3], n_{cp} = 50^2$)), SPIKAN (a) ($2 \times [1,5,5,3*5], n_{cp} = 50^2$) and SPIKAN (b) ($2 \times [1,5,5,3*5], n_{cp} = 100^2$), for the steady lid-driven cavity flow, equations (25)–(31).

conditions, which in turn lead to larger errors near the boundaries. Figure 5 further highlights this, by showing the absolute difference of fields u, v and p with the reference FVM counterpart.

3.3. 1D+1 Allen-Cahn equation

In this example, we test the proposed architecture in what has been shown to be a challenging case for PINNs and PIKANs: the Allen-Cahn equation. This nonlinear PDE describes the reaction-diffusion in phase separation, and is widely used in studies of alloy separation. In its one-dimensional form, the equation is given by

$$\frac{\partial u}{\partial t} - D\frac{\partial^2 u}{\partial x^2} + 5\left(u^3 - u\right) = 0. \tag{32}$$

We examine the case with $D = 10^{-4}$, $(x,t) \in [-1,1] \times [0,1]$. The initial and boundary conditions are the same ones used by [8]:

$$u(x,0) = x^2 \cos(\pi x), \tag{33}$$

$$u(-1,t) = u(1,t) = -1. \tag{34}$$

Previous studies have reported training difficulties for this system in the context of PINNs [40, 41] and PIKANs [8]. These difficulties have been attributed to the presence of a non-trivial fixed point, which typically corresponds to a global minima of the physics loss term $\mathcal{L}_{PDE}$ shown in equation (5) with non-trivial basins of attraction [42]. As a consequence, the solution can only be learned with specific choices of $\lambda_{ic}, \lambda_{bc}$, such that $\lambda_{ic}, \lambda_{bc} \gg \lambda_{pde}$ (cf equation (8)), or under variations in architecture (e.g. via multi-fidelity learning, as in [43, 44]) or adaptive weighting strategies [29]. For the sake of clarity, we limit the scope of the hyperparameter space and focus on the case of an unbiased loss function, where $\lambda_{ic} = \lambda_{bc} = \lambda_{pde} = 1$. This choice allows for a fair analysis of the proposed method in isolation from the effects of correction factors to the loss terms.

In this section, we compare the solutions obtained via PIKAN and SPIKAN training, focusing on the effect of the latent dimension r in the SPIKAN architecture in the accuracy of the solution. A summary of the cases tested is shown in table 3. The reference result, used to compute the L_2 and absolute errors, is obtained

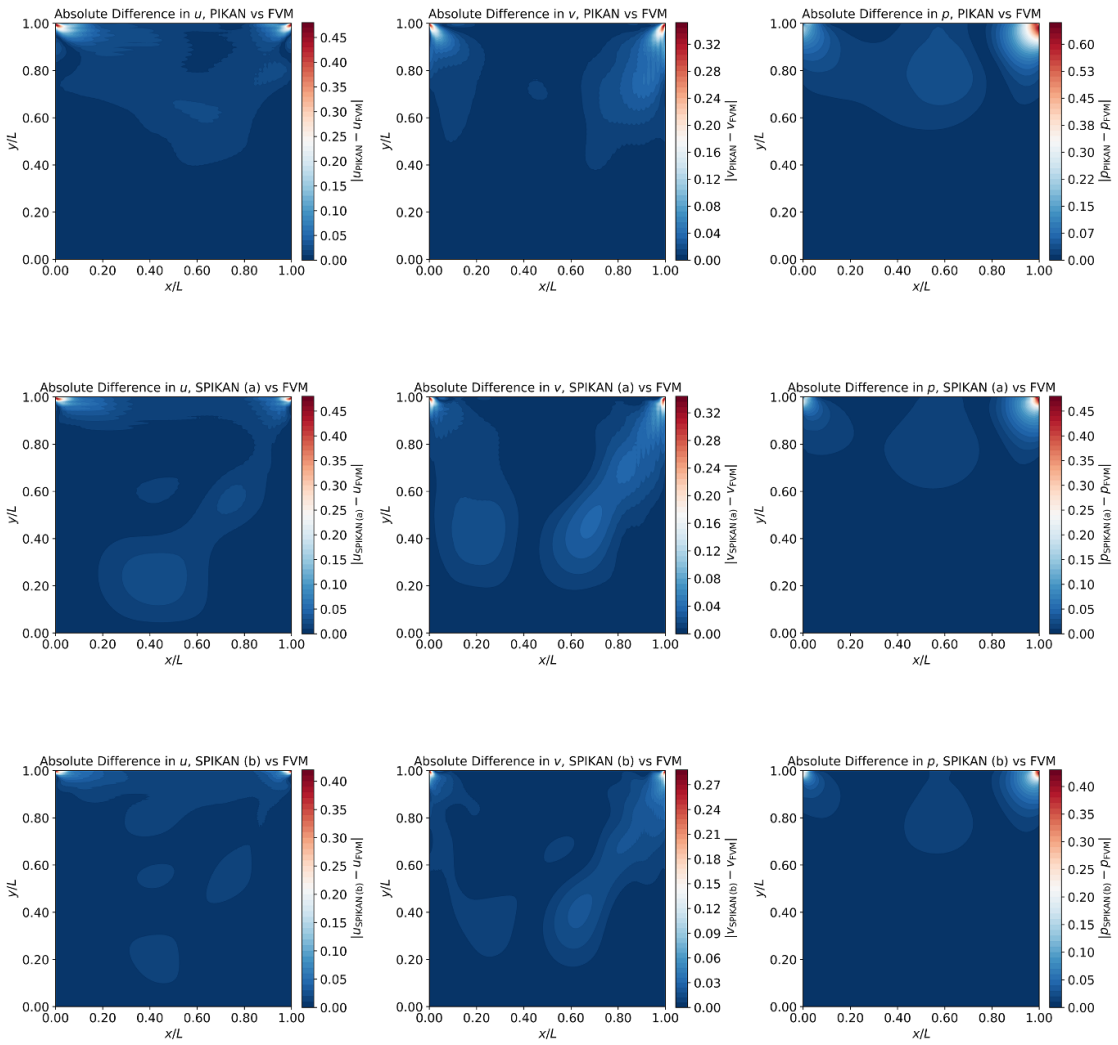

Figure 5. Comparison of absolute difference of predicted u, v and p fields obtained from PIKAN ($[2, 9, 9, 3], n_{cp} = 50^2$) and SPIKAN ($2 \times [1, 5, 5, 3 * 5], n_{cp} = 50^2$) versus the results obtained with FVM for Re $= 100$, for the steady lid-driven cavity flow, equations (25)–(31).

Table 3. Comparison of methods for the 1D+1 Allen-Cahn initial-boundary value problem, equations (32)–(34).

Method	KAN Size	n_{cp}	No. Parameters	L_2 (%)	Time (ms/iter)	Speedup
PIKAN	[2,9,9,1]	320×160	1099	52.92	454.54	1 (baseline)
SPIKAN (a)	$2 \times [1, 5, 5, 1 * 5]$	320×160	1130	94.18	3.53	128.76
SPIKAN (b)	$2 \times [1, 5, 5, 1 * 10]$	640×320	1640	17.15	4.36	104.25
SPIKAN (c)	$2 \times [1, 5, 5, 1 * 20]$	320×160	2660	28.54	5.22	87.08

by solving the initial-boundary value problem equations (32)–(34) numerically using a Fourier pseudospectral method in space and a 4th order Runge-Kutta method in time, on a $n_x = 320, n_t = 1000$ resolution.

Table 3 summarizes the results. As a baseline, we used a PIKAN of size [2,9,9,1], with B-splines of degree $k = 5$ and $n_{cp} = 320 \times 160$ collocation points. We explore three SPIKAN cases, all with $k = 5$ but with increasing values of r. For all cases, we used the Adam optimizer with an initial learning rate of 10^{-3} for 150 000 epochs.

Contour plots of u for the reference solution and the aforementioned networks are shown in figure 6. For the PIKAN solution, our results are in agreement with what was reported in [8], in which u did not converge to the reference solution. This behavior of $u < 0$ is consistent with the local minima observed when the optimization converges to the fixed point, as described in [42]. For the SPIKAN (a) tested, with $r = 5$, we note a similar behavior as of the PIKAN predictions, although obtained with a nearly 70 $\times$ speedup. For

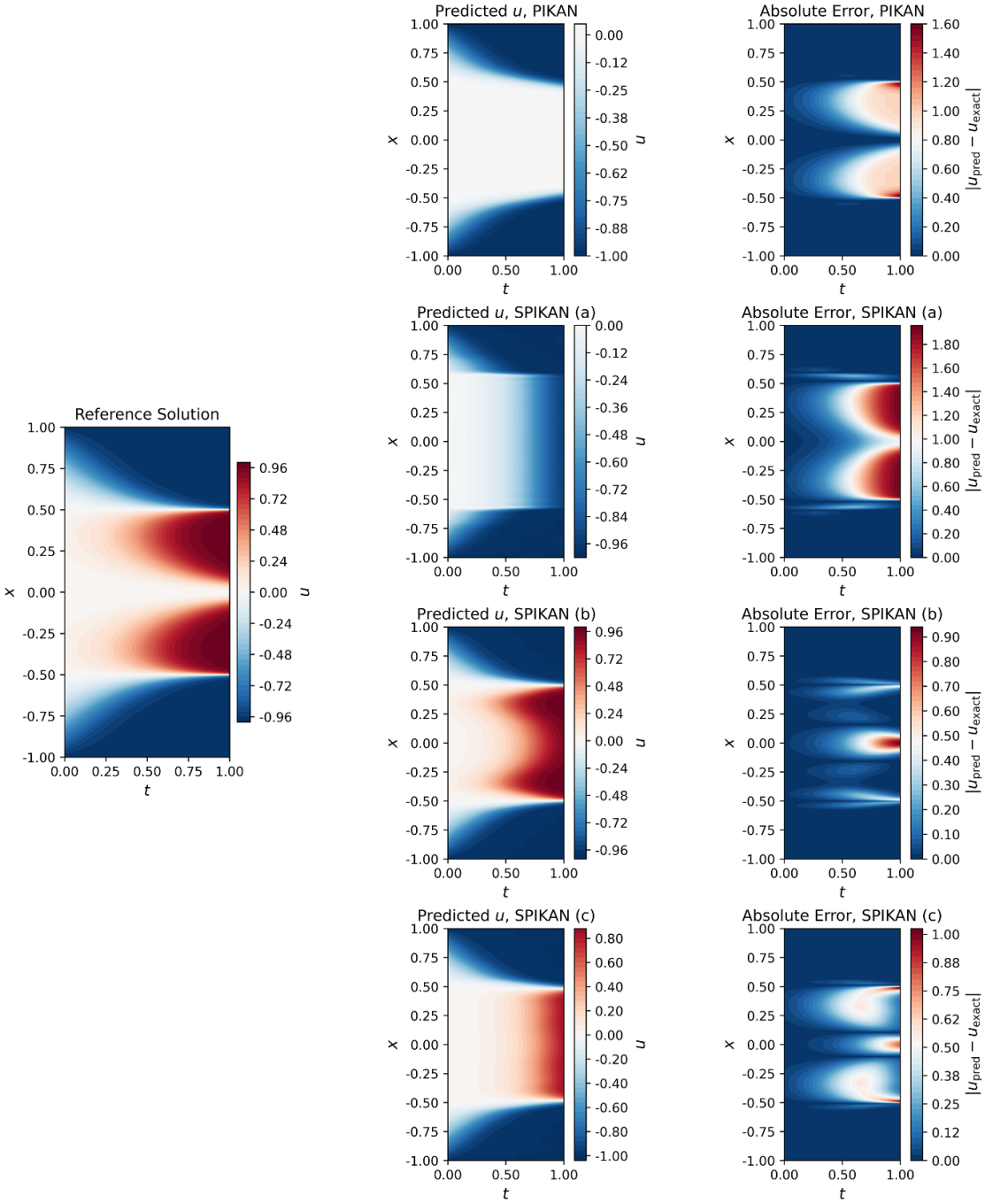

Figure 6. Isocontours of u of the reference solution, PIKAN ([2,9,9,1]) and SPIKAN cases (a) $2 \times [1, 5, 5, 1 * 5]$, (b) $2 \times [1, 5, 5, 1 * 10]$ and (c) $2 \times [1, 5, 5, 1 * 20]$, along with absolute errors for the 1D+1 Allen-Cahn initial-boundary value problem, equations (32)–(34). Implementation details are shown in table 3.

SPIKAN (b), the use of $r = 10$ drastically improves the accuracy of the results, allowing the correct range of u and reducing the absolute errors in the center of the spatial domain ($x \in [-0.5, 0.5]$ region). Despite the improvement for SPIKAN (b), the increase in latent dimension to $r = 20$ in SPIKAN (c) decreases the accuracy of the predictions, which can be caused by a slower learning process given that r increases the overall number of trainable parameters in the network; given that all the cases used a fixed number of iterations, it is possible that this effect would be reduced with more training epochs. We leave a comprehensive study of the impact of r in the stages of learning [29] for future work.

3.4. 2D+1 Klein–Gordon equation

We extend our analysis to the time-dependent, inhomogeneous Klein–Gordon equation in a two-dimensional spatial domain. As in [27], we use the governing equation:

Table 4. Comparison of methods for the 2D+1 Klein–Gordon initial-boundary value problem, equations (35)–(37).

Method	KAN Size	n_{cp}	No. Parameters	L_2 (%)	Time (ms/iter)	Speedup
PIKAN	[3,5,5,1]	50^3	371	2.16	847.45	1 (baseline)
SPIKAN (a)	$3 \times [1,3,3,1*10]$	100^3	704	1.00	3.20	264.83
SPIKAN (b)	$3 \times [1,5,5,1*10]$	150^3	1320	0.76	5.98	141.71
SPIKAN (c)	$3 \times [1,5,5,1*10]$	200^3	1320	0.79	7.82	108.37

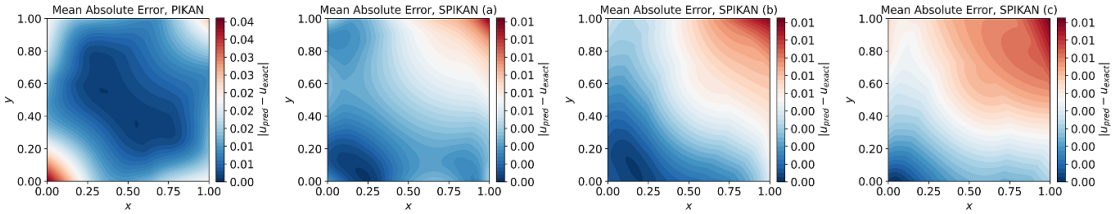

Figure 7. Isocontours of the mean absolute error, averaged over time, of the reference solution and PIKAN ([3,5,5,1]) and SPIKAN cases (a) $3 \times [1,3,3,1*10]$, (b) $3 \times [1,5,5,1*10]$ and (c) $3 \times [1,5,5,1*10]$, for different values of refinement n_{cp}, for the 2D+1 Klein–Gordon initial-boundary value problem, equations (35)–(37). Implementation details are shown in table 4. Note the different scale of the color plots of the absolute error for PIKAN and SPIKAN.

$$\frac{\partial^2 u}{\partial t^2} - \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + u^2 = h(x,y,t), \quad (x,y) \in \Omega,\ t \in \Gamma, \tag{35}$$

subject to the initial and boundary conditions:

$$u(x,y,0) = x + y, \quad (x,y) \in \Omega, \tag{36}$$

$$u(x,y,t) = u_{bc}(x,y), \quad (x,y) \in \partial\Omega,\ t \in \Gamma, \tag{37}$$

where the spatial domain is defined as $\Omega = [0,1]^2$ and the temporal domain as $\Gamma = [0,10]$. For validation purposes, a manufactured solution u

$$u(x,y,t) = (x+y)\cos(t) + xy\sin(t), \tag{38}$$

is used to derive the forcing term $h(x,y,t)$ and the boundary condition $u_{bc}(x,y)$. For this specific choice of u, h takes the form:

$$h(x,y,t) = u^2 - u. \tag{39}$$

Table 4 summarizes the results. The baseline network used for speedup calculations is a PIKAN of size [2,9,9,1], B-splines of degree $k = 3$, with $n_{cp} = 50^3$ collocation points. We note that above 60^3 collocation points, the memory requirement for PIKANs exceeds the vRAM capacity of the GPU used in this work. Such limitation can be alleviated with usage of larger GPUs or mini-batching strategies; the latter, however, comes with additional memory transfer overheads, which further increases the computational time needed for training.

To demonstrate the ability of collocation point refinement, we explore SPIKAN cases with larger values of n_{cp}. As in the PIKAN baseline, we set the degree of the B-splines $k = 3$. For all the cases, we used the Adam optimizer with initial learning rate of 10^{-3} for 50 000 epochs.

Figure 7 shows isocontours of the mean absolute errors, averaged over the time dimension. All the SPIKAN cases outperform the PIKAN baseline in accuracy and computational time. In this 3-dimensional setting, a speedup of $O(100)$ was obtained, even for SPIKAN (c), which has four times the number of trainable parameters of the baseline. The relative L_2 obtained in (c), however, shows a saturation of learning, indicating the need of larger KAN sizes to benefit of larger n_{cp}.

The temporal evolution of the relative L_2 is shown in figure 8. The instantaneous L_2 errors of the SPIKAN cases are lower than the PIKAN baseline. The periodicity of the curves reflects the choice of a periodic manufactured solution u; the spike formations that seem to grow over time, however, seem to be a consequence of the lack of spatio-temporal causality in the loss function formulation. A complete description of this limitation can be found in [45].

3.5. 2D steady lid-driven cavity flow with immersed cylinder

We demonstrate the application of SPIKANs to complex geometries using the immersed boundary approach described in section 2.3.2. Consider the steady, incompressible Navier–Stokes equations in a square cavity containing a circular cylinder:

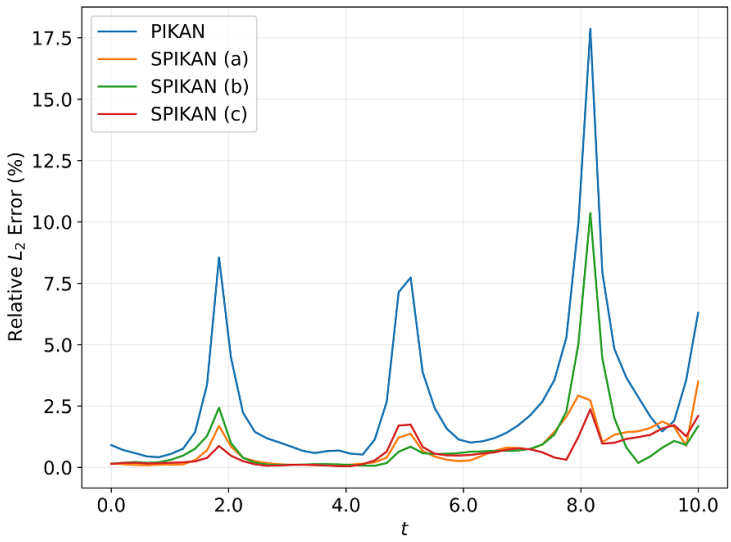

Figure 8. Temporal evolution of the L_2 error for the cases summarized in table 4 for the 2D+1 Klein–Gordon initial-boundary value problem, equations (35)–(37). SPIKAN predictions (a)–(c) achieved $O(100)$ speedup and yield results that are 50%+ more accurate in the L_2 norm versus the PIKAN baseline. The amplification of the periodic spikes reveal the difficulties associated with the lack of spatio-temporal causality, as discussed in [45].

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0, \tag{40}$$

$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{\partial p}{\partial x}+\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right), \tag{41}$$

$$u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{\partial p}{\partial y}+\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right), \tag{42}$$

where the computational domain is $\Omega_c=\Omega\setminus\Omega_b$, with $\Omega=[0,L]^2$ representing the square cavity and Ω_b the circular cylinder centered at $(x_c,y_c)=(0.5,0.5)$ with radius $r_c=0.2L$. We take $L=1$ and $U_0=1$.

The boundary conditions are specified as:

$$u(x,L)=U_0,\quad v(x,L)=0,\quad 0\leqslant x\leqslant L\quad\text{(top boundary, moving lid)}, \tag{43}$$

$$u(x,0)=0,\quad v(x,0)=0,\quad 0\leqslant x\leqslant L\quad\text{(bottom boundary, no-slip)}, \tag{44}$$

$$u(0,y)=0,\quad v(0,y)=0,\quad 0\leqslant y\leqslant L\quad\text{(left boundary, no-slip)}, \tag{45}$$

$$u(L,y)=0,\quad v(L,y)=0,\quad 0\leqslant y\leqslant L\quad\text{(right boundary, no-slip)}, \tag{46}$$

$$u=0,\quad v=0,\quad \text{on } \partial\Omega_b\quad\text{(cylinder surface, no-slip)},. \tag{47}$$

The implementation employs the masking strategy defined in equation (14), where the indicator function $\chi(\boldsymbol{x})$ excludes collocation points inside the cylinder from the physics residual computation. The cylinder boundary conditions are enforced through point-wise evaluation as described in equation (16), while the cavity walls maintain the separable structure due to their alignment with the coordinate axes.

The reference fields were obtained numerically using FVM. The steady-state SIMPLE [38] algorithm was employed to resolve the incompressible Navier–Stokes equations on a structured base mesh of 100×100 cells, with local refinement near the cylinder surface yielding a total of 9612 cells. Convective fluxes were approximated with a second-order linear upwind scheme with gradient-based correction, while the diffusive terms were discretized using a second-order Gauss linear scheme. Non-orthogonal corrections were applied to handle mesh distortion near the cylinder boundary. The iterative solver configuration utilized the Geometric Algebraic Multigrid method for the pressure field and a Gauss-Seidel smoother for the velocity field, with relaxation factors of 0.3 for pressure and 0.7 for velocity to enhance convergence stability. Convergence was achieved with target tolerances of 10^{-4} for both pressure and velocity components. For this problem, we consider Reynolds numbers $\mathrm{Re}=100$. The SPIKAN collocation points are arranged on uniform Cartesian grids of varying resolutions, with the masking strategy automatically handling the immersed cylinder geometry.

In addition to illustrating the use of SPIKANs in complex geometries, this example serves as a comprehensive sensitivity study. We investigate: (i) the effect of B-spline degree $k\in\{3,4,5\}$ on solution

Table 5. Comparison of PIKAN and SPIKAN methods for the 2D lid-driven cavity flow with immersed cylinder, equations (40)–(47) at Re = 100. The table presents: (i) PIKAN baseline results with B-spline degrees $k \in \{3,4,5\}$, (ii) SPIKAN sensitivity study for B-spline degree across three refinement levels, and (iii) SPIKAN sensitivity study for basis activation functions $b(x)$ with fixed $k=5$ and high resolution. For the speedup calculations, we used the best PIKAN case ($[2,9,9,3], k=5$) as baseline. Bold rows denote the best L_2 for u, v, p. All tests were performed using a NVIDIA A100 GPU with 40GB of vRAM.

Method	KAN Size	$b(x)$	k	n_{cp}	$L_2(u,v,p)$ (%)	Time (ms/iter)	Speedup
PIKAN baseline							
PIKAN	[2,9,9,3]	silu	3	100^2	(26.10, 31.04, 234.31)	24.12	2.25
PIKAN	[2,9,9,3]	silu	4	100^2	(24.77, 30.57, 347.14)	36.44	1.49
PIKAN	**[2,9,9,3]**	**silu**	**5**	**100^2**	**(19.80, 25.30, 56.29)**	**54.19**	**1.00 (baseline)**
SPIKAN: B-spline degree study (silu activation, $r=10$)							
SPIKAN	$2\times[1,5,5,3\times10]$	silu	3	100^2	(6.61, 13.81, 39.44)	1.77	30.62
SPIKAN	$2\times[1,5,5,3\times10]$	silu	3	400^2	(7.09, 12.63, 20.93)	3.80	14.26
SPIKAN	**$2\times[1,5,5,3\times10]$**	**silu**	**3**	**800^2**	**(5.80, 12.41, 19.76)**	**4.80**	**11.29**
SPIKAN	$2\times[1,5,5,3\times10]$	silu	4	100^2	(6.38, 13.35, 37.90)	2.30	23.56
SPIKAN	$2\times[1,5,5,3\times10]$	silu	4	400^2	(6.45, 12.54, 38.49)	5.66	9.58
SPIKAN	$2\times[1,5,5,3\times10]$	silu	4	800^2	(7.03, 13.72, 39.98)	8.06	6.72
SPIKAN	$2\times[1,5,5,3\times10]$	silu	5	100^2	(5.01, 9.44, 44.57)	2.94	18.43
SPIKAN	$2\times[1,5,5,3\times10]$	silu	5	400^2	(4.95, 9.76, 45.86)	7.21	7.52
SPIKAN	$2\times[1,5,5,3\times10]$	silu	5	800^2	(5.19, 9.96, 41.68)	11.71	4.63
SPIKAN: Basis activation study ($k=5$, $n_{cp}=800^2$, $r=10$)							
SPIKAN	$2\times[1,5,5,3\times10]$	**silu**	**5**	**800^2**	**(5.19, 9.96, 41.68)**	**11.71**	**4.63**
SPIKAN	$2\times[1,5,5,3\times10]$	tanh	5	800^2	(379.81, 246.66, 305.25)	11.87	4.57
SPIKAN	$2\times[1,5,5,3\times10]$	relu	5	800^2	(4034.09, 175.83, 124.89)	10.78	5.03
SPIKAN	$2\times[1,5,5,3\times10]$	sine	5	800^2	(260.10, 991.47, 558.01)	13.21	4.10

accuracy and computational cost, and (ii) the impact of the basis activation function $b(x)$ by comparing silu, tanh, relu, and sine functions while keeping the B-spline component fixed.

For all tested cases, the optimization was performed using the Adam optimizer with initial learning rate 10^{-3} for 400 000 epochs on a NVIDIA A100 GPU with 40GB of vRAM. For both PIKANs and SPIKANs, the collocation points are arranged in an equidistant manner on uniform Cartesian grids with n_{cp} points for various levels of refinement, with the cylinder boundary discretized using $n_{cp}^{\partial\Omega_b} = \sqrt{n_{cp}}$ points. For PIKAN, we use a mask to remove the points inside the cylinder, and reconstruct collocation points at the cylinder surface to represent the immersed boundary, as it is typically done in PINNs.

Table 5 summarizes the tested cases and results. The baseline network used for the speedup calculations is a PIKAN of size $[2,9,9,3]$ and B-spline degree $k=5$. This choice allows for a fair comparison with SPIKANs of network size $2\times[1,5,5,3\times10]$ in terms of number of trainable parameters. The results show that even at the smallest grid resolution (100^2), SPIKANs achieve substantially lower errors than PIKANs while training 18-31× faster. The separable architecture enables training on $n_{cp}=800^2$ with an 11.3× speedup compared to the best PIKAN case, demonstrating that SPIKANs can leverage finer resolutions that would be computationally prohibitive for traditional PIKANs. We clarify that differences in the computational time (ms/iter) when compared to the steady lid-driven cavity problem shown in table 2, are due primarily to the different GPU employed in this section. This choice was due to limitations in hardware availability, and does not affect the speedup ratios, which were computed consistently with previous sections.

The B-spline degree sensitivity reveals an interesting trade-off: while lower degrees ($k=3$) yield the best global L_2 errors for SPIKANs, the centerline profiles in figure 9 show that higher degrees improve local accuracy near walls and cylinder boundaries. This suggests that the separable formulation with lower-degree splines and finer grids provides optimal balance between computational efficiency and solution quality. SPIKANs demonstrate particularly superior boundary layer resolution near the cylinder, as evidenced in the regions approaching $x/L=0.3$ and $y/L=0.7$.

The base activation function study reveals that the choice of basis function $b(x)$ has a dramatic impact on accuracy. While silu maintains reasonable accuracy, all alternative activations tested lead to catastrophic failure, with errors exceeding 100% for the horizontal velocity component with relu. This critical difference underscores the importance of the silu activation in the KAN formulation, particularly for complex flow problems with immersed boundaries. These findings corroborate the observations of [3] that KANs, and by extension SPIKANs, require careful design choices to achieve robust optimization, with the silu basis function playing an essential role in maintaining numerical stability and convergence properties.

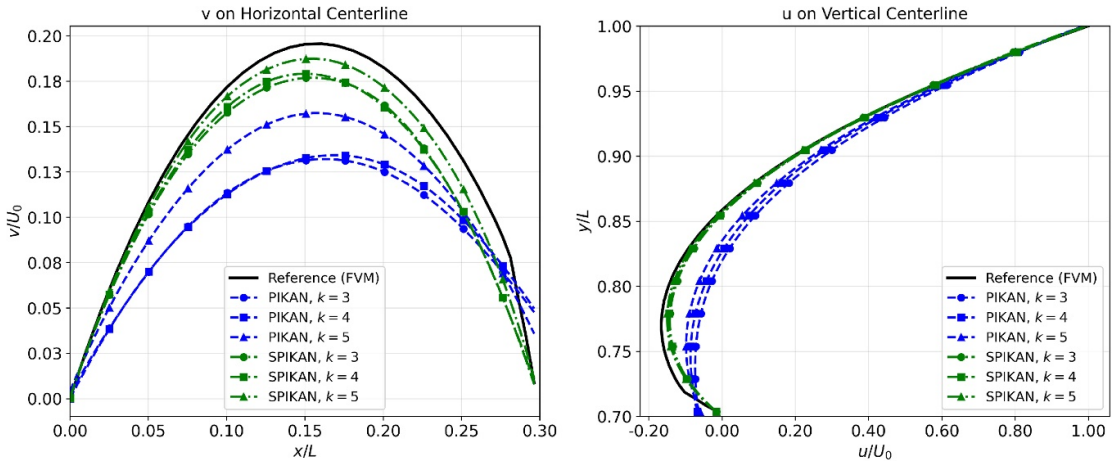

Figure 9. Centerline velocity profiles comparing PIKAN and SPIKAN predictions with the FVM reference for the lid-driven cavity flow with immersed cylinder at Re = 100, equations (40)–(47). Left: vertical velocity v along the horizontal centerline ($y = 0.5$) for $x \in [0, 0.3]$. Right: horizontal velocity u along the vertical centerline ($x = 0.5$) for $y \in [0.7, 1.0]$. Results are shown for varying B-spline degrees $k \in \{3, 4, 5\}$ with markers indicating the degree (circle: $k = 3$, square: $k = 4$, triangle: $k = 5$). Higher B-spline degrees improve predictions near walls and cylinder boundaries for both methods. The SPIKAN result with worst local error ($k = 3$) still outperforms the best PIKAN result ($k = 5$), in particular in regions near the boundary layer profiles.

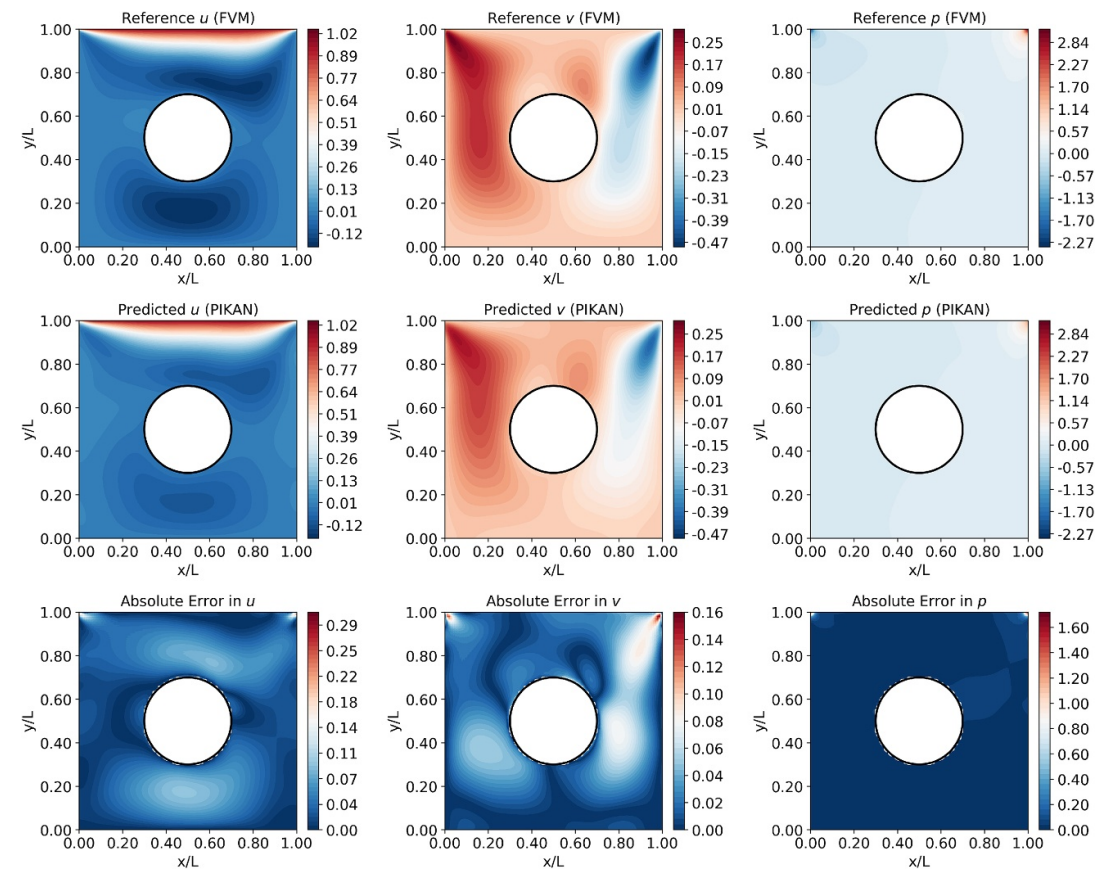

Figure 10. Contour plots comparing the reference FVM solution (top row) with the best PIKAN predictions ([2,9,9,3], $k = 5$, $n_{cp} = 100^2$, middle row) and absolute errors (bottom row) for the lid-driven cavity flow with immersed cylinder at Re = 100, equations (40)–(47). Fields shown are: horizontal velocity u (left), vertical velocity v (center), and pressure p (right). The cylinder is represented by the white circle at the domain center. Despite being the best PIKAN case tested, results exhibit significant errors near the boundaries and cylinder surface.

Figures 10 and 11 compare the velocity and pressure fields for the best PIKAN and SPIKAN configurations for the lid-driven cavity flow with immersed cylinder at Re = 100, equations (40)–(47). The absolute error plots demonstrate that SPIKAN's ability to train on an 800^2 grid yields substantially better accuracy, particularly near the immersed cylinder where gradients are steep. The streamline patterns in

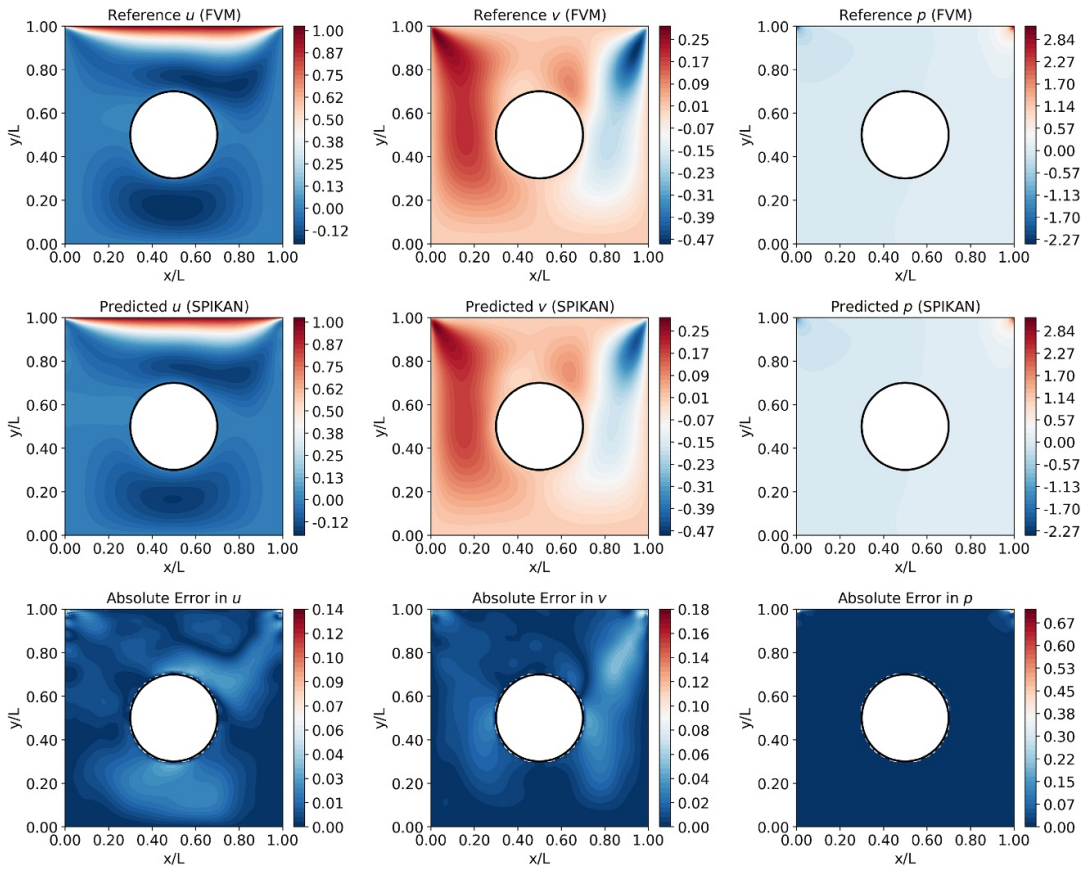

Figure 11. Contour plots comparing the reference FVM solution (top row) with the best SPIKAN predictions ($2 \times [1, 5, 5, 3 \times 10]$, $k = 3$, $n_{cp} = 800^2$, middle row) and absolute errors (bottom row) for the lid-driven cavity flow with immersed cylinder at Re = 100, equations (40)–(47). Fields shown are: horizontal velocity u (left), vertical velocity v (center), and pressure p (right). SPIKAN achieves significantly improved accuracy on a 800^2 grid that would be computationally prohibitive for PIKANs.

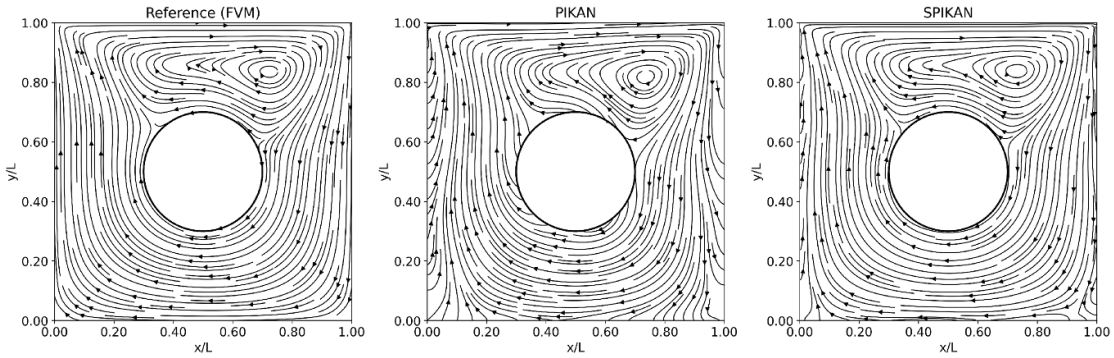

Figure 12. Streamline patterns for the lid-driven cavity flow with immersed cylinder at Re = 100, equations (40)–(47). Left: Reference solution from FVM showing the characteristic primary vortex and corner vortices. Center: Best (global) PIKAN prediction ([2,9,9,3], $k = 5$, $n_{cp} = 100^2$) exhibiting distorted flow patterns and spurious streamlines near boundaries. Right: Best (global) SPIKAN prediction ($2 \times [1, 5, 5, 3 \times 10]$, $k = 3$, $n_{cp} = 800^2$) demonstrating agreement with the reference, accurately capturing both the primary circulation and secondary flow features around the immersed cylinder.

figure 12 show that SPIKAN accurately captures the primary recirculating vortex and corner vortices, validating the effectiveness of the immersed boundary approach. These results confirm that SPIKANs handle complex geometries effectively while maintaining computational efficiency.

4. Conclusions

We have developed SPIKANs, an architecture for separable physics-informed Kolmogorov–Arnold networks. This method extends the work of [27] for KANs, significantly boosting the performance of these networks in both memory usage and computation time. Our tests demonstrate that SPIKAN predictions often yield more

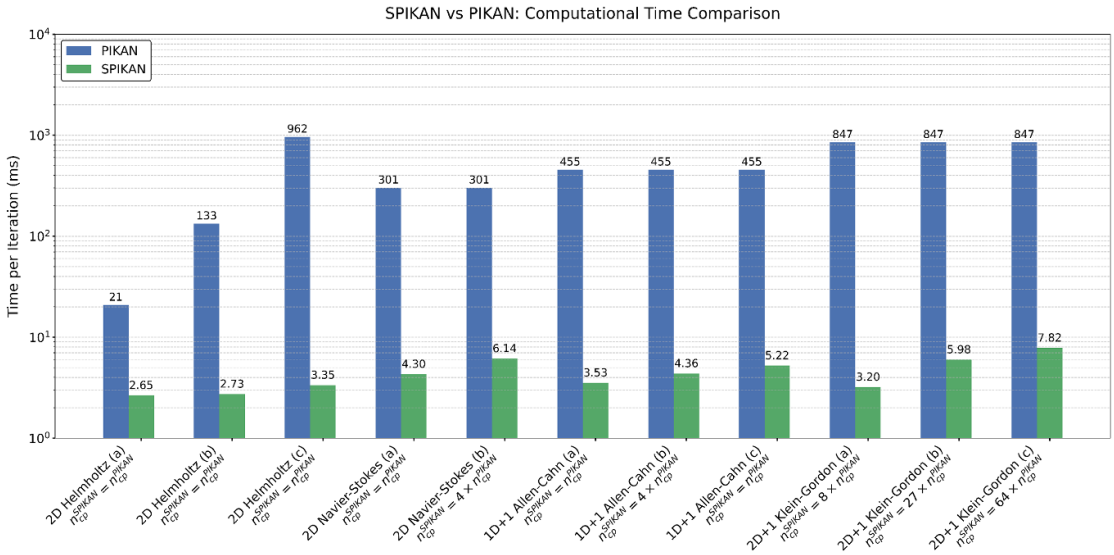

Figure 13. Comparison of computational time for all tests performed in section 3. A detailed description of each case is shown in tables 1–4.

accurate results than those of PIKANs for a comparable number of trainable parameters, while providing speedups of $O(10) - O(100)$ in several benchmark tests.

We summarize the computational times of all results in figure 13. In all tested cases, SPIKANs achieved speedups ranging from 8 $\times$ to 287 $\times$, while maintaining L_2 error comparable to or better than that obtained with PIKANs. This result demonstrates that even for small numbers of collocation points and network sizes, the use of SPIKANs remains advantageous. We emphasize that our intent is not for SPIKANs to compete with classical methods like FVM, that enjoy proven convergence and stability properties, but rather to alleviate the computational bottlenecks that have limited PIKAN applications and enable future research.

An important aspect of SPIKANs involves the need for a factorizable grid of collocation points. While this requirement can be limiting for certain applications, we believe that the use of factorizable grids along with immersed boundary forcing terms, as recently proposed in [46], can help alleviate this issue. Furthermore, by using partition of unity functions to train an ensemble of KANs, as introduced in [7], one could apply different factorizable grids in various areas of the domain. An important feature of SPIKANs is that they can be applied in addition to techniques for improving training and performance for KANs, including the use of other basis functions [16, 23], adaptive weighting such as residual-based attention [8, 29], and adaptive grid refinement [10].

In future work, we will further explore ways to improve the accuracy of this architecture by using stacking [47] and multi-fidelity training [43] and extensions to operator learning with DeepOKANs, along the lines of separable DeepONets [48]. In addition, while the universal approximation theorem introduced in this work guarantees the existence of SPIKANs that can achieve arbitrary accuracy, a comprehensive analysis of the relationship between rank r, B-spline grid size G, and approximation error for specific problem classes remains an important direction for future investigation. Such analysis would provide practical guidance for selecting these hyperparameters and establish *a priori* error bounds for the separable approximation.

Data availability statement

The data that support the findings of this study will be openly available following an embargo at the following URL/DOI: https://github.com/pnnl/spikans.

Acknowledgments

This project was completed with support from the U.S. Department of Energy, Advanced Scientific Computing Research program, under the Uncertainty Quantification for Multifidelity Operator Learning (MOLUcQ) project (Project No. 81739). The computational work was performed using PNNL Institutional Computing at Pacific Northwest National Laboratory. Pacific Northwest National Laboratory (PNNL) is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

Code and data availability

All code, trained models, and data required to replicate the examples presented in this paper will be released upon publication.

Appendix A. SPIKAN algorithm

Algorithm 1. Separable Physics-Informed Kolmogorov–Arnold Networks (SPIKANs).

Require: PDE operator $\mathcal{D}$ with source term f, domain $\Omega \subset \mathbb{R}^n$
Require: Initial operator $\mathcal{I}$ with initial data u_0
Require: Boundary operator $\mathcal{B}$ with boundary data g on $\partial\Omega$
Require: Rank r, B-spline order k, learning rate α, loss weights $\lambda_{\text{pde}}, \lambda_{\text{ic}}, \lambda_{\text{bc}}$
Require: Factorizable grid with N_j points in dimension j: $x^{(j)} = (x_1^{(j)}, \ldots, x_{N_j}^{(j)})$
Require: Immersed boundaries $\Gamma \subseteq \Omega$ (if any) with operator $\mathcal{B}_\Gamma$ and data h
Ensure: Trained parameters $\{\theta_j^*\}_{j=1}^n$

for $j = 1$ to n **do** ▷Initialize n univariate KANs
  Initialize KAN: $f^{(\theta_j)} : \mathbb{R} \to \mathbb{R}^{r \cdot w_{\text{out}}}$ with parameters θ_j
end for
if $\Gamma \neq \emptyset$ **then** ▷If immersed boundaries exist
  Define mask $M : \Omega \to \{0, 1\}$ where $M(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in \Omega \setminus \Gamma$
  Sample $\{\boldsymbol{x}_\Gamma^i\}_{i=1}^{N_\Gamma}$ on immersed boundary Γ
end if
while not converged **do**
  for $j = 1$ to n**do** ▷Evaluate each network on its dimension
    $f^{(\theta_j)}(x^{(j)}) \in \mathbb{R}^{N_j \times r \times w_{\text{out}}}$
  end for
  $\hat{u}(\boldsymbol{x}) = \sum_{k=1}^r \prod_{j=1}^n f_k^{(\theta_j)}(x_j)$ ▷Construct solution via rank-r decomposition
  Compute gradients of $\hat{u}$ as needed for $\mathcal{D}, \mathcal{I}, \mathcal{B}$ using forward-mode AD
  if $\Gamma = \emptyset$ **then** ▷Standard case: no immersed boundaries
    $\mathcal{L}_{\text{pde}} \leftarrow \frac{1}{N_{\text{pde}}} \sum_{i=1}^{N_{\text{pde}}} \left|\mathcal{D}[\hat{u}](\boldsymbol{x}_{\text{pde}}^i) - f(\boldsymbol{x}_{\text{pde}}^i)\right|^2$
  else ▷With immersed boundaries: use masking
    $N_{\text{pde}}^{\text{eff}} \leftarrow \sum_{\boldsymbol{x}} M(\boldsymbol{x})$
    $\mathcal{L}_{\text{pde}} \leftarrow \frac{1}{N_{\text{pde}}^{\text{eff}}} \sum_{\boldsymbol{x}} M(\boldsymbol{x}) \left|\mathcal{D}[\hat{u}](\boldsymbol{x}) - f(\boldsymbol{x})\right|^2$
  end if
  $\mathcal{L}_{\text{ic}} \leftarrow \frac{1}{N_{\text{ic}}} \sum_{i=1}^{N_{\text{ic}}} \left|\mathcal{I}[\hat{u}](\boldsymbol{x}_{\text{ic}}^i, 0) - u_0(\boldsymbol{x}_{\text{ic}}^i)\right|^2$ ▷Initial condition loss
  $\mathcal{L}_{\text{bc}} \leftarrow \frac{1}{N_{\text{bc}}} \sum_{i=1}^{N_{\text{bc}}} \left|\mathcal{B}[\hat{u}](\boldsymbol{x}_{\text{bc}}^i) - g(\boldsymbol{x}_{\text{bc}}^i)\right|^2$ ▷Outer boundary loss
  if $\Gamma \neq \emptyset$ **then** ▷Compute immersed boundary loss if applicable
    $\mathcal{L}_\Gamma \leftarrow 0$
    for $i = 1$ to N_Γ **do**
      $\hat{u}(\boldsymbol{x}_\Gamma^i) \leftarrow \sum_{k=1}^r \prod_{j=1}^n f_k^{(\theta_j)}(x_{\Gamma,j}^i)$ ▷Point-wise evaluation
      $\mathcal{L}_\Gamma \leftarrow \mathcal{L}_\Gamma + \left|\mathcal{B}_\Gamma[\hat{u}](\boldsymbol{x}_\Gamma^i) - h(\boldsymbol{x}_\Gamma^i)\right|^2$
    end for
    $\mathcal{L}_\Gamma \leftarrow \mathcal{L}_\Gamma / N_\Gamma$
    $\mathcal{L} \leftarrow \lambda_{\text{pde}}\mathcal{L}_{\text{pde}} + \lambda_{\text{ic}}\mathcal{L}_{\text{ic}} + \lambda_{\text{bc}}\mathcal{L}_{\text{bc}} + \lambda_\Gamma\mathcal{L}_\Gamma$
  else
    $\mathcal{L} \leftarrow \lambda_{\text{pde}}\mathcal{L}_{\text{pde}} + \lambda_{\text{ic}}\mathcal{L}_{\text{ic}} + \lambda_{\text{bc}}\mathcal{L}_{\text{bc}}$
  end if
  for $j = 1$ to n **do** ▷Update all network parameters
    $\theta_j \leftarrow \theta_j - \alpha \nabla_{\theta_j} \mathcal{L}$
  end for
end while
return $\{\theta_j^*\}_{j=1}^n$

Appendix B. Universal approximation property of SPIKANs

Theorem 1 (universal approximation property of SPIKANs). *Let $\Omega = \prod_{i=1}^n X_i$ be a compact product domain in $\mathbb{R}^n$. For any function $u \in L^2(\Omega, \mathbb{R})$ and any tolerance $\varepsilon > 0$, there exists a Separable Kolmogorov–Arnold Network (SPIKAN) $\hat{u}$ of the form*

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{r} \prod_{i=1}^{n} f_j^{(i)}(x_i) \tag{48}$$

such that

$$\|u-\hat{u}\|_{L^2(\Omega)}<\varepsilon, \tag{49}$$

where r is a sufficiently large rank and each $f_j^{(i)}$ is a univariate KAN (B-spline function) with a sufficiently fine grid.

Proof. The proof is structured in four steps, incorporating intermediate approximations to satisfy the prerequisites of each subsequent step, following the analytical approach of [27].

Step 1: Approximation by a Continuous Function. The space of continuous functions, $C(\Omega)$, is dense in the Hilbert space $L^2(\Omega,\mathbb{R})$. Therefore, for any $u\in L^2(\Omega,\mathbb{R})$ and any $\varepsilon>0$, we can first choose a continuous function $u_c\in C(\Omega)$ such that:

$$\|u-u_c\|_{L^2(\Omega)}<\frac{\varepsilon}{2}. \tag{50}$$

Step 2: Approximation by Separable Continuous Functions. Now we approximate the continuous function u_c. By the Stone-Weierstrass theorem [49], the algebra of finite sums of separable continuous functions (formed by products of univariate continuous functions) is dense in $C(\Omega)$ under the supremum norm, and thus also in $L^2(\Omega,\mathbb{R})$. We can therefore find a rank $r\in\mathbb{N}$ and a set of ideal continuous basis functions $\{g_j^{(i)}\in C(X_i)\}_{i=1,j=1}^{n,r}$ such that their sum of products, $u_g=\sum_{j=1}^{r}\prod_{i=1}^{n}g_j^{(i)}$, satisfies:

$$\left\|u_c-u_g\right\|_{L^2(\Omega)}<\frac{\varepsilon}{4}. \tag{51}$$

Since each $g_j^{(i)}$ is continuous on a compact set, it is bounded. Let $M=\max_{i,j}\left\|g_j^{(i)}\right\|_{L^\infty(X_i)}$.

Step 3: Univariate Approximation by KANs with Uniform Error Bounds. Here we use the fact that a univariate KAN is fundamentally a B-spline function (or a composition of them). By classical results from spline approximation theory, the set of B-spline functions is dense in the space of continuous functions $C(X_i)$ under the uniform norm. This is formally proven by error bounds showing that the approximation error vanishes as the grid size approaches zero [50, Ch XII, Eq (2)]. Therefore, for any ideal continuous function $g_j^{(i)}$ from Step 2 and any $\delta>0$, there exists a univariate KAN $f_j^{(i)}$ with a sufficiently fine grid such that:

$$\left\|g_j^{(i)}-f_j^{(i)}\right\|_{L^\infty(X_i)}<\delta. \tag{52}$$

This uniform bound directly implies that $\left\|f_j^{(i)}\right\|_{L^\infty(X_i)}\leqslant\left\|g_j^{(i)}\right\|_{L^\infty(X_i)}+\delta\leqslant M+\delta$. For convenience, we assume $\delta\leqslant 1$, so $\left\|f_j^{(i)}\right\|_{L^\infty(X_i)}\leqslant M+1$.

Step 4: Bounding the Total Approximation Error. Let the SPIKAN be $\hat{u}=\sum_{j=1}^{r}\prod_{i=1}^{n}f_j^{(i)}$. We bound the total error using the triangle inequality:

$$\|u-\hat{u}\|_{L^2(\Omega)}\leqslant\|u-u_c\|_{L^2(\Omega)}+\left\|u_c-u_g\right\|_{L^2(\Omega)}+\left\|u_g-\hat{u}\right\|_{L^2(\Omega)}<\frac{\varepsilon}{2}+\frac{\varepsilon}{4}+\left\|u_g-\hat{u}\right\|_{L^2(\Omega)}. \tag{53}$$

The final term is $\left\|\sum_{j=1}^{r}(\prod_{i=1}^{n}g_j^{(i)}-\prod_{i=1}^{n}f_j^{(i)})\right\|_{L^2(\Omega)}\leqslant\sum_{j=1}^{r}\left\|\prod_{i=1}^{n}g_j^{(i)}-\prod_{i=1}^{n}f_j^{(i)}\right\|_{L^2(\Omega)}$. To bound the norm for a single rank-1 term, we use a telescoping sum identity:

$$\prod_{i=1}^{n}g_i-\prod_{i=1}^{n}f_i=\sum_{k=1}^{n}\left(\prod_{i=1}^{k-1}f_i\right)(g_k-f_k)\left(\prod_{i=k+1}^{n}g_i\right). \tag{54}$$

Applying this identity within the norm, along with the triangle inequality, yields:

$$\left\|\prod_{i=1}^{n}g_j^{(i)}-\prod_{i=1}^{n}f_j^{(i)}\right\|_{L^2(\Omega)}\leqslant\sum_{k=1}^{n}\left\|\left(\prod_{i=1}^{k-1}f_j^{(i)}\right)\left(g_j^{(k)}-f_j^{(k)}\right)\left(\prod_{i=k+1}^{n}g_j^{(i)}\right)\right\|_{L^2(\Omega)} \tag{55}$$

$$\leqslant\sum_{k=1}^{n}\left(\prod_{i=1}^{k-1}\left\|f_j^{(i)}\right\|_{L^\infty(X_i)}\right)\left\|g_j^{(k)}-f_j^{(k)}\right\|_{L^2(\Omega)}\left(\prod_{i=k+1}^{n}\left\|g_j^{(i)}\right\|_{L^\infty(X_i)}\right) \tag{56}$$

$$\leqslant \sum_{k=1}^{n} \left(\prod_{i=1}^{k-1} \left\| f_j^{(i)} \right\|_{L^\infty(X_i)} \right) \left(\left\| g_j^{(k)} - f_j^{(k)} \right\|_{L^\infty(X_k)} \sqrt{\mathrm{Vol}(\Omega)} \right) \left(\prod_{i=k+1}^{n} \left\| g_j^{(i)} \right\|_{L^\infty(X_i)} \right) \tag{57}$$

$$\leqslant n(M+1)^{n-1} \sqrt{\mathrm{Vol}(\Omega)} \cdot \delta. \tag{58}$$

Here, equation (56) repeatedly applies the inequality $\|AB\|_{L^2(\Omega)} \leqslant \|A\|_{L^\infty(\Omega)} \|B\|_{L^2(\Omega)}$. Equation (57) uses the relation $\|h\|_{L^2(\Omega)} \leqslant \|h\|_{L^\infty(\Omega)} \sqrt{\mathrm{Vol}(\Omega)}$, where $\mathrm{Vol}(\Omega)$ is the volume of the domain, to convert the error term to the L-infinity norm established in Step 3. Equation (58) simplifies the expression using the constant bounds.

Let $C = n(M+1)^{n-1}\sqrt{\mathrm{Vol}(\Omega)}$, a global constant. We choose δ from Step 3 such that $\delta < \varepsilon/4rC$. Then the error for a single product term is bounded by $\varepsilon/4r$. Summing over all r terms:

$$\left\| u_g - \hat{u} \right\|_{L^2(\Omega)} \leqslant \sum_{j=1}^{r} \frac{\varepsilon}{4r} = \frac{\varepsilon}{4}. \tag{59}$$

Substituting this back into equation (53), we get the final bound:

$$\|u - \hat{u}\|_{L^2(\Omega)} < \frac{\varepsilon}{2} + \frac{\varepsilon}{4} + \frac{\varepsilon}{4} = \varepsilon. \tag{60}$$

□

Remark (on quantitative rates). This proof establishes existence. A quantitative error bound can be formulated by directly bounding the error as a sum of two terms: the tensor approximation error and the spline approximation error. The error for the best rank-r approximation of u is given by the $(r+1)$-th singular value, $\sigma_{r+1}(u)$, of the corresponding tensor operator. From classical spline theory, the error of approximating a sufficiently smooth (e.g. C^{k+1}) basis function with a spline of order $k+1$ (i.e. piecewise polynomial of degree k) on a grid with mesh size proportional to $1/G$ is $O(G^{-(k+1)})$ [50, ch XII]. This suggests a total error bound of the form:

$$\|u - \hat{u}\|_{L^2(\Omega)} \lesssim \sigma_{r+1}(u) + r \cdot C(n, M) \cdot G^{-(k+1)},$$

which highlights the trade-off between increasing rank r (to decrease the first term, related to the intrinsic separability of the function) and increasing grid size G (to decrease the second term, related to the complexity of the univariate components).

Remark (assumptions and limitations of the theorem). The universal approximation property of SPIKANs, as proven, rests on several assumptions and has important limitations. First, it assumes the target function u belongs to the Hilbert space $L^2(\Omega, \mathbb{R})$. While general, this excludes functions with non-square-integrable singularities. Second, and more critically, the proof is an existence theorem. It guarantees that for any $\varepsilon > 0$, a SPIKAN of sufficient rank r and univariate KAN complexity (grid density) *exists*, but it does not provide a constructive method for determining the minimal required r or the grid parameters for a given function u and tolerance ε. The constants involved in the error bounds depend on the specific (and unknown) ideal basis functions, meaning the theorem offers no *a priori* estimate of the model size needed for a given problem.

Note that this theorem only addresses the approximation error. It completely decouples from the two other critical sources of error in practice: estimation error arising from learning from finite data and optimization error. The proof implicitly assumes that the optimal set of KAN parameters can be found by the training algorithm. The existence of a good approximation does not guarantee its discoverability via gradient-based methods, which may encounter non-convex loss landscapes and convergence to local minima. Therefore, the theorem provides a guarantee of the model's expressive power, but not its trainability or generalization performance on a specific task.

ORCID iDs

Bruno Jacob 0009-0001-5361-3105
Amanda A Howard 0000-0002-6411-6198
Panos Stinis 0000-0002-9928-5637

References

[1] Raissi M, Perdikaris P and Karniadakis G E 2019 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
[2] Karniadakis G E, Kevrekidis I G, Lu L, Perdikaris P, Wang S and Yang L 2021 Physics-informed machine learning *Nat. Rev. Phys.* **3** 422–40
[3] Liu Z, Wang Y, Vaidya S, Ruehle F, Halverson J, Soljačić M, Hou T Y and Tegmark M 2024 Kan: Kolmogorov-Arnold networks (arXiv:2404.19756)
[4] Liu Z, Ma P, Wang Y, Matusik W and Tegmark M 2024 Kan 2.0: Kolmogorov-arnold networks meet science (arXiv:2408.10205)
[5] Vaca-Rubio C J, Blanco L, Pereira R and Caus M 2024 Kolmogorov-Arnold networks (KANs) for time series analysis (arXiv:2405.08790)
[6] Samadi M E, Müller Y and Schuppert A 2024 Smooth Kolmogorov-Arnold networks enabling structural knowledge representation (arXiv:2405.11318)
[7] Howard A A, Jacob B, Murphy S H, Heinlein A and Stinis P 2024 Finite basis Kolmogorov-Arnold networks: domain decomposition for data-driven and physics-informed problems (arXiv:2406.19662)
[8] Shukla K, Toscano J D, Wang Z, Zou Z and Karniadakis G E 2024 A comprehensive and fair comparison between mlp and kan representations for differential equations and operator networks *Comput. Methods Appl. Mech. Eng.* **431** 117290
[9] Abueidda D W, Pantidis P and Mobasher M E 2025 Deepokan: Deep operator network based on kolmogorov arnold networks for mechanics problems *Comput. Methods Appl. Mech. Eng.* **436** 117699
[10] Rigas S, Papachristou M, Papadopoulos T, Anagnostopoulos F and Alexandridis G 2024 Adaptive training of grid-dependent physics-informed kolmogorov-arnold networks *IEEE Access* (https://doi.org/10.1109/ACCESS.2024.3504962)
[11] Ranasinghe N, Xia Y, Seneviratne S and Halgamuge S 2024 Ginn-kan: interpretability pipelining with applications in physics informed neural networks (arXiv:2408.14780)
[12] Wang Y, Sun J, Bai J, Anitescu C, Eshaghi M S, Zhuang X, Rabczuk T and Liu Y 2025 Kolmogorov–arnold-informed neural network: A physics-informed deep learning framework for solving forward and inverse problems based on kolmogorov–arnold networks *Comput. Methods Appl. Mech. Eng.* **433** 117518
[13] Toscano J D, Käufer T, Maxey M, Cierpka C and Karniadakis G E 2024 Inferring turbulent velocity and temperature fields and their statistics from lagrangian velocity measurements using physics-informed kolmogorov-arnold networks (arXiv:2407.15727)
[14] Patra S, Panda S, Parida B K, Arya M, Jacobs K, Bondar D I and Sen A 2024 Physics informed kolmogorov-arnold neural networks for dynamical analysis via efficent-kan and wav-kan (arXiv:2407.18373)
[15] Shuai H and Li F Physics-informed kolmogorov-arnold networks for power system dynamics *IEEE Open Access J. Power Energy* **12** 46–58
[16] So C C and Yung S P 2024 Higher-order-relu-kans (HRKANs) for solving physics-informed neural networks (PINNs) more accurately, robustly and faster (arXiv:2409.14248)
[17] Gao Z and Karniadakis G E 2025 Scalable bayesian physics-informed kolmogorov-arnold networks (arXiv:2501.08501)
[18] Kashefi A and Mukerji T 2025 Physics-informed KAN pointnet: Deep learning for simultaneous solutions to inverse problems in incompressible flow on numerous irregular geometries (arXiv:2504.06327)
[19] Daryakenari N A, Shukla K and Karniadakis G E 2025 Representation meets optimization: training pinns and pikans for gray-box discovery in systems pharmacology (arXiv:2504.07379)
[20] Kiyani E, Shukla K, Urbán J F, Darbon J and Karniadakis G E 2025 Which optimizer works best for physics-informed neural networks and kolmogorov-arnold networks? (arXiv:2501.16371)
[21] Xu Z and Lv B 2025 Enhancing physics-informed neural networks with a hybrid parallel kolmogorov-arnold and mlp architecture (arXiv:2503.23289)
[22] Toscano J D, Oommen V, Varghese A J, Zou Z, Daryakenari N A, Wu C and Karniadakis G E 2025 From PINNs to PIKANs: Recent advances in physics-informed machine learning *Mach. Learn. Computat. Sci. Eng.* **1** 1–43
[23] Bozorgasl Z and Chen H 2024 Wav-kan: Wavelet Kolmogorov-Arnold networks (arXiv:2405.12832)
[24] Li Z 2024 Kolmogorov-Arnold networks are radial basis function networks (arXiv:2405.06721)
[25] Sidharth S S, Keerthana A R, Gokul R, Anas K P 2024 Chebyshev polynomial-based Kolmogorov-Arnold networks: An efficient architecture for nonlinear function approximation (arXiv:2405.07200)
[26] Yu T, Qiu J, Yang J and Oseledets I 2024 Sinc Kolmogorov-Arnold network and its applications on physics-informed neural networks (arXiv:2410.04096)
[27] Cho J, Nam S, Yang H, Yun S-B, Hong Y and Park E Separable physics-informed neural networks *Advances in Neural Information Processing Systems* vol 36
[28] McClenny L D and Braga-Neto U M 2023 Self-adaptive physics-informed neural networks *J. Comput. Phys.* **474** 111722
[29] Anagnostopoulos S J, Toscano J D, Stergiopulos N and Karniadakis G E 2024 Residual-based attention in physics-informed neural networks *Comput. Methods Appl. Mech. Eng.* **421** 116805
[30] Chen W, Howard A A and Stinis P 2025 Self-adaptive weights based on balanced residual decay rate for physics-informed neural networks and deep operator networks *J. Comput. Phys.* **542** 114226
[31] Wang S, Teng Y and Perdikaris P 2021 Understanding and mitigating gradient flow pathologies in physics-informed neural networks *SIAM J. Sci. Comput.* **43** A3055–81
[32] Wang S, Sankaran S, Wang H and Perdikaris P 2023 An expert's guide to training physics-informed neural networks (arXiv:2308.08468)
[33] Wang S, Yu X and Perdikaris P 2022 When and why pinns fail to train: a neural tangent kernel perspective *J. Comput. Phys.* **449** 110768
[34] Cui S, Cao M, Liao Y and Wu J 2025 Physics-informed kolmogorov–arnold networks: Investigating architectures and hyperparameter impacts for solving navier–stokes equations *Phys. Fluids* **37** 3
[35] Jacob B, Howard A A and Stinis P 2025 Separable physics-informed Kolmogorov–Arnold networks (available at: https://github.com/pnnl/spikans)
[36] Bradbury J *et al* 2018 JAX: composable transformations of Python+NumPy programs (available at: http://github.com/google/jax)
[37] Rigas S and Papachristou M 2024 jaxKAN: A JAX-based implementation of Kolmogorov-Arnold networks (available at: https://github.com/srigas/jaxKAN)
[38] Patankar S V and Spalding D B 1983 A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows *Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion* (Elsevier) pp 54–73

[39] Ghia U, Ghia K N and Shin C 1982 High-re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method *J. Comput. Phys.* **48** 387–411
[40] Wight C L and Zhao J 2020 Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks (arXiv:2007.04542)
[41] Mattey R and Ghosh S 2022 A novel sequential method to train physics informed neural networks for Allen-Cahn and Cahn-Hilliard equations *Comput. Methods Appl. Mech. Eng.* **390** 114474
[42] Rohrhofer F M, Posch S, Gößnitzer C and Geiger B C 2022 On the role of fixed points of dynamical systems in training physics-informed neural networks (arXiv:2203.13648)
[43] Howard A, Fu Y and Stinis P 2024 A multifidelity approach to continual learning for physical systems *Mach. Learn.: Sci. Technol.* **5** 025042
[44] Heinlein A, Howard A A, Beecroft D and Stinis P 2024 Multifidelity domain decomposition-based physics-informed neural networks for time-dependent problems (arXiv:2401.07888)
[45] Wang S, Sankaran S and Perdikaris P 2022 Respecting causality is all you need for training physics-informed neural networks (arXiv:2203.07404)
[46] Sundar R, Majumdar D, Lucor D and Sarkar S 2024 Physics-informed neural networks modelling for systems with moving immersed boundaries: Application to an unsteady flow past a plunging foil *J. Fluids Struct.* **125** 104066
[47] Howard A A, Murphy S H, Ahmed S E and Stinis P 2023 Stacked networks improve physics-informed training: applications to neural networks and deep operator networks (arXiv:2311.06483)
[48] Mandl L, Goswami S, Lambers L and Ricken T 2025 Separable physics-informed deeponet: Breaking the curse of dimensionality in physics-informed machine learning *Comput. Methods Appl. Mech. Eng.* **434** 117586
[49] Rudin W 1976 *Principles of mathematical analysis* 3rd edn (McGraw-Hill Inc)
[50] De Boor C 2001 *A Practical Guide to Splines* (Springer)